UNIVERSIDAD MAYOR DE SAN SIMÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

DIRECCIÓN DE POSGRADO

DIPLOMADO CIENCIA DE DATOS

TERCERA VERSIÓN

MODELO DE PREDICCION DE INGRESOS POR LA COBRANZA DE FACTURAS DE LA EPSA "THIKA KHATU"

MONOGRAFIA PRESENTADA PARA OBTENER EL GRADO DE LICENCIATURA EN INGENIERÍA EN INFORMÁTICA
MODALIDAD DOBLE TITULACIÓN

POSTULANTE : JAVIER ERICK MARTÍNEZ BARRIGA
TUTOR : ING. MARWIN ANTONIO TORREZ MONTAÑO

Cochabamba – Bolivia
2025

MODELO DE PREDICCION DE INGRESOS POR LA COBRANZA DE FACTURAS DE LA EPSA “THIKA KHATU”

Por

Javier Erick Martínez Barriga

El presente documento, Trabajo de Grado es presentado a la Dirección de Posgrado de la Facultad de Ciencias y Tecnología en cumplimiento parcial de los requisitos para la obtención del grado académico de Licenciatura en Ingeniería Informática, modalidad Doble Titulación, habiendo cursado el Diplomado “Ciencia de Datos” propuesta por el Centro de Estadística Aplicada (CESA) en su segunda versión.

ASESOR/TUTOR

Ing. Marwin Antonio Torrez Montaño

COMITÉ DE EVALUACIÓN

Ing. M.Sc. Patiño Tito Ronald Edgar. (Presidente)
Ing. M.Sc Guillen Salvador Roxana. (Coordinador)
Ing. M.Sc Espinoza Orosco José. (Tribunal)
Lic. Garcia Perez Carmen Rosa. (Tribunal)

DIRECCIÓN DE POSGRADO
FACULTAD DE CIENCIAS Y TECNOLOGÍA

DIRECCIÓN DE POSGRADO, FACULTAD DE CIENCIAS Y TECNOLOGIA
Cochabamba, Bolivia

Aclaración

Este documento describe el trabajo realizado como parte del programa de estudios de Diplomado “Ciencia de Datos” en el Centro de Estadística Aplicada CESA y la Dirección de Posgrado de la Facultad de Ciencias y Tecnología. Todos los puntos de vista y opiniones expresadas en el mismo son responsabilidad exclusiva del autor y no representa necesariamente a la institución.

Resumen

En el contexto actual de las empresas prestadoras de servicios de agua (EPSA), garantizar una gestión financiera eficiente y sostenible se ha vuelto una necesidad prioritaria, especialmente en entornos donde los recursos son limitados y la planificación a largo plazo depende de ingresos estables. La recaudación por cobro de facturas representa una de las principales fuentes de financiamiento para estas entidades, por lo que anticipar sus variaciones puede mejorar significativamente la toma de decisiones estratégicas. En este marco, el presente trabajo se plantea como una propuesta de solución basada en técnicas de aprendizaje automático, específicamente en modelos de series de tiempo, con el fin de estimar de manera precisa los ingresos futuros y así contribuir a una administración más eficaz de los recursos económicos de la entidad.

Inicialmente, se realizó una exploración profunda del conjunto de datos proporcionado por la EPSA, que incluye información diaria sobre consumos, facturación y pagos de los usuarios. A continuación, se aplicaron procesos de limpieza y transformación de los datos, como la imputación de valores nulos, el tratamiento de valores atípicos y la normalización mediante escalado Min-Max, con el objetivo de preparar los datos para su posterior modelado. También se descartaron variables irrelevantes para mejorar la eficiencia del análisis y se definió una estrategia de división entre datos de entrenamiento y prueba.

Posteriormente, se implementaron dos alternativos de modelos predictivos de series de tiempo, las redes neuronales LSTM y el modelo Prophet, conocidas por su capacidad para capturar patrones temporales complejos y adaptarse a secuencias con variabilidad significativa. Las métricas empleadas para medir la precisión del modelo fueron RMSE, MAE y MAPE. Finalmente, se seleccionó la configuración de cada modelo con mejor rendimiento y se las comparó con los resultados de un trabajo de referencia con características similares, con el propósito de contextualizar y validar el desempeño del modelo desarrollado.

Los resultados obtenidos muestran que, aunque el modelo Prophet logra el mejor desempeño en términos de MAPE (17.82%), el modelo LSTM supera a Prophet en las métricas MAE (231.96) y RMSE (342.58). Esto indica que el LSTM ofrece predicciones más cercanas a los valores reales en promedio, aunque con mayor variabilidad relativa. En particular, se observó que, al aumentar el número de pasos temporales, el rendimiento del LSTM tiende a disminuir, destacando la importancia de una adecuada configuración de este parámetro. Estos hallazgos refuerzan la eficacia del modelo LSTM para predecir ingresos por facturación, especialmente en contextos donde la precisión absoluta es prioritaria.

Palabras clave: EPSA, Machine Learning, series temporales, redes neuronales, RNN, LSTM

Dedico este trabajo a mi familia, por su apoyo incondicional, su paciencia y sus palabras de aliento en los momentos más difíciles. A mis padres, por enseñarme el valor del esfuerzo y la perseverancia; y a todas aquellas personas que, con su confianza y cariño, me impulsaron a seguir adelante y dar lo mejor de mí en cada etapa de este camino.

Agradecimientos

A mis padres, por ser el pilar fundamental de mi vida. Gracias por su amor, sacrificio y por brindarme siempre el apoyo necesario para seguir adelante, incluso en los momentos más difíciles.

A mi familia, por su constante ánimo, por creer en mí y por acompañarme con cariño y comprensión a lo largo de todo este proceso académico.

A mi tutor, por su guía, paciencia y compromiso durante el desarrollo de este trabajo. Sus observaciones y aportes fueron clave para mejorar la calidad del proyecto.

A mi enamorada, por su apoyo incondicional, por motivarme a dar lo mejor de mí y por estar presente en cada paso de este camino.

A los docentes del diplomado, por compartir sus conocimientos con claridad y pasión, y por fomentar un entorno de aprendizaje enriquecedor y desafiante.

A la EPSA Thika Khatu, por facilitar el acceso a los datos necesarios para este estudio y por permitir que este trabajo tenga una aplicación práctica en un contexto real.

Tabla de contenidos

Lista de figuras

Lista de tablas

1. Introducción

El acceso al agua potable es un derecho humano fundamental reconocido por las Naciones Unidas, pero aún existen grandes desigualdades en su disponibilidad y distribución. Mientras que en algunas regiones el agua es abundante y fácilmente accesible, en otras comunidades, especialmente en zonas rurales y en países en desarrollo, obtener agua limpia sigue siendo un desafío diario. Factores como el crecimiento poblacional, el cambio climático y la contaminación han agravado la escasez de agua en diversas partes del mundo. Además, la falta de infraestructura adecuada y la gestión ineficiente de los recursos hídricos dificultan el suministro continuo y seguro de agua para la población. (Nieto, 2011)

En Bolivia, el acceso al agua potable ha sido un desafío histórico, especialmente en áreas rurales y periurbanas. Si bien en los últimos años se han realizado esfuerzos significativos para mejorar la cobertura del servicio, aún persisten problemas relacionados con la disponibilidad, calidad y gestión del recurso. Además, conflictos sociales relacionados con la privatización del agua, como la "Guerra del Agua" en Cochabamba en el año 2000, han evidenciado la importancia de una gestión pública y equitativa del recurso.

Existen empresas públicas en cada departamento de Bolivia encargadas del suministro de agua potable a sus ciudadanos, como SAMAPA en La Paz, SEMAPA en Cochabamba o SAGUAPAC en Santa Cruz. Aunque estas empresas abarcan a un porcentaje de la población, no tienen una cobertura total, lo que conlleva a que los pobladores de algunas comunidades, zonas e incluso barrios tengan que optar por otros medios para abastecerse de agua. (Kruse, 2005)

En Cochabamba, el suministro de agua proviene principalmente de represas. Sin embargo, en varias zonas existen pozos o vertientes de agua natural que son aprovechadas por las comunidades cercanas. En particular, varios distritos del norte de la ciudad utilizan estas fuentes para abastecerse, lo que ha llevado a la creación de redes de distribución propias o particulares. Esto implica no solo la gestión del sistema de distribución, sino también la administración de los recursos económicos necesarios para su mantenimiento. No obstante, muchas de estas pequeñas entidades enfrentan dificultades tanto en la gestión del recurso hídrico como en la administración financiera, debido a la falta de una estructura organizativa sólida. (Kruse, 2005)

En este contexto, la propuesta de este proyecto es aplicar modelos de aprendizaje automático basados en series de tiempo para analizar los datos históricos de facturación de estas pequeñas organizaciones. Mediante este enfoque, se identificarán patrones y tendencias en el comportamiento de los ingresos económicos derivados de la cobranza de facturas, lo que permitirá realizar predicciones más precisas y tomar decisiones más idóneas respecto a la administración de estas entidades.

1.1. Antecedentes

En Bolivia, la provisión de servicios básicos, especialmente el acceso al agua potable, sigue siendo un desafío en muchas regiones debido a limitaciones en infraestructura y gestión. Si bien, el Gobierno Nacional indica que la cobertura de agua en el 2023 habría superado el 87,9 % (95,5 % de cobertura en el área urbana y 69,4 % en el ámbito rural) y en el caso del saneamiento superaría el 72,1 % (Carballo, 2024), la realidad era distinta décadas atrás cuando las ciudades estaban desarrollando el acceso a los servicios básicos. Se estima que, en 1992, la cobertura de agua potable era del 57,2% y el saneamiento alcanzaba el 28%. (UDAPE, 2013)

Debido a las limitaciones del sector público en Bolivia para proporcionar servicios básicos e infraestructura en el área metropolitana de Cochabamba, numerosas organizaciones vecinales han implementado estrategias de autogestión para suplir estas carencias, especialmente en el acceso al agua potable. Estas iniciativas son llevadas a cabo por pequeños operadores locales, conocidos como Entidades Prestadoras de Servicios de Agua Potable (EPSA), que suelen estar conformados por organizaciones barriales con el objetivo principal de garantizar el suministro de agua a sus comunidades. Para ello, administran y mantienen redes y sistemas de distribución propios, contribuyendo así a asegurar un acceso equitativo y continuo a este recurso esencial.

Es deber del Estado es gestionar, regular, proteger y planificar el uso adecuado y sostenible de los recursos hídricos, con participación social, garantizando el acceso al agua a todos sus habitantes. Por lo tanto, el Estado, delega esta función a las EPSA (Ley N° 2066, 2000) a través de Licencias y Registros otorgadas por la Autoridad de Fiscalización y Control Social de Agua Potable y Saneamiento Básico (AAPS). Estas deben brindar un servicio sostenible y de calidad en la captación, transporte, almacenamiento, tratamiento y distribución del agua a la población dentro del área de prestación del servicio, autorizada por la AAPS (Decreto Supremo Nº 0071, 2009)

La relación entre la gestión del agua y las organizaciones vecinales en Bolivia se remonta a inicios de la década de 1980. Sin embargo, esta relación se fortaleció a principios de los años 2000, cuando el proceso de privatización de los servicios básicos se intensificó con la promulgación de la Ley N° 2029 de 1999. En Cochabamba, la privatización implicó la transferencia del Servicio Municipal de Agua Potable y Alcantarillado (SEMAPA) y sus activos a la empresa estadounidense Bechtel. Lo más relevante de esta ley es que, además de fomentar la transferencia de infraestructura de servicios a empresas privadas, también permitía la concesión de fuentes de agua superficiales y subterráneas a esta compañía. Esto generó un fuerte conflicto social, conocido como la "Guerra del Agua", ante el riesgo de que las organizaciones vecinales y sus EPSA perdieran el derecho a extraer agua subterránea, lo que amenazaba la supervivencia de los operadores locales. La población logró desplazar a la empresa y una nueva Ley (N° 2066) que otorgó a las juntas vecinales derechos administrativos, económicos y de gestión del recurso agua, bajo un marco legal que reconoce al acceso a este recurso como derecho humano. (Barrera Cordero, 2009)

La región metropolitana de Cochabamba, conformada por los municipios de Cochabamba, Quillacollo, Sacaba, Vinto, Sipe Sipe, Colcapirhua y Tiquipaya, alberga aproximadamente 1,2 millones de habitantes (INE, 2016). De esta población, el 95 % cuenta con acceso a servicios de agua (OCNU, 2015); sin embargo, solo entre el 25 % y 30 % recibe el suministro a través de operadores públicos. El resto depende de centenares de EPSA, que extraen agua de pozos subterráneos y distribuyen el recurso en gran parte del valle. Según estimaciones de 2013, en la jurisdicción metropolitana podrían existir alrededor de 1400 pozos, aunque el Gobierno Autónomo Municipal de Quillacollo y el Plan Municipal de Ordenamiento Territorial (2016) identificaron más de 500 solo en ese municipio. (Cabrera, 2018)

No obstante, las EPSA enfrentan importantes desafíos que afectan la sostenibilidad de los servicios que prestan. Entre las principales dificultades se encuentran las limitaciones técnicas y financieras, especialmente en las zonas periurbanas, donde la demanda de agua sigue en aumento. Además, la infraestructura existente resulta insuficiente para abastecer a una población en constante crecimiento. A estos problemas se suman los efectos del cambio climático, el crecimiento demográfico acelerado y el alto índice migratorio, factores que imponen nuevos retos a la gestión y distribución del recurso. (Marston, 2014)

A lo largo de los años, debido a la importancia que tiene el agua en Cochabamba, diversas investigaciones han abordado distintas problemáticas respecto a este recurso. Existen trabajos de investigación sobre el tratamiento de aguas, como el que se realizó en el 2020 por estudiantes de la universidad de San Simón en el cual estudiaron la eficiencia que tienen las plantas de tratamiento de aguas residuales en Cochabamba (Mercado Guzman, Cosio Grageda, & Copa Mitma, 2020). También la calidad del agua es un tema de investigación, como la que se realizó el 2006 por estudiantes del departamento de ingeniería de la universidad Católica Boliviana, enfocada en la evaluación de la calidad de las aguas del rio Rocha en Cochabamba (Toledo Medrano & Amurrio Derpic, 2006). Y de la misma manera, la exploración de aguas subterráneas tiene relevancia en Cochabamba ya que se encontraron diversas fuentes a lo largo de la región, como se evidencia en el proyecto realizado para una tesis de maestría en la universidad San Francisco Xavier de Chuquisaca, en la que destaca el potencial de explotación de agua subterránea en Cliza, Cochabamba (Zapata, 2015).

Aunque en Cochabamba se han desarrollado diversas investigaciones sobre el agua, no se han encontrado estudios que utilicen modelos de predicción basados en series de tiempo para estimar el ingreso económico que se genera con este recurso. Sin embargo, esta metodología ha sido aplicada en otros ámbitos, como en la predicción de heladas meteorológicas en Cochabamba, donde un estudio realizado en 2016 por estudiantes de la Universidad Católica Boliviana destacó el uso de máquinas de aprendizaje extremo (Riabani Mercado, Garcia Fernandez, & Herrera Acebey, 2016). De manera similar, en 2011, un estudiante de la misma universidad utilizó modelos ARIMA y SARIMA para analizar la demanda de potencia eléctrica en Bolivia (Sanjines Tudela, 2011).

1.2. Justificación

Como se explicó previamente, el acceso al agua potable es un derecho fundamental y un recurso esencial para el bienestar de la población. Sin embargo, en Cochabamba, la provisión de este servicio depende en gran medida de las EPSA, las cuales enfrentan múltiples desafíos técnicos, financieros y operativos. La falta de infraestructura suficiente, las variaciones en la demanda, el impacto del cambio climático y el crecimiento poblacional han generado una creciente presión sobre estos pequeños operadores, dificultando la sostenibilidad del servicio.

En este contexto, el presente proyecto busca desarrollar modelos de predicción basados en series de tiempo que permita estimar los ingresos por facturación de estas entidades. Su implementación beneficiaría a las EPSA mejorando la capacidad de previsión financiera, lo que facilitaría la mejor planificación de sus presupuestos, reducir la incertidumbre económica y garantizar la sostenibilidad de sus operaciones a largo plazo.

Finalmente, la mejora en la gestión del agua a través de la predicción contribuiría a la sostenibilidad del servicio y a la reducción de conflictos entre las comunidades que dependen de estos operadores. Al contar con una planificación más eficiente, las EPSA podrían garantizar un acceso más equitativo y permanente al recurso, fortaleciendo su rol dentro del sistema de distribución de agua en Cochabamba.

En definitiva, este proyecto representa una herramienta estratégica para mejorar la eficiencia operativa de las EPSA, asegurando un servicio más estable y sostenible, y contribuyendo al bienestar de la población que depende de ellas.

1.3. Planteamiento del problema

Las EPSA enfrentan múltiples desafíos en la gestión del recurso hídrico, que afectan tanto la eficiencia operativa como la sostenibilidad financiera del servicio. La incertidumbre en la planificación financiera es un problema crítico para las EPSA, ya que su operatividad depende directamente de los ingresos generados por la facturación del servicio. Sin una proyección clara de estos ingresos, las EPSA enfrentan dificultades para administrar sus recursos de manera eficiente, lo que afecta la planificación a corto y largo plazo. Esta falta de previsión repercute en la toma de decisiones relacionadas con el mantenimiento de la infraestructura, ya que no contar con un estimado preciso de los ingresos puede llevar a postergar reparaciones o mejoras necesarias en la red de distribución, aumentando el riesgo de fallas operativas. Sin una estimación confiable de los ingresos futuros, las EPSA no pueden determinar con certeza si cuentan con la capacidad económica para invertir en la ampliación de redes o perforación de nuevos pozos, lo que restringe su crecimiento y capacidad de cobertura.

Otro aspecto afectado es la sostenibilidad económica de estas entidades. Al no disponer de herramientas para prever fluctuaciones en los pagos de los usuarios, las EPSA pueden experimentar periodos de déficit financiero, dificultando la continuidad del servicio. Esto se agrava en casos donde existen altos índices de morosidad o cambios en el consumo de agua que afectan la facturación esperada. En consecuencia,

la falta de planificación financiera puede comprometer la estabilidad de estas organizaciones y poner en riesgo la provisión constante del recurso hídrico a la población.

A pesar de contar con registros históricos de facturación, consumo y pagos, muchas EPSA no aprovechan plenamente esta información para realizar análisis predictivos que les permitan anticiparse a escenarios financieros adversos. La ausencia de herramientas analíticas y modelos de predicción limita su capacidad de tomar decisiones basadas en datos, lo cual es esencial para gestionar eficientemente un recurso tan vital como el agua. Actualmente, la mayoría de estas entidades operan bajo esquemas reactivos, en los que las decisiones financieras y operativas se toman en función de la disponibilidad inmediata de recursos, sin contar con estimaciones confiables sobre el comportamiento futuro del ingreso. Esto no solo afecta su capacidad para enfrentar imprevistos, sino que también debilita su planificación estratégica a mediano y largo plazo.

En este contexto, la implementación de modelos estadísticos de predicción, especialmente aquellos basados en series de tiempo, representa una oportunidad para dotar a las EPSA de herramientas que les permitan proyectar el ingreso por facturación con mayor precisión. Estas estimaciones pueden facilitar la planificación financiera, optimizar la asignación de recursos y contribuir a la sostenibilidad del servicio.

1.4. Objetivo general

Desarrollar un modelo de predicción de aprendizaje automático, basado en series de tiempo para estimar la recaudación de ingresos por cobro de facturas de la EPSA "Thika Khatu", contribuyendo a una administración más eficaz del recurso y a la mejora de la sostenibilidad financiera de la entidad.

1.4.1. Objetivos específicos

- Tratar los datos mediante procesos de limpieza, transformación y depuración para garantizar su calidad, consistencia y estructura antes del análisis.
- Construir una serie de tiempo a partir de los datos originales mediante un proceso de agrupación, transformación y estructuración temporal que permita su análisis y modelado adecuado.
- Implementar modelos predictivos de series de tiempo adecuados para estimar el consumo futuro de agua, considerando las características del conjunto de datos.
- Evaluar el desempeño de las predicciones de los modelos aplicando métricas de validación que permitan evaluar su precisión, confiabilidad y utilidad en contextos reales.

2. Marco teórico

En este capítulo se exponen de forma muy puntual los conceptos, metodologías y otros elementos esenciales que serán aplicados en el desarrollo del proyecto. Estos conceptos no solo actuarán como los pilares teóricos de nuestro enfoque, sino que también servirán como las directrices que darán forma y sustancia a nuestro análisis y toma de decisiones.

2.1. Análisis predictivo

El análisis predictivo consiste en la tecnología que aprende de la experiencia para prever el comportamiento futuro de los individuos y facilitar la toma de decisiones. En términos más generales, esta disciplina forma parte de la minería de datos y se centra en la extracción de información para identificar patrones y tendencias que permitan predecir eventos desconocidos, ya sean pasados, presentes o futuros. Su fundamento radica en la detección de relaciones entre variables en eventos previos, las cuales pueden ser aprovechadas para estimar posibles resultados en nuevas situaciones. Sin embargo, la precisión de estas predicciones depende en gran medida de la calidad del análisis de los datos y de la validez de las suposiciones utilizadas. (Siegel, 2013)

Para realizar un análisis predictivo, es fundamental contar con una gran cantidad de datos, tanto históricos como recientes, que permitan identificar patrones de comportamiento y extraer conocimiento. Este procedimiento se lleva a cabo mediante el aprendizaje computacional, donde los sistemas informáticos tienen la capacidad de aprender de forma autónoma, desarrollando nuevas habilidades y generando información a partir de los datos.

2.2. Series de tiempo

Una serie de tiempo es una secuencia de datos ordenados cronológicamente según el momento en que se registran. (Wei, 2005) Este tipo de información es fundamental en numerosos campos del conocimiento. En meteorología, por ejemplo, se registran variables como la velocidad del viento por hora, las temperaturas máximas y mínimas diarias, y las precipitaciones anuales. En geofísica, se monitorean continuamente los movimientos sísmicos con el fin de anticipar posibles terremotos. En las ciencias sociales, se analizan series anuales sobre tasas de natalidad y mortalidad, accidentes domésticos y distintos tipos de criminalidad. Estas secuencias temporales están presentes en prácticamente todas las áreas, desde el análisis financiero y económico hasta el monitoreo de la actividad en redes sociales. Aunque muchas veces pasen desapercibidas, contienen información valiosa para identificar patrones, detectar anomalías y comprender el comportamiento dinámico de los fenómenos a lo largo del tiempo.

A diferencia de otros tipos de datos, las series temporales presentan una estructura dependiente del tiempo, lo que significa que sus valores no son generados de manera independiente o aleatoria. Además,

su variabilidad puede cambiar a lo largo del tiempo, suelen seguir una tendencia y pueden contener patrones cíclicos. (Wei, 2005)

2.3. Componentes de las series de tiempo

Para analizar y modelar una serie de tiempo de manera efectiva, es fundamental comprender sus componentes principales. Una serie temporal no es simplemente una secuencia de valores registrados en distintos momentos, sino que puede estar influenciada por diversos factores que afectan su comportamiento a lo largo del tiempo. La descomposición de una serie de tiempo permite identificar patrones subyacentes y distinguir entre tendencias de largo plazo, fluctuaciones periódicas y variaciones aleatorias. (Peixeiro, 2022) A continuación, se describen las principales componentes que conforman una serie temporal y su impacto en el análisis predictivo.

2.3.1. Tendencia

La tendencia en una serie de tiempo representa los cambios de largo plazo que ocurren de manera progresiva en los datos, ya sea en forma de incrementos o decrementos sostenidos a lo largo del tiempo. Este componente refleja el comportamiento general de la serie y puede estar influenciado por factores estructurales, económicos, sociales o tecnológicos. (Peixeiro, 2022)

Por ejemplo, en el análisis del consumo de agua en una ciudad, una tendencia creciente podría indicar un aumento sostenido en la demanda debido al crecimiento poblacional, mientras que una tendencia decreciente podría reflejar la implementación de políticas de ahorro de agua o mejoras en la eficiencia del sistema de distribución.

2.3.2. Estacionalidad

El componente estacional de una serie de tiempo representa los patrones que se repiten a intervalos regulares dentro de un período determinado. Estos ciclos estacionales están asociados a factores recurrentes que influyen en los datos de manera predecible, como las estaciones del año, los días de la semana o los horarios del día. (Peixeiro, 2022)

La estacionalidad refleja cómo los valores de la serie se desvían de la tendencia general. En ciertos momentos, esta desviación es positiva, lo que genera picos en el gráfico de observaciones, mientras que, en otros casos, la desviación es negativa, produciendo caídas en los valores registrados. Estas fluctuaciones estacionales suelen repetirse de manera predecible en intervalos regulares, lo que permite identificar patrones y anticipar comportamientos futuros en la serie de tiempo.

2.3.3. Ciclicidad

Los ciclos en las series temporales representan fluctuaciones recurrentes en los datos que no siguen un patrón fijo como la estacionalidad, pero que ocurren en intervalos de tiempo prolongados. A diferencia de las variaciones estacionales, los ciclos no tienen una duración constante, ya que dependen de factores externos como condiciones económicas, políticas o ambientales. (Cryer & Chan, 2008)

Estos patrones cíclicos pueden observarse en una amplia variedad de contextos, como el crecimiento económico, las oscilaciones en los mercados financieros o la demanda de ciertos productos a lo largo de los años. Un ciclo típico se compone de fases de expansión, donde los valores aumentan progresivamente, y fases de contracción, donde los valores disminuyen antes de iniciar un nuevo crecimiento.

2.3.4. Estacionariedad

La estacionariedad se refiere a las series de tiempo cuyas propiedades estadísticas no cambian con el tiempo. En otras palabras, tiene una media, una varianza y una autocorrelación constantes, y estas propiedades son independientes del tiempo. Una manera rápida de verificar si una serie de tiempo es estacionaria o no, es con su tendencia, si existe una tendencia ya sea creciente o decreciente, no es estacionaria, caso contrario, si la tendencia es una constante, la serie de tiempo si es estacionaria. (Peixeiro, 2022)

Muchos modelos de predicción asumen que los datos son estacionarios. Para aplicar estos modelos, es fundamental verificar que la serie cumple con esta condición; de lo contrario, las estimaciones obtenidas no serán confiables. Por lo tanto, para el análisis de una serie de tiempo, primero debemos determinar si es una serie estacionaria o no. Si no es estacionaria, será necesario aplicar transformaciones para convertirla estacionaria. Sin embargo, en la práctica, es poco común encontrar series temporales completamente estacionarias en su forma original, ya que muchos fenómenos presentan tendencias o patrones estacionales que deben ser tratados antes del análisis.

Para verificar si una serie de tiempo es estacionaria existe la prueba de Dickey-Fuller. En palabras sencillas, esta prueba realiza una regresión en la que se analiza la relación entre el valor actual de la serie y su valor en el periodo anterior. El objetivo es verificar si existe una dependencia significativa que tienda a corregir los desvíos, es decir, que fuerce a la serie a regresar hacia un nivel constante a lo largo del tiempo. Si se detecta esta tendencia de corrección, se concluye que la serie es estacionaria. En cambio, si el valor actual depende fuertemente de su valor anterior sin mostrar tendencia a estabilizarse, se considera que la serie presenta una raíz unitaria, indicio de no estacionariedad. (Kleiber & Zeileis, 2008)

Formalmente, la prueba contrasta dos hipótesis:

- **Hipótesis nula (H_0):** la serie presenta una raíz unitaria, es decir, no es estacionaria.
- **Hipótesis alternativa (H_1):** la serie no presenta una raíz unitaria, es decir, es estacionaria.

La decisión se toma a partir del valor p (p-value) obtenido en la prueba, si el valor p es menor que un nivel de significancia previamente definido, que normalmente es 0.05, se rechaza la hipótesis nula, concluyendo que la serie es estacionaria.

2.3.5. Ruido blanco

El ruido blanco se define como una secuencia de variables aleatorias independientes e idénticamente distribuidas, con media cero y varianza constante (Udo Moffat & Alphonus Akpan, 2019). En otras

palabras, se trata de un conjunto de valores que no presentan ninguna estructura predecible ni patrón discernible en el tiempo.

Este componente es clave en el modelado de datos temporales, ya que representa la parte completamente aleatoria de una serie de tiempo, es decir, aquellas fluctuaciones que no pueden explicarse por tendencias, estacionalidades o ciclos. En muchos casos, los modelos de predicción buscan transformar una serie de tiempo en una serie de ruido blanco, eliminando componentes estructurales, para luego analizar mejor la información relevante en los datos.

2.3.6. Autocorrelación

La correlación es una medida estadística que evalúa la relación entre dos variables, determinando qué tan estrechamente varían juntas. Cuando dos variables están altamente correlacionadas, significa que los cambios en una de ellas están asociados con cambios en la otra. La correlación puede ser positiva, cuando ambas variables aumentan o disminuyen simultáneamente, o negativa, cuando una variable aumenta mientras la otra disminuye. (Maki, 2010)

La autocorrelación, por otro lado, es un concepto similar, pero aplicado a una misma serie temporal en diferentes momentos. En lugar de analizar la relación entre dos variables distintas, la autocorrelación mide el grado de dependencia entre los valores actuales de una serie de tiempo y sus valores pasados. En términos simples, nos indica en qué medida los valores anteriores influyen en los valores futuros de la serie. (Maki, 2010)

2.4. Aprendizaje computacional

El análisis predictivo puede realizarse únicamente con herramientas de probabilidad y estadística, pero para obtener resultados altamente precisos es fundamental contar con grandes volúmenes de datos, que en algunos casos pueden alcanzar millones de registros. Este nivel de procesamiento requiere un alto grado de precisión y eficiencia, algo que resulta difícil para los seres humanos debido a limitaciones en la capacidad de interpretar grandes cantidades de información y la influencia de sesgos cognitivos. En contraste, las computadoras, gracias a los avances tecnológicos de las últimas décadas, han superado estas limitaciones. Su capacidad para procesar grandes volúmenes de datos de manera rápida y eficiente, analizar múltiples variables simultáneamente y construir modelos matemáticos precisos las convierte en la herramienta ideal para llevar a cabo este tipo de análisis. (Alpaydin, 2020)

En este contexto, el aprendizaje (*learning*) se refiere a la capacidad de un modelo matemático-estadístico para ajustar y optimizar sus parámetros de tal manera que sus valores de salida coincidan lo más próximo posible a los datos con los que fue entrenado. Este proceso es potenciado por el uso de computadoras, que permiten una optimización eficiente y a gran escala, dando origen al término *Machine Learning* (ML). (Alpaydin, 2020)

A lo largo de su evolución, la ciencia de la computación ha tenido como principal objetivo asistir al ser humano en la realización de diversas tareas, replicando sus procesos a través de algoritmos ejecutados por computadoras. Esto ha llevado al desarrollo de algoritmos especializados en tareas específicas, pero

con el tiempo se ha evidenciado la dificultad de diseñar algoritmos que puedan emular ciertas habilidades humanas con la misma naturalidad y eficiencia. Acciones como reconocer rostros de manera instantánea, reaccionar rápidamente ante obstáculos, conducir un vehículo o mantener una conversación en un idioma extranjero representan desafíos complejos para la programación convencional. Por lo cual, existen numerosos problemas para los cuales no disponemos de algoritmos determinísticos eficaces, pero contamos con grandes volúmenes de datos que pueden proporcionar información valiosa sobre estos fenómenos. Aquí es donde entra en juego el *Machine Learning*, en la exploración de patrones dentro de los datos para comprender procesos. (Liebowitz, 2020)

Dentro de este campo, la ciencia de la computación enfrenta dos desafíos principales. Primero, durante la etapa de entrenamiento del modelo, se requieren algoritmos eficientes para abordar problemas de optimización, además de la capacidad de almacenar y procesar grandes cantidades de datos. Segundo, una vez que el modelo ha sido entrenado, su fase de inferencia también debe ser eficiente para garantizar su aplicabilidad en escenarios del mundo real. (Alpaydin, 2020)

2.5. Modelos predictivos

Un modelo predictivo es una herramienta matemática o computacional que utiliza datos históricos y variables explicativas con el objetivo de estimar el valor futuro de una variable de interés. Se basa en principios de la estadística inferencial y del aprendizaje automático, permitiendo identificar patrones en los datos y generar predicciones precisas. Cuando se aplica en el ámbito empresarial, se convierte en un recurso valioso para mejorar la planificación, optimizar recursos y respaldar una toma de decisiones más informada y estratégica. (Contreras Morales, Ferreira Correa, & Mauricio, 2017)

2.5.1. Prophet

Prophet es un modelo predictivo desarrollado por Facebook para realizar pronósticos sobre datos de series de tiempo. Su enfoque se basa en un modelo aditivo, lo que significa que descompone la serie de tiempo en varios componentes que se suman entre sí para formar la predicción final. Estos componentes incluyen una tendencia no lineal, estacionalidades (como patrones anuales, semanales o diarios), y efectos especiales como los días festivos. Prophet está diseñado para funcionar especialmente bien con series de tiempo que presentan patrones estacionales definidos y que cuentan con varios ciclos de datos históricos. Además, es una herramienta robusta, puede manejar sin problemas datos faltantes, cambios repentinos en la tendencia de los datos (como un cambio de comportamiento en los usuarios), y valores atípicos sin que estos afecten de manera significativa la calidad del modelo. (Prophet official page, s.f.)

Una de las principales ventajas del modelo Prophet frente a otros enfoques más complejos, es su alto nivel de facilidad de uso. Prophet automatiza gran parte del proceso de modelado, lo que permite a los usuarios generar pronósticos sin necesidad de conocimientos teóricos profundos en series de tiempo o aprendizaje automático. Sin embargo, esta simplicidad también conlleva ciertas limitaciones. Al tratarse de un modelo más cerrado, ofrece menos posibilidades para comprender en detalle cómo se realiza el aprendizaje o para ajustar parámetros avanzados, lo cual puede ser una desventaja si no se alcanzan los niveles de precisión deseados. Según sus desarrolladores, el objetivo de Prophet es ofrecer un modelo de

regresión modular y sencillo que funcione bien con parámetros por defecto, pero que al mismo tiempo permita seleccionar componentes específicos y realizar ajustes cuando sea necesario. (Korstanje, 2021)

Como ya se mencionó, es un modelo aditivo que está compuesto por tres componentes: tendencia, estacionalidad y los *holidays*. El término *holidays* se refiere a fechas específicas que pueden generar un comportamiento inusual o repetitivo en la serie de tiempo. Aunque el nombre sugiere días festivos, no se limita únicamente a feriados oficiales, también puede incluir eventos recurrentes o particulares como cierres de mes, fechas de corte, campañas promocionales o cualquier otro acontecimiento que influya significativamente en la variable que se desea predecir. Estos tres elementos del modelo Prophet están combinados en la siguiente ecuación 2.1.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \quad (2.1.)$$

En esta formulación, $g(t)$ representa la función de tendencia, encargada de modelar los cambios no periódicos en la serie temporal. Por su parte, $s(t)$ captura los patrones estacionales periódicos, como las variaciones semanales o anuales. El término $h(t)$ incorpora los efectos de los *holidays*. Finalmente, el término de error ϵ_t refleja variaciones no explicadas por los componentes anteriores; en el modelo, se asume paramétricamente que este error se distribuye normalmente. (Taylor & Letham, 2017)

2.5.2. LSTM

Las redes LSTM (*Long Short-Term Memory*) constituyen una evolución dentro del campo de las redes neuronales artificiales. Se trata de una variante avanzada de las redes neuronales recurrentes, diseñada específicamente para abordar los desafíos del procesamiento de datos secuenciales. Para comprender su relevancia y funcionamiento, es útil considerar brevemente el contexto del que emergen.

Las redes neuronales artificiales o ANN (Artificial Neural Network) son modelos computacionales que intentan imitar, de forma simplificada, el proceso de toma de decisiones de las redes de neuronas del sistema nervioso central de seres humanos. Esta simulación se realiza a nivel básico, replicando el comportamiento de las neuronas y su interconexión, y su diseño se basa en conocimientos neurofisiológicos sobre el funcionamiento de las neuronas biológicas y la forma en que estas se organizan en redes. Aunque no buscan una representación exacta, las redes neuronales artificiales ofrecen una aproximación funcional que permite resolver tareas complejas de procesamiento y aprendizaje a partir de datos. (Graupe, 2013)

Como se muestra en la figura 2-1, las ANN están formadas por un conjunto de unidades de procesamiento denominadas "nodos". Estos transmiten datos entre sí, igual que en el cerebro las neuronas se transmiten impulsos eléctricos. Estos nodos están repartidos en tres tipos de capas: capa de entrada, capas ocultas y capa de salida. Independientemente de la capa de la que forme parte, cada nodo realiza algún tipo de tarea o función de procesamiento sobre cualquier entrada que reciba del nodo anterior. Esencialmente, cada nodo contiene una fórmula matemática. Si el resultado de aplicar esa fórmula matemática a la entrada supera un determinado umbral, normalmente llamada función de activación, el nodo pasa los datos a la siguiente capa de la red neuronal. Si la salida está por debajo del umbral, no se pasa ningún dato a la capa siguiente. (Sivanandam, 2009)

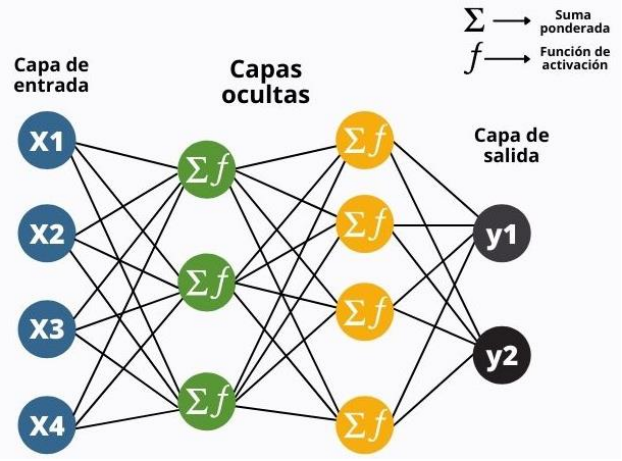


Figura 2-1: Elementos de una ANN
Fuente: https://abdatum.com, 2021

La ventaja que tiene las ANN se encuentra en la capacidad de volverse más sofisticada. Al aumentar la cantidad de capas ocultas, aumenta la capacidad de aprender representaciones más complejas y abstracta de los datos. La característica que describe la cantidad de capas ocultas que tiene una ANN se le llama "profundidad", y se considera que una ANN es profunda cuando tiene más de tres capas, incluyendo las de entrada y salida. De ahí el termino *Aprendizaje Profundo* (Deep Learning).

A diferencia de los modelos tradicionales de aprendizaje automático, cuyo rendimiento depende en gran medida de una cuidadosa ingeniería de características, el aprendizaje profundo permite que las redes neuronales descubran automáticamente representaciones relevantes directamente a partir de los datos sin procesar. La ingeniería de características consiste en transformar los datos de entrada de forma que faciliten la tarea del algoritmo posterior. Aunque esta técnica sigue teniendo utilidad en el aprendizaje profundo, una de las principales fortalezas de las redes neuronales es su capacidad para identificar y extraer representaciones útiles a partir de ejemplos, sin necesidad de intervención manual. Esta autonomía es precisamente lo que otorga al aprendizaje profundo su notable eficacia. A menudo, estas características creadas automáticamente son mejores que las creadas a mano. (Stevens, Antiga, & Viehmann, 2020)

Dentro del campo del aprendizaje profundo, han surgido diversas arquitecturas de redes neuronales diseñadas para abordar distintos tipos de problemas y estructuras de datos. Entre las más conocidas se encuentran las redes neuronales recurrentes (RNN), especialmente diseñadas para trabajar con datos secuenciales, como generación de texto, reconocimiento de voz o predicción de series temporales. El funcionamiento de las RNN se basa en procesar cada elemento de la serie de tiempo (x_t) de forma individual, manteniendo al mismo tiempo una "memoria" de lo aprendido anteriormente. Esto se logra a través del estado oculto (h_t), una representación interna generada por las capas ocultas al procesar cada dato. Como se puede ver en la figura 2-2, este estado se retroalimenta y se transfiere al procesamiento del siguiente elemento de la serie de tiempo, permitiendo que la red utilice la información de entradas anteriores para interpretar mejor las siguientes. Gracias a este mecanismo, las RNN son especialmente útiles en tareas donde el orden y el contexto de los datos son fundamentales. (Peixeiro, 2022)

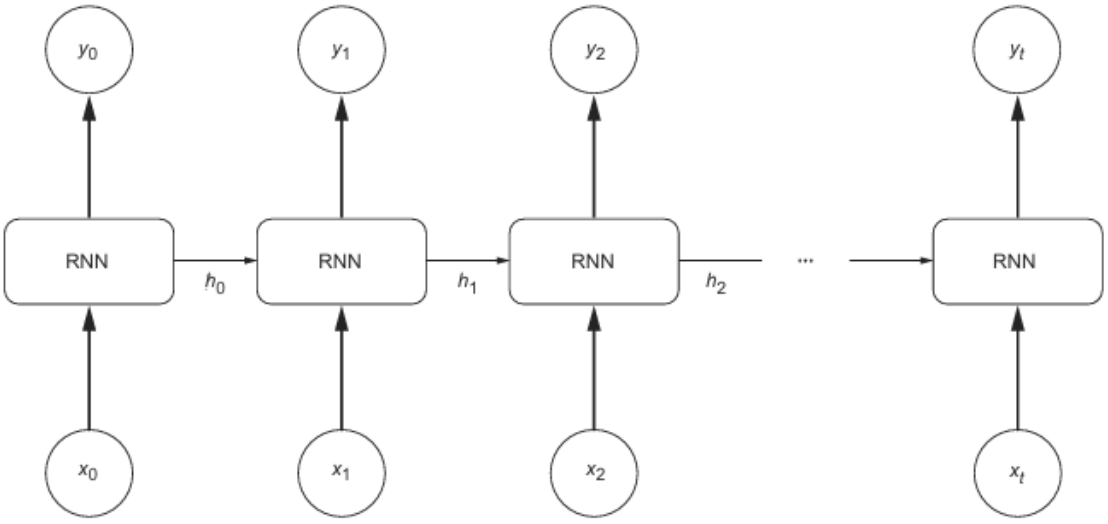


Figura 2-2: Comportamiento de los estados ocultos de las RNN
Fuente: Peixeiro, 2022

Sin embargo, las RNN presentan una limitación importante, aunque pueden recordar información a través de los estados ocultos, su memoria es de corto plazo (*short-term memory*). Esto se debe a un fenómeno conocido como desvanecimiento del gradiente. Durante el entrenamiento, el gradiente indica a la red cómo ajustar los valores para aprender de los datos de entrada. No obstante, a medida que la red procesa más elementos de la secuencia, este gradiente pierde fuerza, lo que hace que la información proveniente de los primeros pasos se vuelva cada vez menos influyente. Afortunadamente, las redes LSTM fueron específicamente diseñadas para superar esta dificultad.

Las LSTM incorporan un módulo especial llamado "celda de memoria", que es capaz de almacenar y conservar información durante períodos prolongados. Este módulo se regula mediante tres compuertas: la compuerta de entrada, la de olvido y la de salida, las cuales controlan qué información se añade, se descarta o se utiliza, respectivamente. Gracias a este mecanismo, las LSTM pueden aprender dependencias a largo plazo y manejar de forma más efectiva secuencias complejas, lo que mejora las predicciones de secuencias sin importar la cantidad de elementos que tengan. (Brownlee, 2019)

La figura 2-3 muestra el funcionamiento general de una LSTM, es muy similar a las RNN con la diferencia que ahora se incorpora un nuevo tipo de retroalimentación adicional. En este esquema, cada celda LSTM recibe como entrada tanto el dato actual de la secuencia (x_t) como la información proveniente del paso anterior, es decir, el estado oculto (h_{t-1}) y el estado de memoria que se va a almacenar en la celda de memoria (c_t). A partir de esta información, cada capa oculta procesa internamente los datos y produce una salida ($\hat{y}_t$) junto con una actualización de su estado de memoria, que es transmitida a la siguiente celda en la secuencia.

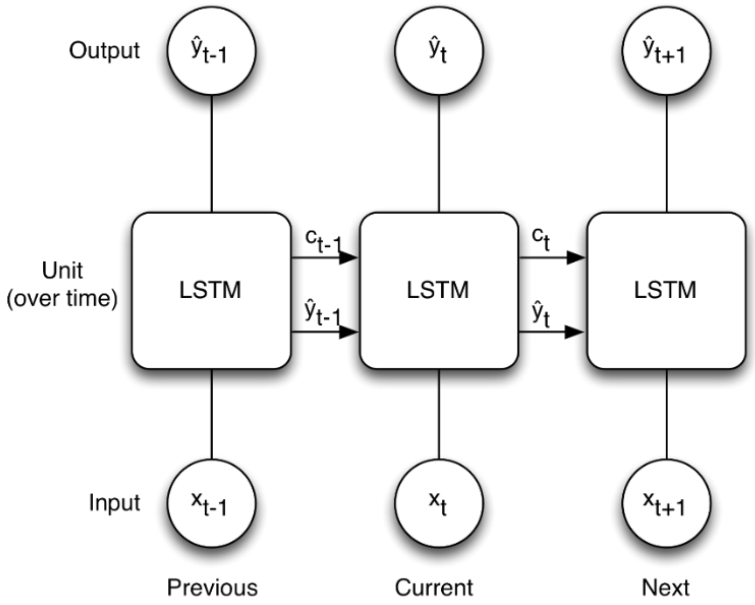


Figura 2-3: Diagrama del funcionamiento de la arquitectura LSTM
Fuente: https://medium.com, 2023

Este mecanismo permite que la red no solo procese la entrada actual, sino que también conserve y utilice información relevante de entradas pasadas. De esta manera, las LSTM logran capturar dependencias a largo plazo dentro de las secuencias, superando las limitaciones de las redes recurrentes tradicionales. Cada elemento de la serie de tiempo representa una etapa en la que la red analiza un nuevo elemento de la serie de datos, acumulando contexto a medida que avanza. (Brownlee, 2019)

2.6. Métricas de evaluación

Las métricas de evaluación son herramientas fundamentales en el desarrollo de modelos de aprendizaje automático, ya que permiten cuantificar el desempeño de un modelo al comparar sus predicciones con los valores reales. Su función principal es proporcionar una medida objetiva del grado de precisión, error o ajuste de un modelo, lo cual resulta esencial para tomar decisiones informadas sobre su efectividad y su viabilidad para ser aplicado en un contexto real. En el caso particular de modelos predictivos basados en series de tiempo, estas métricas ayudan a identificar qué configuración del modelo es más adecuada para capturar los patrones temporales presentes en los datos, y permiten realizar comparaciones entre diferentes enfoques o estudios similares.

2.6.1. RMSE (Root Mean Squared Error)

La RMSE o raíz del error cuadrático medio, es una métrica ampliamente utilizada para evaluar la precisión de modelos de regresión, especialmente en series de tiempo. Se calcula como la raíz cuadrada del promedio de los errores al cuadrado entre los valores reales y los valores predichos. (Salomon, 2007) Matemáticamente, se expresa como se muestra en la ecuación 2.2.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (2.2.)$$

Donde y_i representa los valores reales, $\hat{y}_i$ los valores predichos por el modelo, y n el número total de observaciones.

El RMSE penaliza de forma más severa los errores grandes debido a la elevación al cuadrado, por lo que es sensible a valores atípicos. Cuanto menor sea el valor del RMSE, mejor será el ajuste del modelo a los datos reales. Su unidad de medida es la misma que la variable que se está prediciendo, lo que facilita su interpretación práctica

2.6.2. MAE (Mean Absolute Error)

El MAE o error absoluto medio, es una métrica de evaluación que mide la precisión de un modelo de regresión calculando el promedio de las diferencias absolutas entre los valores predichos y los valores reales. A diferencia de la RMSE, el MAE no eleva al cuadrado los errores, por lo que todos los errores tienen el mismo peso, lo que lo hace menos sensible a valores atípicos (Peixeiro, 2022). Su fórmula se muestra en la ecuación 2.3.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (2.3.)$$

Donde y_i representa los valores reales, $\hat{y}_i$ los valores predichos por el modelo, y n el número total de observaciones.

Esta métrica es especialmente útil porque es fácil de interpretar, representa directamente el promedio de la diferencia absoluta entre los valores predichos y los reales. Por ejemplo, un MAE de 150 indica que, en promedio, el modelo se equivoca por 150 unidades monetarias al estimar el ingreso diario. Esto lo convierte en una opción clara y comprensible para comunicar la precisión de las predicciones a audiencias no técnicas.

2.6.3. MAPE (Mean of the Absolute Percentage Errors)

El MAPE, o error porcentual absoluto medio, es una métrica ampliamente utilizada para evaluar la precisión de modelos de predicción, especialmente en el contexto de series de tiempo. Se calcula como el promedio del valor absoluto del error porcentual entre los valores reales y los valores predichos por el modelo. (Peixeiro, 2022) Su fórmula se muestra en la ecuación 2.4.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (2.4.)$$

Donde y_i representa los valores reales, $\hat{y}_i$ los valores predichos por el modelo, y n el número total de observaciones.

Una de las principales ventajas del MAPE es que es fácil de interpretar y es independiente de la escala de los datos. Esto significa que sin importar si se están trabajando con valores pequeños o grandes (por ejemplo, desde montos en centavos hasta ingresos en millones), el MAPE siempre se expresará como un porcentaje. De esta manera, el MAPE indica qué tanto, en promedio, se desvía el modelo con respecto a los valores reales, sin importar si las predicciones fueron mayores o menores (Peixeiro, 2022). Por ejemplo, un MAPE de 12% implica que, en promedio, las predicciones del modelo se desvían un 12% de los valores reales.

2.7. CRISP-DM

CRISP-DM es el acrónimo de *Cross Industry Standard Process for Data Mining,* el cual es una metodología estándar ampliamente utilizada para el desarrollo de proyectos de minería de datos y ciencia de datos, independientemente del área de aplicación. Su objetivo es proporcionar una guía estructurada y comprensible para transformar problemas de negocio en soluciones basadas en datos, a través de una serie de fases bien definidas. Este marco de trabajo tiene como objetico llevar a cabo grandes proyectos de minería de datos menos costosos, más fiables, repetibles, manejables y rápidos. (IBM, s.f.)

La Metodología está compuesta por seis fases, el cual dependen entre sí, tanto en secuencia o de forma cíclica, pudiendo tener interacciones que permitan mejorar la aproximación obtenida en otras fases anteriores:

- **Comprensión del negocio (Business Understanding):** En esta primera fase, el objetivo es entender el problema desde la perspectiva del negocio u organización. Se identifican los objetivos principales, las preguntas que se quieren responder y los posibles beneficios. También se define el enfoque analítico más adecuado para abordar el problema. En esta etapa, es fundamental mantener una comunicación constante con los actores involucrados para alinear los objetivos técnicos con los objetivos del negocio.
- **Comprensión de los datos (Data Understanding):** Aquí se recopilan los datos disponibles, se evalúa su calidad y se exploran sus principales características. Es una fase exploratoria que permite detectar errores, valores atípicos, estructuras inesperadas o relaciones importantes entre variables. Esta etapa también ayuda a determinar si los datos disponibles son suficientes y adecuados para cumplir los objetivos del proyecto o si es necesario obtener información adicional.
- **Preparación de los datos (Data Preparation):** En esta etapa se realiza el tratamiento y la limpieza de los datos. Implica tareas como eliminar valores faltantes o duplicados, transformar variables, seleccionar atributos relevantes y crear nuevas variables que puedan ser útiles para el análisis. Esta fase suele ser una de las más extensas, ya que la calidad de los datos tiene un impacto directo en el desempeño del modelo. El resultado es un conjunto de datos listo para ser utilizado en la etapa de modelado.
- **Modelado (Modeling):** Una vez que los datos están preparados, se seleccionan y aplican técnicas de modelado adecuadas al problema (por ejemplo, regresión, árboles de decisión, redes

neuronales, etc.). En esta etapa también se afinan los parámetros de los modelos y se evalúan mediante técnicas de validación. A menudo es necesario regresar a la fase de preparación si se detecta que los datos requieren nuevos ajustes.

- **Evaluación (Evaluation):** Después de construir uno o varios modelos, se evalúa su desempeño en función de métricas específicas y su capacidad para responder a las preguntas del negocio. No se trata solo de elegir el modelo con mayor precisión, sino de verificar si cumple con los objetivos definidos al inicio del proyecto. También se consideran los riesgos potenciales del modelo, como el sobreajuste o la interpretabilidad.
- **Despliegue (Deployment):** Finalmente, se implementa el modelo para que genere valor real. Esto puede implicar integrarlo en un sistema de información, generar informes periódicos, o automatizar decisiones. También se establecen mecanismos para monitorear el rendimiento del modelo a lo largo del tiempo, ya que los datos pueden cambiar y es posible que se necesite actualizar el modelo.

Cada una de estas fases interactúa entre sí, como se muestra en la figura 2-4. En algunos casos, si es necesario, se puede regresar a una fase anterior con el fin de mejorar el desarrollo del proceso.

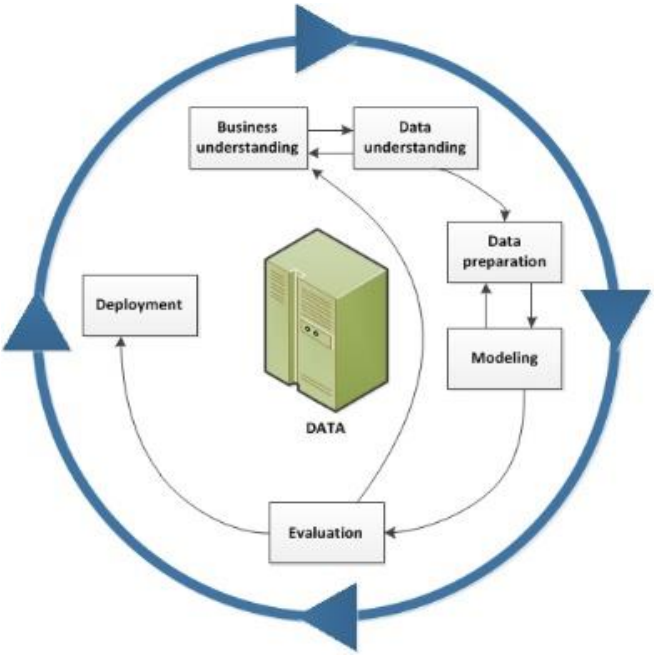


Figura 2-4: Ciclo de vida de minería de datos
Fuente: https://www.ibm.com, 2021

3. Marco metodológico

El desarrollo de modelos predictivos basados en series de tiempo requiere un enfoque estructurado que abarque desde la recolección y procesamiento de datos, hasta la evaluación del desempeño del modelo. En este estudio, se explorarán distintos métodos y herramientas para analizar datos históricos, identificar patrones y construir un modelo capaz de realizar predicciones con precisión. A continuación, se detalla la metodología utilizada, incluyendo las fases de preparación de datos, diseño del modelo y configuración de los parámetros para el entrenamiento.

3.1. Área de estudio

Como ya se sabe, este proyecto se basa en la predicción de ingresos por la cobranza de facturas en las EPSA, por lo tanto, se procuró obtener los datos auténticos de alguna de estas entidades ubicadas en Cochabamba. Los datos que se utilizan en este trabajo pertenecen a la EPSA *Thika Khatu*, la cual se encuentra en el distrito 6 del municipio de Tiquipaya, de la provincia de Quillacollo, Cochabamba. Esta EPSA, provee de agua a la OTB que lleva su mismo nombre, la cual tiene aproximadamente 850 habitantes (Martinez Soto, 2025). En la figura 3-1 se puede observar su localización y distribución interna.

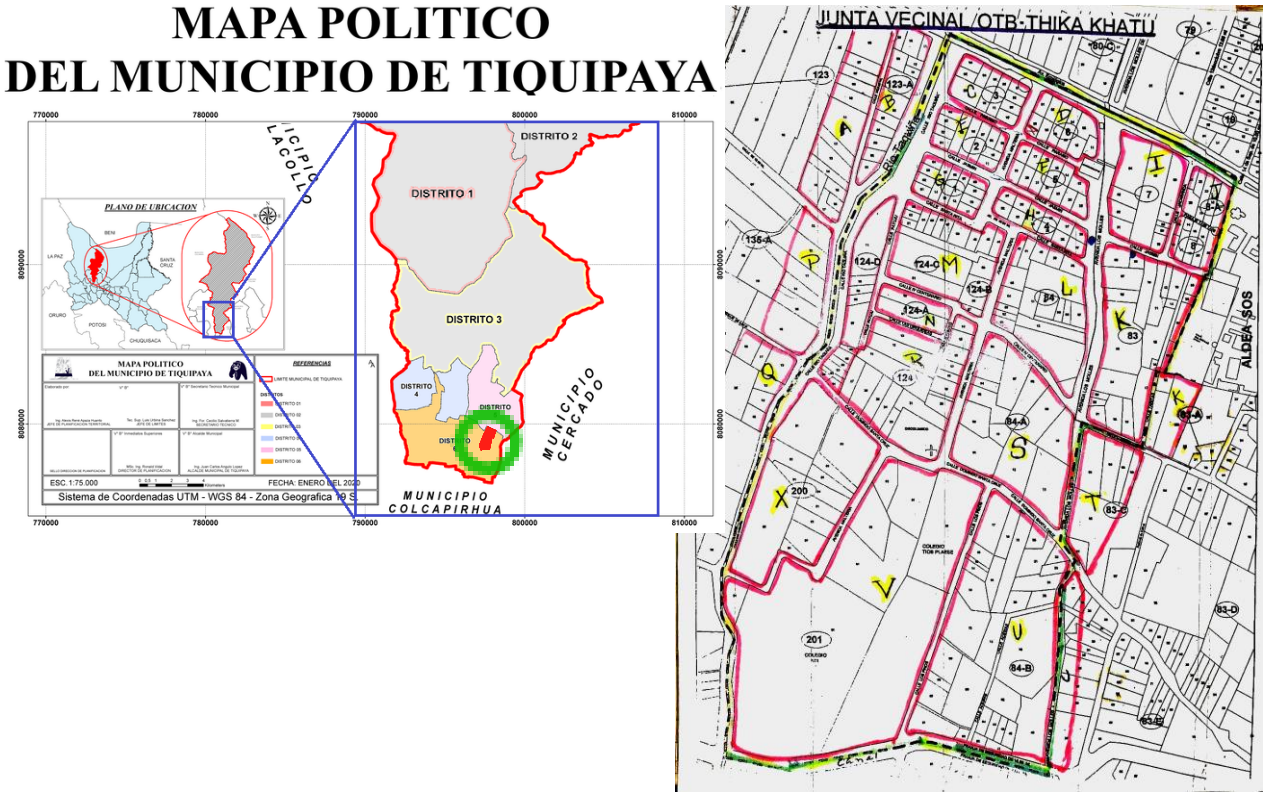


Figura 3-1: Ubicación OTB Thika Khatu
Fuente: Gobierno Autónomo de Tiquipaya y OTB Thika Khatu, 2024

3.2. Flujograma metodológico

En la figura 3-2, se presenta un flujograma que resume las principales etapas que se desarrollan durante el proyecto. Este diagrama refleja de manera estructurada el proceso seguido, desde la recolección y preparación de los datos hasta la implementación, evaluación y comparación del modelo predictivo propuesto.

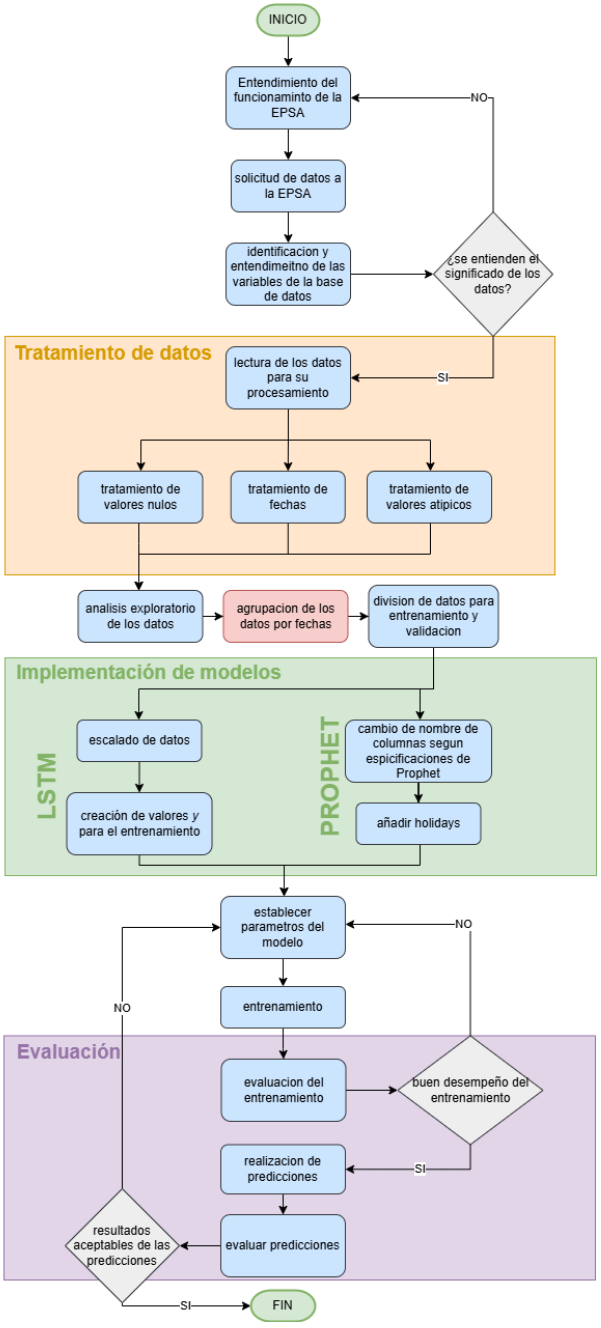


Figura 3-2: Flujograma metodológico

Fuente: Elaboración propia, 2025

Cada fase del flujograma se describe a continuación:

Entendimiento del funcionamiento de la EPSA: Se investiga el contexto operativo de la EPSA, su estructura administrativa y los procesos relacionados con la recaudación de ingresos, para comprender mejor el objetivo del proyecto.

Solicitud de datos a la EPSA: Se realiza el requerimiento formal de acceso a la base de datos institucional, la cual contenía información histórica sobre el consumo y cobro de facturas.

Identificación y entendimiento de las variables de la base de datos: Se analiza las variables disponibles en los archivos entregados, identificando aquellas relevantes para el problema de predicción y comprendiendo su significado.

Lectura de los datos para su procesamiento: Se cargan los datos en el entorno de desarrollo para su posterior manipulación, asegurando la correcta interpretación de formatos y estructuras.

Tratamiento de datos nulos: Se identifican los valores faltantes y se aplican estrategias de imputación adecuadas para evitar sesgos o errores durante el modelado.

Tratamiento de fechas: Se estandarizan los formatos de fecha y se extraen componentes temporales útiles, como día, mes o año, para mejorar la organización temporal de la información.

Tratamiento de valores atípicos: Se analizan y atenúan los valores extremos que pueden distorsionar los resultados del modelo.

Análisis exploratorio de datos: Se realizan visualizaciones y cálculos estadísticos para entender mejor el comportamiento de las variables y detectar patrones relevantes.

Agrupación de los datos por fechas: Se organizan los datos en función de su componente temporal, utilizando una frecuencia diaria para facilitar la modelación de la serie de tiempo.

División de datos para entrenamiento y validación: Se separan los datos en conjuntos de entrenamiento y validación, respetando el orden temporal para evitar fugas de información.

Escalado de datos: Se normalizan los valores numéricos mediante el método de mínimos y máximos, para asegurar un entrenamiento más estable del modelo.

Creación de valores *y* para el entrenamiento: Se generan secuencias de entrada *x* y sus respectivos valores objetivo *y* utilizando distintas ventanas de tiempo (time steps) como configuración del modelo LSTM.

Cambio de nombre de columnas según especificaciones de Prophet: Se cambia el nombre de las columnas de la serie de tiempo, ya que para entrenar el modelo Prophet es necesario que las columnas estén nombradas como *ds* y *y*.

Añadir holidays: Se establecen algunas fechas a final de cada año como *holidays* porque se identifican tendencias elevadas en estas épocas.

Establecer parámetros del modelo: Se define la arquitectura de la red LSTM, así como parámetros como el número de épocas, tamaño de batch, función de pérdida y optimizador.

Entrenamiento: Se entrena el modelo LSTM utilizando los datos preparados, permitiendo que aprenda patrones temporales para realizar predicciones.

Evaluación del entrenamiento: Se analizan las métricas obtenidas durante el entrenamiento, verificando si el modelo estaba aprendiendo adecuadamente sin sobreajustarse.

Realización de predicciones: Se generan predicciones sobre el conjunto de validación, utilizando el modelo entrenado y siguiendo la misma estructura de entrada.

Evaluar predicciones: Se comparan las predicciones con los valores reales utilizando las métricas RMSE, MAE y MAPE, para medir el rendimiento y la precisión del modelo.

3.3. Fuentes de información

El desarrollo de modelos predictivos basados en series de tiempo requiere contar con datos históricos confiables y bien estructurados. En este contexto, las fuentes de información juegan un papel crucial, ya que constituyen la base sobre la cual se entrenan, validan y evalúan los modelos. Esta sección describe los orígenes de toda la información utilizada en el proyecto. Cabe recalcar que toda la información recabada es secundaria, es decir, los datos son proporcionados por la EPSA Thika Khatu, y no son recolectados ni medidos de forma directa por el autor.

3.3.1. Entrevista

Como primer paso en la recolección de información, se coordina una entrevista con el presidente de la OTB Thika Khatu. Esta instancia permite esclarecer aspectos clave del contexto operativo y administrativo de la OTB, facilitando una mejor interpretación de los datos disponibles. Durante la entrevista se recopilan datos demográficos relevantes de la zona, incluyendo información sobre la población atendida y mapas de ubicación geográfica de la comunidad. Además, se obtiene una descripción detallada del sistema de suministro de agua, desde su captación hasta su distribución final, lo que permite identificar los puntos críticos del proceso y las variables más relevantes a considerar en el análisis posterior.

Además de la entrevista, se solicita acceso de facturas emitidas por la OTB a sus usuarios. Estas facturas no solo brindan una visión más clara sobre el formato, la estructura y los conceptos que se manejan en el sistema de facturación, sino que también resultan de gran utilidad en etapas posteriores del proyecto, permitiendo interpretar correctamente ciertos campos de la base de datos original, cuya descripción no está documentada.

Entre los datos más relevantes que se muestran en la factura de la figura 3-3, se encuentra la fecha de pago, cliente, categoría del domicilio, consumo de agua, un detalle de los conceptos que se cobra y el total a pagar. Los demás datos serán explicados a mayor detalle más adelante cuando se exponga la información de la base de datos.

ENTIDAD PRESTADORA DE SERVICIOS DE AGUA POTABLE
EPSA - THIKA KHATU
TELF: 4313779 No: 3668
COBRO DE SERVICIOS

LUGAR Y FECHA DE PAGO	DIRECCION DEL CLIENTE
Linde: 26/10/2024	"Thika Khatu"/
NOMBRE DEL CLIENTE	NIT/C/CLIENTE
	0

CICLO	FECHA DE EMISION	FECHA DE VENCIMIENTO	DIAS VENC.	CATEGORIA
	30/09/2024	30/10/2024	---	Domestica

CODIGO CLIENTE	CONCEPTO	IMPORTE EN Bs.
C 267	Consumo de Agua	39.00
	Reposicion Comprobante	1.30
	Recargo por falta de pago	0.00
	Falta a Reunion	0.00
	Trabajos/Desfile/Bloqueo	0.00
	Multa/Alteracion	0.00
	Reconexion	0.00
	CREDITO FISCAL	40.30
	TOTAL A CANCELAR	40.30

FECHA Y LECTURA ANTERIOR	FECHA Y LECTURA ACTUAL	PERIODO DE CONSUMO	PERIODO DE COBRANZA	CONSUMO m³
1082.00	1102.00		Septiembre 2024	20.00

SON: CUARENTA Y 30/100

NOTA:

SERVICIO DE AGUA POTABLE THIKA KHATU 26 OCT 2024 CANCELADO LINDE - OESTE

Pague su cuenta antes de la fecha de vencimiento, evitará molestias y recargos innecesarios por corte del servicio

Figura 3-3: Factura emitida por la EPSA Thika Khatu

Fuente: Elaboración propia, 2025

3.3.2. Base de datos

La EPSA dispone de un sistema informático de facturación que facilita tanto el proceso de cobranza como la gestión administrativa de los registros de facturas. Si bien no se puede acceder a información detallada sobre la estructura y el funcionamiento interno de este sistema, es posible el acceso a su base de datos, la cual contiene registros históricos que datan desde el año 2020 hasta inicios del 2025.

Originalmente, la base de datos cuenta con 19.890 registros. Cada uno de ellos corresponde a una factura emitida por la EPSA y está compuesto por 53 variables. Es importante señalar que, inicialmente, no se dispone de una documentación precisa sobre el significado de muchas de estas variables, pero al mismo tiempo, se identifica que algunas de ellas no almacenaban información útil, ya que no son utilizadas para su propósito original, presentando campos completamente vacíos o con muy pocos valores registrados. Por esta razón, se decide no enfocarse en indagar en el significado de algunas variables y colocar en su descripción "descartar", indicando que no son consideradas para los análisis posteriores. Todas las descripciones de las variables se encuentran en el anexo B.

3.4. Tratamiento de datos

El tratamiento de los datos inicia a partir de la información extraída de un respaldo de la base de datos principal de la EPSA, el cual se encuentra en formato xlsx. Este archivo contiene el historial de facturación y otros registros administrativos relevantes para el análisis. Antes de proceder con cualquier tipo de modelado o exploración, es necesario realizar una serie de pasos de transformación y limpieza para estructurar adecuadamente los datos, corregir inconsistencias y asegurar su calidad. Esta sección

detalla las tareas realizadas para preparar el conjunto de datos de forma que sea utilizable en las siguientes fases del proyecto.

Cabe recalcar, que todo el proceso, desde el tratamiento y limpieza de los datos, entrenamiento de modelos y validación de los resultados, se lleva a cabo utilizando el lenguaje de programación Python dentro del entorno interactivo de Jupyter Notebook. Para estas tareas, se hace uso principalmente de la biblioteca Pandas, ampliamente reconocida por sus potentes herramientas para la manipulación y análisis de datos estructurados. Esta biblioteca permite trabajar con estructuras tipo *DataFrame*, lo que facilita operaciones como transformación y limpieza.

3.4.1. Lectura de datos

En particular, Pandas cuenta con una función específica para la lectura de archivos en formato Excel, lo cual resultó ideal para el archivo recibido. La figura 3-4 ilustra cómo se realiza la carga inicial de los datos utilizando esta función.

```
os.getcwd()
```

```
'C:\\Users\\javie\\cursos\\diplomado\\modulo 6 - trabajo final\\anexo principal'
```

```
df = pd.read_excel("./lecturas.xlsx")
```

```
df.head()
```

	SOCIO	N_SOCIO	DIRECCION	NRO	FONO	RUC_CLI	FECHA	COD	CONCEPTO	IMPORTE	CATEGORIA	MES_COBRO	LEC_ANTES	LECT_HOY
0	260.0	260	Thika Khatu	NaN	NaN	NaN	NaN	NaN	NaN	18.0	Domestica	Junio	40.0	52.0
1	260.0	260	Thika Khatu	NaN	NaN	NaN	NaN	NaN	NaN	18.0	Domestica	Julio	52.0	52.0
2	260.0	260	Thika Khatu	NaN	NaN	NaN	NaN	NaN	NaN	18.0	Domestica	Agosto	52.0	52.0
3	260.0	260	Thika Khatu	NaN	NaN	NaN	NaN	NaN	NaN	18.0	Domestica	Septiembre	52.0	52.0
4	260.0	260	Thika Khatu	NaN	NaN	NaN	NaN	NaN	NaN	18.0	Domestica	Octubre	52.0	52.0

```
df.shape
```

```
(19889, 53)
```

Figura 3-4: Lectura del archivo de datos con pandas

Fuente: Elaboración propia, 2025

Antes de proceder con la lectura del archivo, se verifica que este se encuentre ubicado en la misma ruta del proyecto. Para ello, se utiliza la biblioteca **os**, que permite interactuar con el sistema operativo y gestionar rutas de archivos de forma eficiente. Esta verificación asegura que el archivo esté accesible desde el entorno de trabajo, evitando errores al momento de su carga.

3.4.2. Tratamiento de valores nulos

A simple vista es evidente la presencia de datos vacíos, sin embargo, es fundamental identificar con precisión cuántos valores faltantes hay y en qué columnas se encuentran. Para esta tarea, se emplea el método **isnull()** del objeto DataFrame de la biblioteca Pandas, que permite detectar valores nulos en el

conjunto de datos. Al combinarlo con la función **sum()**, se obtiene un resumen que indica la cantidad de datos vacíos por columna.

Como se puede observar en la figura 3-5, algunas columnas contienen una gran cantidad de valores nulos, por ejemplo, en columnas como NRO, COD o EMISION, más del 90 % de los datos están vacíos, por lo que es ideal eliminarlas. Además, se aprovecha este proceso para descartar otras columnas que no aportaban valor al análisis ni al entrenamiento del modelo, como aquellas que contienen información identificativa (por ejemplo, el código de socio, número de casa o número de teléfono). Este tipo de datos, al ser únicos para cada individuo, no ofrecen patrones útiles que un modelo pueda generalizar. También se toma en cuenta las columnas con datos textuales (como las direcciones) porque no pueden ser interpretados por los modelos predictivos ya que estos son modelos matemáticos, lo que significa que solo se basan en datos numéricos. Por estas razones, es adecuado excluir estas variables del conjunto de datos final.

```
df.isnull().sum()
```

SOCIO	16	ATRASO	104	RECONEX	19850
N_SOCIO	26	TOTAL	101	FALTAS	117
DIRECCION	298	LITE	1117	BANDI	753
NRO	19670	EMISION	19883	OTROS	147
FONO	12054	MEDIDOR	10319	BENEMERITO	135
RUC_CLI	18966	VENCE	128	ZONA	178
FECHA	1018	ALUMBRADO	146	DESCBENEME	142
COD	19864	BASURA	144	ALCANTARI	135
CONCEPTO	1010	CERTIF	19472	ORDENES	153
IMPORTE	102	TITULO	19872	HEXA	19700
CATEGORIA	100	INGRESO	132	NUMFAC	1122
MES_COBRO	103	BANDERA	136	FACTURA	1017
LEC_ANTES	108	VALOR	138	COPIA	113
LECT_HOY	110	CARNET	4637	REUNION	18086
CREDITO	145	FLEC	115	TRABAJOS	18844
CONSUMO	104	MINIMO	19885	PASIBLE	139
		CORTE	10688	LEIANTES	115
		MULTAS	19875	GESTION	116
				CPU	19889

Figura 3-5: Cantidad de valores nulos o vacíos por columnas

Fuente: Elaboración propia, 2025

Considerando todos los motivos mencionados, la depuración de columnas reduce el número total de variables de 53 a 20, lo que representa una disminución significativa en la cantidad de valores nulos presentes en el conjunto de datos.

A pesar de la reducción considerable en la cantidad de valores nulos tras la depuración de columnas, aún persisten algunos datos faltantes, como se puede observar en la figura 3-6. La mayoría de las columnas presenta una cantidad reducida de valores nulos, no más de 150 registros y, tras una revisión, se descubre que las filas afectadas suelen contener múltiples campos vacíos o inconsistentes. Esto sugiere posibles errores de registro o problemas en el proceso de respaldo de la base de datos. En el caso de la columna FECHA, que tiene una mayor cantidad de valores nulos, la ausencia de datos no representa un error o inconsistencia, sino que indica que la factura correspondiente aún no ha sido pagada, lo cual, no resulta útil en nuestro caso ya que las facturas que aún no fueron pagadas no representan un ingreso a la EPSA.

Es por esto que se decide eliminar las filas que contengan valores nulos por completo, utilizando la función **dropna()** del objeto DataFrame de Pandas, que facilita la eliminación de filas con datos faltantes, como se muestra en la figura 3-7.

```
df.isnull().sum()
```

```
FECHA         1018
IMPORTE        102
CATEGORIA      100
MES_COBRO      103
LEC_ANTES      108
LECT_HOY       110
CONSUMO        104
ATRASO         104
TOTAL          101
VENCE          128
INGRESO        132
VALOR          138
FLEC           115
FALTAS         117
BENEMERITO     135
ZONA           178
DESCBENEME     142
PASIBLE        139
LEIANTES       115
GESTION        116
```

Figura 3-6: Cantidad de valores nulos o vacíos por columnas después de la depuración

Fuente: Elaboración propia, 2025

```
df = df.dropna(subset=['FECHA', 'IMPORTE', 'CATEGORIA', 'MES_COBRO', 'LEC_ANTES',
       'LECT_HOY', 'CONSUMO', 'ATRASO', 'TOTAL', 'VENCE', 'INGRESO', 'VALOR',
       'FLEC', 'FALTAS', 'BENEMERITO', 'ZONA', 'DESCBENEME', 'PASIBLE',
       'LEIANTES', 'GESTION'])
df.isnull().sum()
```

```
FECHA         0
IMPORTE       0
CATEGORIA     0
MES_COBRO     0
LEC_ANTES     0
LECT_HOY      0
CONSUMO       0
ATRASO        0
TOTAL         0
VENCE         0
INGRESO       0
VALOR         0
FLEC          0
FALTAS        0
BENEMERITO    0
ZONA          0
DESCBENEME    0
PASIBLE       0
LEIANTES      0
GESTION       0
```

Figura 3-7: Cantidad de valores nulos o vacíos después de la depuración por filas

Fuente: Elaboración propia, 2025

3.4.3. Tratamiento de fechas

El tratamiento adecuado de las fechas es fundamental en el análisis de datos temporales, especialmente cuando se trabaja con series de tiempo. En este proyecto, es necesario agrupar los datos a partir de las fechas, por lo cual es necesario garantizar que los datos sean consistentes.

La base de datos cuenta con seis columnas que contienen información temporal, pero en este tratamiento solo se toma en cuenta a la columna FECHA, ya que será el criterio a partir el cual se genera la serie de tiempo, que se explicará posteriormente. Las demás columnas no se toman en cuenta en ningún ámbito. Inicialmente Pandas no reconoce su tipo de dato como fecha de forma adecuada. Esto se debe, en parte, a que algunas entradas solo registran el año o el año junto con el mes, como se puede observar en la figuro 3-8. Además, las fechas están representadas como cadenas de texto sin separadores que distingan claramente entre año, mes y día. Esta falta de formato estándar dificulta su interpretación y análisis, por lo que es necesario aplicar transformaciones para unificarlas y convertirlas correctamente en objetos de tipo fecha.

FECHA	count
0	1
202210	1
2022090	1
20221	1
202	1

Figura 3-8: Inconsistencia de formato en las fechas

Fuente: Elaboración propia, 2025

Por lo tanto, se procede a depurar todos aquellos registros cuya representación de fecha no cuenta con una extensión de ocho dígitos, como se muestra en la figura 3-9. Esta longitud era necesaria para asegurar que las fechas siguieran el formato esperado (AAAAMMDD) y, por ende, puedan ser transformadas correctamente al tipo de dato fecha.

```
filtered_df = df[(df['FECHA'].str.len() == 8) | (df['FECHA'].isna())].copy()
filtered_df.shape
```

```
(18798, 20)
```

Figura 3-9: Depuración de fechas con formato inconsistente

Fuente: Elaboración propia, 2025

Otro problema identificado es la presencia de fechas que, aunque cuentan con la cantidad correcta de dígitos, no representan una fecha válida ni coherente. Es decir, existían valores que, si bien tienen ocho dígitos, no corresponden a una estructura cronológica real. Para solucionar este inconveniente, la fecha es descompuesta en tres columnas separadas: día, mes y año. Luego, cada una de estas columnas se convierte al tipo de dato entero, como se muestra en la figura 3-10.

```
filtered_df['FECHA'] = filtered_df['FECHA'].astype(str)
filtered_df['AÑO'] = filtered_df['FECHA'].str[:4]
filtered_df['MES'] = filtered_df['FECHA'].str[4:6]
filtered_df['DIA'] = filtered_df['FECHA'].str[6:8]
filtered_df['AÑO'] = filtered_df['AÑO'].astype('Int64')
filtered_df['MES'] = filtered_df['MES'].astype('Int64')
filtered_df['DIA'] = filtered_df['DIA'].astype('Int64')
```

Figura 3-10: Creacion de columnas AÑO, FECHA y DIA

Fuente: Elaboración propia, 2025

Esto permite verificar que los valores se encuentren dentro de rangos válidos, asegurando así que la fecha sea coherente y esté correctamente estructurada. En la columna correspondiente al año, se considera válido cualquier valor comprendido entre 1980 y 2026, con el fin de cubrir un periodo amplio pero realista. Para el mes, se acepta únicamente valores entre 1 y 12, que corresponden a los doce meses del año. Finalmente, en la columna del día, se verifica que los valores estén dentro del rango de 1 a 31, abarcando así todos los posibles días del mes.

Finalmente, una vez verificada la consistencia de las fechas y confirmada su estructura uniforme, se eliminan las columnas DIA, MES y AÑO, ya que fueron generadas únicamente para realizar dicha validación. Posteriormente, se convierte la columna FECHA al tipo de dato *datetime* utilizando la función to_datetime() de la biblioteca Pandas, tal como se muestra en la figura 3-11.

```
filtered_df = filtered_df.drop(columns=['AÑO', 'MES', 'DIA'])
filtered_df["FECHA"] = pd.to_datetime(filtered_df["FECHA"], format='%Y%m%d')
```

Figura 3-11: Transformación de tipo de datos a datatime

Fuente: Elaboración propia, 2025

3.4.4. Tratamiento de valores atípicos

Para este paso se considera usar la corrección por rango intercuartílico (IQR, por sus siglas en inglés), el cual es una medida estadística que describe la dispersión de un conjunto de datos, centrándose en el rango donde se concentra la mayoría de los valores. Se calcula como la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1), es decir, IQR = Q3 - Q1. El primer cuartil representa el valor debajo del cual se encuentra el 25% de los datos, mientras que el tercer cuartil representa el valor debajo del cual se encuentra el 75% de los datos. Así, el rango intercuartílico abarca el 50% central de los datos, proporcionando una medida robusta de dispersión menos sensible a los valores extremos. El IQR se utiliza comúnmente para detectar valores atípicos, cualquier dato que esté por debajo de Q1 - 1.5 × IQR o por encima de Q3 + 1.5 × IQR se considera un posible valor atípico.

En este caso, la columna TOTAL es seleccionada para evaluar la presencia de valores atípicos, ya que representa el monto total pagado en cada factura y será la base para la generación de las series de tiempo con las que los modelos serán entrenados. Se identifican 2296 valores atípicos, los cuales son filtrados, dejando la base de datos con un total de 16,500 registros.

3.5. Agrupación de datos

Cada registro de la base de datos representa una factura individual, lo que implica que en un mismo día pueden haberse pagado varias facturas. Sin embargo, la cantidad de facturas pagadas diariamente no es constante, lo cual dificulta construir una secuencia temporal uniforme para el análisis. Para solucionar este inconveniente, se opta por realizar la sumatoria de todos los pagos registrados en un mismo día, obteniendo así el monto total recaudado diariamente por la EPSA en concepto de cobro de facturas. Este resultado, mostrado en la figura 3-12, se consigue mediante una agrupación de los registros por fecha, sumando los valores de la columna TOTAL.

FECHA	TOTAL
2021-04-01	89.18
2021-04-03	168.92
2021-04-05	452.21
2021-04-06	208.30
2021-04-07	482.62
2021-04-08	537.85
2021-04-09	632.39
2021-04-10	648.63

Figura 3-12: Agrupación de datos por fecha

Fuente: Elaboración propia, 2025

También se establece una frecuencia para la serie de tiempo con el objetivo de organizar los datos de manera regular. Esto es importante porque, al agrupar los registros, pueden omitirse inadvertidamente ciertos días en los que no se registra ningún pago, como se observa en la figura 3-12, donde no aparece un total para el día 2021-04-02. Sin embargo, lo correcto es que la fecha esté presente en la serie, aunque el monto total sea cero. Para definir esta frecuencia se utiliza la función **asfreq()**, que permite especificar distintos tipos de frecuencia, como diaria, semanal o mensual. En este caso particular, es necesario establecer una frecuencia personalizada semanal, pero excluyendo los domingos, como se puede ver en la figura 3-13, ya que este día la EPSA no atiende al público. Para lograrlo, se emplea la clase **CustomBusinessDay()**.

```
custom_freq = CustomBusinessDay(weekmask='Mon Tue Wed Thu Fri Sat')
grouped_df = grouped_df.asfreq(custom_freq)
```

```
grouped_df.isna().sum()
```

```
TOTAL    112
dtype: int64
```

Figura 3-13: Creación de frecuencia personalizada

Fuente: Elaboración propia, 2025

Al establecer la frecuencia se puede observar que evidentemente hay días que estaban siendo omitidos, los cuales aparecen con valor nulo, por lo cual estos son reemplazados por el valor 1. El motivo de remplazar los valores nulos con 1 en vez de 0 es debido a que, en posteriores procedimientos, se calcularán las métricas de evaluación con las que se podrán determinar la calidad de las predicciones. La métrica MAPE no pude ser calculada con valores igual a 0, lo cual es un inconveniente a la hora de utilizar esta métrica, por lo tanto, se toma esta decisión que no representa un cambio significativo en el entrenamiento ni en las predicciones comparándolo en el caso de que los valores nulos fueran remplazados por 0.

Como resultado de la agrupación se obtiene una serie de tiempo que va desde la fecha 2021-04-01 hasta el 2025-01-11, dando un total de 1185 registros. La figura 3-14, muestra gráficamente los datos de la serie.

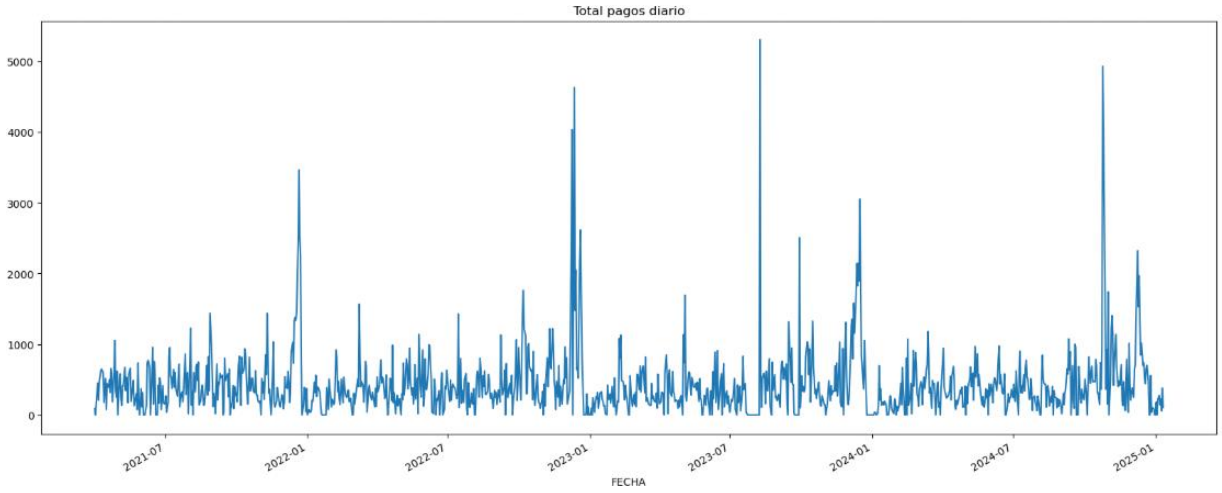


Figura 3-14: Serie de tiempo del total de pagos diarios

Fuente: Elaboración propia, 2025

3.5.1. Tratamiento de valores atípicos en la serie de tiempo

Una vez conformada la serie de tiempo, se procede a realizar un nuevo tratamiento de valores atípicos. Este tratamiento complementa el que ya se había aplicado previamente sobre los datos originales, dado que aún persisten registros con valores extremos en la serie de tiempo generada, que pueden afectar negativamente el proceso de entrenamiento de los modelos predictivos.

En la Figura 3-15, la línea roja indica el límite superior determinado por el rango intercuartílico, por encima del cual los valores son considerados atípicos (outliers). A la derecha de la figura se muestra un gráfico de caja que permite visualizar la distribución de los datos y la presencia de estos valores atípicos. Aunque la mayoría de los valores atípicos se encuentran entre aproximadamente 1.000 y 2.500, también se identifican valores extremos que superan los 5.000.

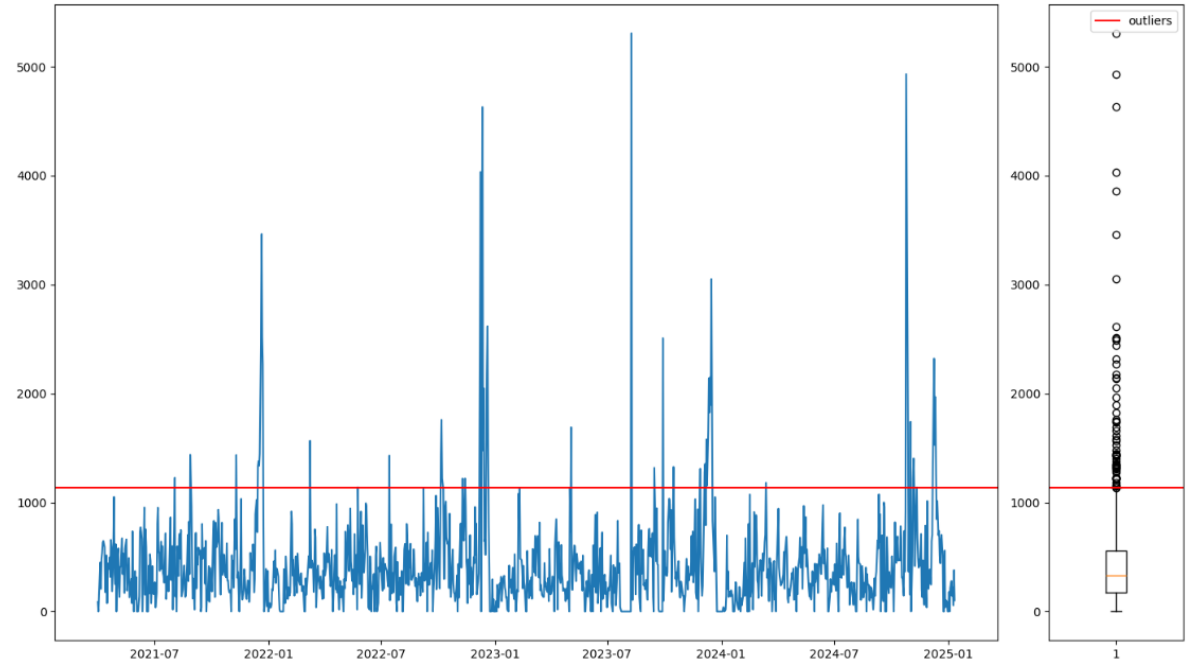


Figura 3-15: Valores atípicos en la serie de tiempo

Fuente: Elaboración propia, 2025

En este caso, para el tratamiento de los valores atípicos se aplica la técnica de *Winsorización*, la cual consiste en reemplazar los valores extremos por otros menos alejados del centro de la distribución, generalmente utilizando percentiles o los límites del rango intercuartílico como referencia. Específicamente, se opta por reemplazar todos los valores superiores al percentil 99 por el propio valor de dicho percentil. Esta decisión se basa en el objetivo de tratar únicamente los valores más extremos, que son precisamente los que superan este umbral. No se elige el límite superior del rango intercuartílico como valor de corte, ya que esto implica reemplazar una mayor cantidad de datos, lo cual puede reducir significativamente la variabilidad de la serie, como se muestra en la figura 3-16. Esto, a su vez, genera una limitación artificial en el comportamiento de los datos, lo que el modelo puede aprender e imitar durante el proceso de predicción, afectando negativamente la calidad y realismo de las proyecciones.

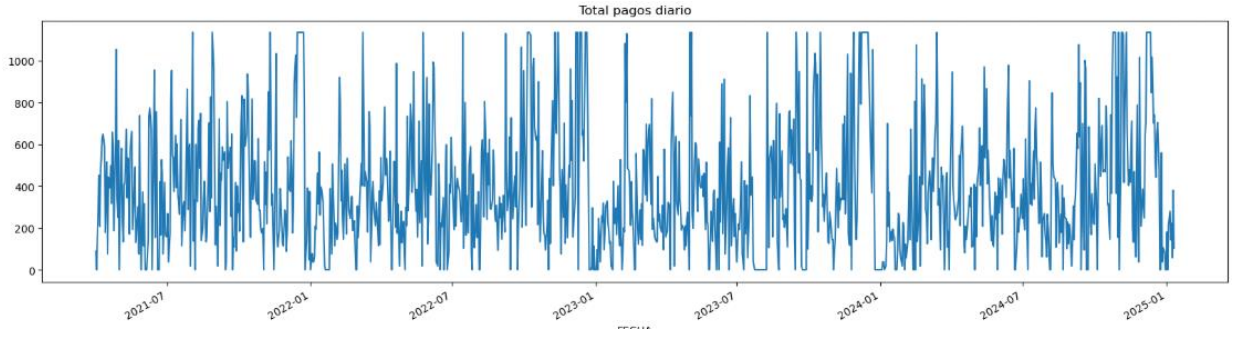


Figura 3-16: Winsorización de los datos por el valor superior del rango intercuartílico

Fuente: Elaboración propia, 2025

3.5.2. División de datos

Una práctica habitual en el entrenamiento de modelos de ML consiste en dividir los datos en conjuntos de entrenamiento y validación. Sin embargo, en el caso de las series de tiempo, esta división no puede hacerse de manera aleatoria como suele hacerse en otros tipos de modelos, debido a que los datos están organizados en función del tiempo y existe una dependencia secuencial entre ellos. Seleccionar los datos aleatoriamente rompe esta relación temporal y afecta la validez del modelo. Por esta razón, la división adecuada en series de tiempo consiste en respetar el orden cronológico, utilizando generalmente el primer 80% de los datos para el entrenamiento y reservando el 20% final para la validación. En total, el conjunto de entrenamiento queda con 948 registros y el de validación con 237 registros.

3.6. Análisis exploratorio

Antes de entrenar un modelo de predicción para una serie de tiempo, es importante analizar algunas de sus características fundamentales. Entre los aspectos más relevantes se encuentran la estacionariedad, la estacionalidad y la autocorrelación. Evaluar estos factores permite entender mejor el comportamiento de la serie y tomar decisiones informadas sobre el tipo de modelo más adecuado para trabajar con los datos. A continuación, se presenta un análisis detallado de cada uno de estos aspectos aplicado a la serie de tiempo obtenida.

3.6.1. Estacionariedad

Para verificar la estacionariedad de la serie de tiempo se utiliza la prueba de Dickey-Fuller, para la cual, la biblioteca Statsmodels nos ofrece la implementación de esta prueba en la función **adfuller()**. Los resultados se muestran en la figura 3-17.

```
sts.adfuller(grouped_df.TOTAL)
```

```
(-9.231847951276192,
 1.6701603812531289e-15,
 12,
 1172,
 {'1%': -3.4359418774356696,
  '5%': -2.864009233598981,
  '10%': -2.5680846730014326},
 16835.134157640336)
```

Figura 3-17: Resultados de la prueba Dickey-Fuller

Fuente: Elaboración propia, 2025

En esta respuesta se puede observar los siguientes valores:

- Estadístico de prueba: -9.2318
- Valor p: 1.6702e-15
- Número de rezagos usados: 12
- Número de observaciones usadas: 1172
- Valores críticos (para niveles de confianza del 1%, 5% y 10%)

- Valor de la estadística de información: 16835.1341 (este último no siempre es tan relevante para la interpretación básica)

Como se puede observar, el valor p es mucho menor que el nivel de significancia comúnmente utilizado (0.05), se rechaza la hipótesis nula que plantea la existencia de una raíz unitaria, es decir, que la serie es estacionaria. Además, el estadístico de prueba (-9.2318) es menor que todos los valores críticos a los niveles del 1%, 5% y 10%, lo cual refuerza aún más la conclusión de rechazar la hipótesis nula. Por lo tanto, se concluye que la serie de tiempo analizada es estacionaria, lo que significa que sus propiedades estadísticas, como la media y la varianza, se mantienen constantes a lo largo del tiempo.

3.6.2. Descomposición de la serie de tiempo

Para analizar la tendencia y la estacionalidad de la serie de tiempo se utiliza la herramienta **seasonal_descompose()** de la biblioteca Statsmodels, esta nos permite descomponer en tres elementos nuestra serie de tiempo, como se muestra en la figura 3-18.

```
aditivo = seasonal_decompose(grouped_df.TOTAL, model="additive", period=26)
aditivo.plot()
plt.show()
```

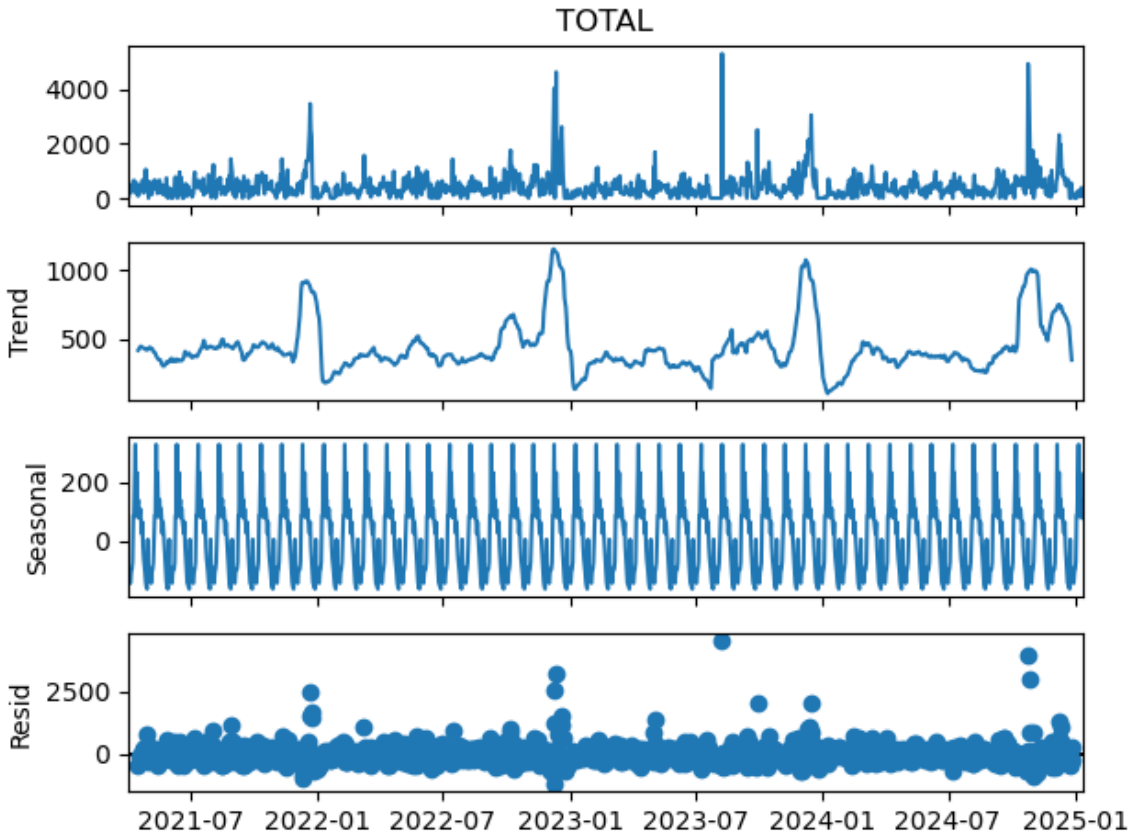


Figura 3-18: Descomposición de la serie de tiempo

Fuente: Elaboración propia, 2025

En el resultado de esta descomposición se pueden ver cuatro componentes principales:

- **Serie original**: La primera gráfica muestra el comportamiento original de la serie de tiempo. Se observan varios picos pronunciados a lo largo del período analizado, lo que indica la existencia de eventos esporádicos donde los cobros fueron significativamente mayores al promedio.
- **Tendencia (Trend):** La segunda gráfica representa la tendencia de la serie. Aquí se evidencia una evolución a lo largo del tiempo, donde se aprecian ciertos incrementos y descensos graduales. La tendencia captura el comportamiento de largo plazo, aislando los efectos de la estacionalidad y las fluctuaciones aleatorias.
- **Estacionalidad (Seasonal):** La tercera gráfica muestra la componente estacional. Se observa un patrón claramente repetitivo, con ciclos que se repiten aproximadamente cada 26 días, lo cual indica que existe una estacionalidad regular en la serie. Este comportamiento sugiere que los cobros presentan variaciones sistemáticas en función del tiempo, probablemente relacionadas con factores como la periodicidad de facturación o el comportamiento de pago de los usuarios.

  Durante el análisis de la estacionalidad, se prueban inicialmente períodos de 1 y 6 días, considerando que la EPSA opera todos los días excepto domingos. Sin embargo, no se identifican patrones estacionales claros, ya que las variaciones son irregulares. Finalmente, al utilizar un período de 26 días (correspondiente a los días de un mes sin contar domingos), se logra evidenciar una estacionalidad más consistente en los cobros.
- **Residuo (Resid):** La cuarta gráfica representa el residuo, es decir, la parte de la serie que no es explicada ni por la tendencia ni por la estacionalidad. En esta gráfica se observan fluctuaciones aparentemente aleatorias, aunque existen algunos valores atípicos destacados, donde los residuos son considerablemente más altos que el resto.

3.6.3. Autocorrelación

Se realizo un análisis de autocorrelación a la serie de tiempo con ayuda de la función **plot_acf()** de la biblioteca Statsmodels. Esto facilita detectar si existe dependencia temporal entre los valores de la serie.

Como se observa en la figura 3-19, se realiza un análisis de 40 rezagos, es decir, los valores son analizados con los 40 valores anteriores a él. Se puede observar que en los primero 10 rezagos muestran autocorrelaciones positivas y significativas, la franja azul indica el rango de insignificancia. Esto indica que el valor de un día está fuertemente relacionado con los valores de los 10 días anteriores inmediatos.

Aunque el análisis de autocorrelación muestra la relación entre los valores de la serie de tiempo en diferentes rezagos, no permite identificar directamente cuáles rezagos influyen de manera más significativa en el valor actual, ya que las correlaciones observadas están afectadas por la influencia de rezagos anteriores. Por este motivo, es necesario complementar el análisis con un estudio de autocorrelación parcial (PACF), el cual mide la correlación entre el valor actual y un rezago específico, eliminando el efecto de los rezagos intermedios.

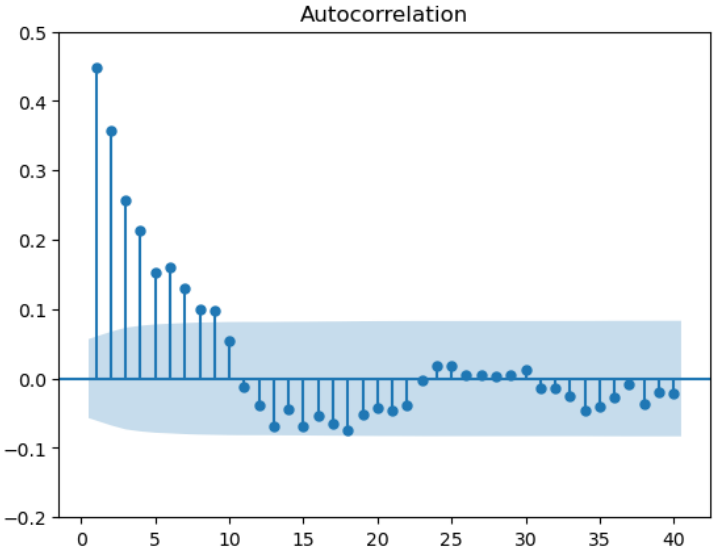


Figura 3-19: Análisis de autocorrelación de la serie de tiempo

Fuente: Elaboración propia, 2025

En la figura 3-20, se observa que los primeros rezagos presentan correlaciones significativas, en especial el rezago 1, seguido por una menor influencia en el rezago 2. A partir del rezago 3 en adelante, la mayoría de los coeficientes caen dentro del intervalo de confianza, indicando que no presentan una correlación parcial significativa con el valor actual de la serie. Esto sugiere que el valor inmediato anterior y, en menor medida, el segundo anterior, tienen un efecto directo sobre el valor actual, mientras que los rezagos más lejanos no contribuyen de manera relevante una vez controlada la influencia de los primeros. Este comportamiento es consistente con una estructura de dependencia a corto plazo en la serie de tiempo.

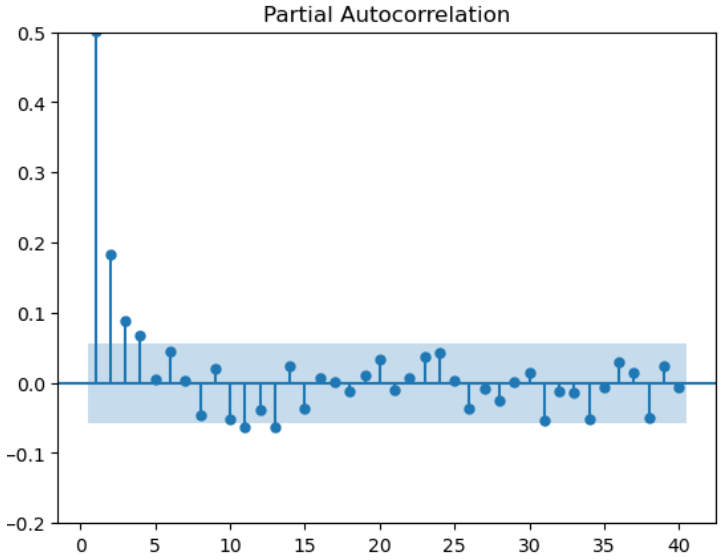


Figura 3-20: Análisis de autocorrelación parcial de la serie de tiempo

Fuente: Elaboración propia, 2025

3.7. Entrenamiento de los modelos

En esta sección se expone el proceso de entrenamiento de los modelos predictivos. Se detallan las configuraciones a utilizar y los conjuntos de datos que se emplean. Los modelos que se emplean son LSTM y Prophet.

3.7.1. LSTM

El entrenamiento de un modelo LSTM requiere una preparación previa adecuada de los datos. Aunque las ANN suelen ser robustas frente a datos con un preprocesamiento limitado, llevar a cabo esta etapa resulta beneficioso para asegurar un aprendizaje efectivo. A continuación, se describen los elementos clave del proceso.

- **Escalado de datos**

El escalado de datos es una técnica de preprocesamiento que consiste en transformar las variables numéricas para que se encuentren dentro de un mismo rango, normalmente entre 0 y 1. Esta transformación es especialmente importante en modelos como las redes neuronales, ya que mejora la eficiencia del entrenamiento, evita que ciertas variables dominen sobre otras y contribuye a una mejor convergencia del modelo. Para este caso se utiliza la técnica de normalizado Min-Max, el cual utiliza el valor mínimo y máximo de los datos para realizar el escalado entre los valores 0 y 1, como se muestra en la figura 3-21.

FECHA	TOTAL
2024-04-11	0.226934
2024-04-12	0.087541
2024-04-13	0.229032
2024-04-15	0.285109
2024-04-16	0.217523
...	...
2025-01-07	0.059330
2025-01-08	0.091875
2025-01-09	0.023641
2025-01-10	0.157940
2025-01-11	0.042586

Figura 3-21: Datos de la serie de tiempo escalados

Fuente: Elaboración propia, 2025

- **Creación de variables *y***

En el entrenamiento de redes neuronales, es fundamental proporcionar explícitamente tanto los datos de entrada como las respuestas esperadas, también conocidas como variables objetivo *y*. Esto permite al modelo aprender la relación entre los patrones de entrada y los resultados deseados. En el caso de las series de tiempo, una práctica común consiste en utilizar una secuencia de valores pasados, por ejemplo, los valores correspondientes a diez días consecutivos como entrada (*x*), y el valor del día siguiente, el undécimo, como la salida esperada (*y*).

En nuestro caso particular, tanto los datos de entrada como las respuestas objetivo, provienen de la misma serie de tiempo, por lo que es necesario generar el conjunto de variables *y* de forma explícita. Para ello, se crea una función que se encarga de construir estas secuencias de valores objetivo, la cual se muestra en la figura 3-22, asegurando así que el modelo pueda aprender correctamente a predecir valores futuros en función del comportamiento pasado.

```
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        v = X.iloc[i:(i + time_steps)].values
        Xs.append(v)
        ys.append(y.iloc[i + time_steps])
    return np.array(Xs), np.array(ys)
```

Figura 3-22: Implementación de la función que crea el conjunto de variables objetivo

Fuente: Elaboración propia, 2025

Esta función nos devuelve tanto el conjunto de las variables *x* como la de las variables *y*. El parámetro **time_steps** define la cantidad de observaciones consecutivas que se utilizarán como entrada para predecir el siguiente valor objetivo. Es decir, para cada ejemplo de entrenamiento, se toma una secuencia de "time_steps" valores del conjunto de entrada y se asocia con el valor de salida correspondiente al momento siguiente. De esta manera, el modelo aprende a establecer una relación entre un conjunto de datos pasados y su evolución futura.

- **Arquitectura**

La arquitectura del modelo LSTM a implementar se compone de cuatro capas LSTM, cada una seguida de una capa de Dropout para la regularización, como se ve en la figura 3-23. Cada capa LSTM cuenta con 250 unidades ocultas (neuronas) y además están configuradas con un time_step que puede variar entre 1, 6 y 26, ya que son las posibles variaciones estacionales que se identifica en la serie de tiempo. Cada capa LSTM esta seguida de una capa Dropout, que tiene la función de reducir el riego de sobreajuste del modelo, es decir, para evitar que el modelo memorice los datos en vez de aprender de ellos para realizar una predicción más generalizada. Al final, la capa Dense, ubicada al final del modelo, tiene la función de transformar el vector de salida de 250 características, generado por la última capa LSTM, en un único valor de predicción, permitiendo así que el modelo entregue un resultado final coherente con el objetivo de predicción de la serie temporal.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 250)	252,000
dropout (Dropout)	(None, 1, 250)	0
lstm_1 (LSTM)	(None, 1, 250)	501,000
dropout_1 (Dropout)	(None, 1, 250)	0
lstm_2 (LSTM)	(None, 1, 250)	501,000
dropout_2 (Dropout)	(None, 1, 250)	0
lstm_3 (LSTM)	(None, 250)	501,000
dropout_3 (Dropout)	(None, 250)	0
dense (Dense)	(None, 1)	251

Figura 3-23: Arquitectura de la red LSTM

Fuente: Elaboración propia, 2025

- **Entrenamiento**

Ya definido los datos y la arquitectura de la LSTM, se procede a entrenarlo. Para esto se llama al método **fit()** del modelo, como se muestra en la figura 3-24, en el cual se emplearon 100 épocas (epochs=100), lo que indica que el modelo recorre el conjunto de entrenamiento completo 100 veces durante el proceso de aprendizaje. Se definió un tamaño de lote de 32 (batch_size=32), lo que significa que los datos se dividen en pequeños grupos de 32 muestras que se procesan simultáneamente para actualizar los pesos.

```
history_1 = model_1.fit(X_train_1,
                        y_train_1,
                        epochs=100,
                        batch_size=32,
                        shuffle=False)
```

Figura 3-24: Parámetros de entrenamiento de la LSTM

Fuente: Elaboración propia, 2025

Con el objetivo de explorar cómo la estacionalidad de la serie afecta el desempeño del modelo, se realizan tres entrenamientos utilizando distintos valores para el parámetro time_steps para la secuencia de los datos *y* (y_train). En el primer caso se utiliza un time_step de 1, asumiendo una posible estacionalidad diaria. Posteriormente, se entrena el modelo con un time_step de 6, considerando que los cobros podrían presentar un patrón semanal, dado que la atención se realiza 6 días a la semana. Finalmente, se realiza un tercer entrenamiento con un time_step de 26, basado en la hipótesis de una estacionalidad mensual, descontando los domingos.

3.7.2. Prophet

El uso del modelo Prophet requiere preparar los datos en un formato específico, definir los eventos especiales que pueden alterar la tendencia, y ajustar los parámetros que guían el comportamiento del modelo durante la predicción. A continuación, se describe los procesos de esta preparación incluyendo el entrenamiento.

- **Adaptación de la serie de tiempo**

Para entrenar correctamente el modelo Prophet, es necesario adaptar el conjunto de datos al formato que este requiere. Específicamente, las columnas que representan la fecha y el valor a predecir deben renombrarse como *ds* (date stamp) y *y* (value), respectivamente, como se muestra en la figura 3-25. Esta convención es obligatoria, ya que Prophet está diseñado para reconocer automáticamente estas etiquetas y usarlas como entrada para el entrenamiento del modelo. Por lo tanto, aunque los datos originales puedan tener nombres más descriptivos, es fundamental realizar este cambio para asegurar la compatibilidad con el modelo y evitar errores durante su ejecución. Esta conversión se aplica tanto al conjunto de entrenamiento como al conjunto de validación.

	ds	y
0	2021-04-01	89.18
1	2021-04-02	1.00
2	2021-04-03	168.92
3	2021-04-05	452.21
4	2021-04-06	208.30

Figura 3-25: Serie de tiempo con las columnas renombradas

Fuente: Elaboración propia, 2025

- **Creación de los *holidays***

Como se explicó en la sección 2.5.1, la funcionalidad de *holidays* en Prophet permite identificar y modelar comportamientos inusuales dentro de la serie de tiempo, de modo que estos sean considerados de forma más precisa durante el entrenamiento del modelo. En esta serie de tiempo, se detecta un patrón recurrente que ocurre cada último mes del año, como se muestra en la figura 3-26. Este comportamiento puede estar relacionado con el hecho de que muchos usuarios tienden a ponerse al día con pagos atrasados en diciembre, mes en el que generalmente se perciben mayores ingresos, lo que facilita saldar facturas acumuladas de meses anteriores.

Para definir los *holidays* en Prophet, es necesario especificarlos en un DataFrame, el cual posteriormente se pasa como parámetro durante la creación del modelo. En este caso, se considera como *holiday* el 5 de diciembre de cada año, asignando un rango de consideración de cinco días antes y cinco días después de esta fecha, el cual, busca capturar el comportamiento inusual que puede iniciarse y extenderse dentro de ese intervalo. La figura 3-27 muestra cómo es definida esta configuración.

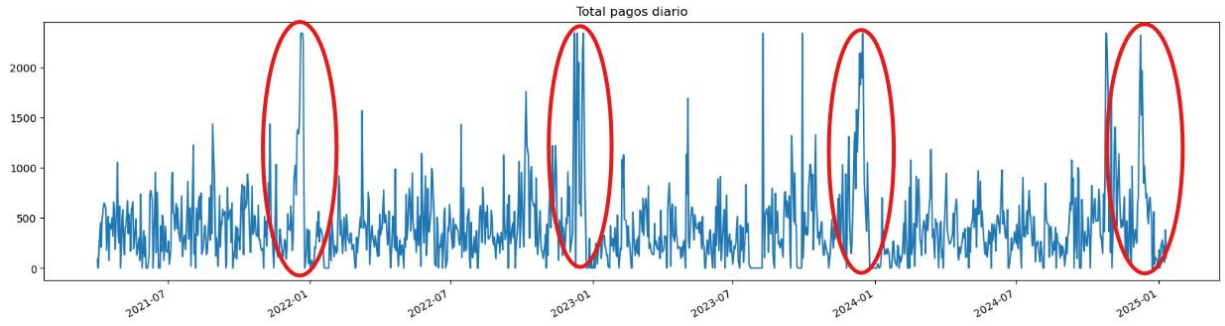


Figura 3-26: *holidays* en la serie de tiempo

Fuente: Elaboración propia, 2025

```
fin_de_año = pd.DataFrame({
    'holiday': 'fin de año',
    'ds': pd.to_datetime(['2021-12-05', '2022-12-05', '2023-12-05', '2024-12-05']),
    'lower_window': -5,
    'upper_window': 5
})
```

Figura 3-27: Definición de los *holidays*

Fuente: Elaboración propia, 2025

- ## Entrenamiento

Para el entrenamiento del modelo Prophet, se realiza una configuración específica de sus parámetros con el objetivo de mejorar la capacidad predictiva del modelo frente a las características particulares de la serie temporal en estudio.

En primer lugar, como se muestra en la figura 3-28, se incorpora el parámetro holidays, en la que se asigna el DataFrame que se creó en la anterior sección. Prophet toma esta información para ajustar mejor la curva de predicción en esos intervalos.

Se ajusta también el parámetro changepoint_range al valor de 0.95. Este parámetro define la proporción inicial de la serie de tiempo en la que Prophet buscará posibles puntos de cambio en la tendencia. Por defecto, el valor es 0.8, pero al ampliar este rango al 95% del total de datos, permite que el modelo identifique cambios estructurales en la tendencia más cerca del final del periodo de entrenamiento, lo cual es relevante en este caso ya que existe un *holiday* que podría ser excluido si se toma el 80% de los datos.

Asimismo, se modifica el parámetro yearly_seasonality, que por defecto es un valor booleano y, cuando está habilitado, Prophet estima automáticamente un valor adecuado para la estacionalidad anual. Sin embargo, en este caso se establece manualmente en 50, lo que indica que se utilizarán 50 términos de la serie de Fourier para representar esta estacionalidad. Esto proporciona mayor flexibilidad al modelo para capturar patrones anuales complejos o no lineales que no serían captados adecuadamente con los valores predeterminados, que generalmente oscilan entre 10 y 20 términos.

Además, se añade una estacionalidad personalizada de tipo mensual utilizando **add_seasonality()**. Prophet no incluye una estacionalidad mensual por defecto, por lo que es necesario definirla manualmente. El período de 30.5 días representa la duración promedio de un mes, y se usa un fourier_order de 15 para permitir que el modelo capte variaciones mensuales con suficiente detalle. Esta configuración es especialmente útil ya que se han identificado ciclos mensuales recurrentes en el comportamiento del consumo de agua.

Finalmente, se entrena el modelo con la función **fit()**, en el que se manda como parámetro el DataFrame que contiene la serie de tiempo de entrenamiento.

```
prophet = Prophet(holidays=fin_de_año, changepoint_range=0.95, yearly_seasonality=50)
prophet.add_seasonality(name='monthly', period=30.5, fourier_order=15)
prophet.fit(df_total_prophet)
```

Figura 3-28: Definición de los parámetros de Prophet

Fuente: Elaboración propia, 2025

4. Análisis de Resultados y Discusión

En esta sección se presentan y analizan los resultados obtenidos tras el entrenamiento y evaluación de los modelos propuestos. Se evalúa los rendimientos utilizando métricas estadísticas relevantes para comparar los modelos bajo distintas configuraciones. Posteriormente, se comparan estos resultados con los obtenidos en un proyecto similar, con el objetivo de contextualizar y discutir los hallazgos alcanzados, identificando tanto las fortalezas como las posibles limitaciones del enfoque adoptado en este estudio.

4.1. Resultados del tratamiento de los datos

La base de datos cuenta en su forma íntegra con 19.899 datos inicialmente. Los procesos de tratamiento de los datos que se realizaron fueron: tratamiento de valores nulos, tratamiento de fechas, tratamiento de valores atípicos, agrupación de los datos por fechas y como ultimo la división de los datos para el entrenamiento y la validación del modelo.

4.1.1. Valores nulos

En el tratamiento de valores nulos, se identificaron un total de 301.337 valores faltantes en toda la base de datos. Para abordar esta situación, se optó por eliminar un poco más de la mitad de las columnas, así como algunas filas cuyos registros contenían una gran proporción de datos nulos.

Posteriormente, al agrupar los registros por fecha, fue necesario realizar un nuevo tratamiento de valores nulos. Esto debido a que algunas fechas no estaban presentes en la serie de tiempo original, y al establecer una secuencia cronológica continua, estas fechas ausentes aparecieron con registros vacíos, un total de 112 fechas. Para permitir el cálculo correcto del MAPE, se decidió reemplazar estos valores nulos con el valor de 1 en lugar de 0, evitando así que el cálculo del resultado arrojara un valor infinito.

En total, se eliminaron 1086 registros, lo cual representa solo el 5,46% del total, una proporción baja que no afecta significativamente la representatividad del conjunto de datos. Asimismo, se eliminaron 33 columnas, equivalentes al 60% del total, principalmente porque la mayoría de ellas presentaban un alto porcentaje de valores faltantes o no aportaban información relevante para los objetivos del proyecto. Esta depuración permitió reducir la dimensionalidad del conjunto de datos y centrarse en las variables más útiles para el análisis predictivo, mejorando así la calidad del modelo.

4.1.2. Tratamiento de fechas

En cuanto al tratamiento de las fechas, también fue necesario realizar una depuración específica, ya que se identificaron ciertos registros con valores inconsistentes en este campo. Algunas filas contenían fechas incompletas, mal formateadas o directamente no representaban fechas válidas, lo cual impedía su correcta interpretación y análisis dentro de la serie temporal. Dado que el componente temporal es fundamental para el desarrollo del modelo predictivo, se optó por eliminar estos registros, ya que su inclusión podría afectar negativamente la calidad del modelo y generar resultados erróneos. En total, se eliminaron 7

registros, lo que representa aproximadamente el 0,04% del conjunto de datos. Esta proporción es mínima y no afecta de manera significativa la representatividad ni la robustez del análisis general.

4.1.3. Valores atípicos

Para el tratamiento de valores atípicos se aplicó la técnica del rango intercuartílico (IQR), enfocándose en la columna TOTAL, ya que esta fue la variable utilizada posteriormente para la agrupación por fechas y representa directamente los montos recaudados. A través de este método se identificaron y filtraron 2296 registros considerados fuera de rango. Estos registros representaron aproximadamente el 11.54% del total de datos. Si bien este porcentaje podría parecer elevado, se considera una cantidad no significativa en relación con el volumen total de datos disponibles, que supera ampliamente los 16.000 registros. Por tanto, la eliminación no compromete la representatividad ni la calidad del conjunto de datos restante, y permite contar con una base más coherente y adecuada para el análisis y la predicción.

4.2. Agrupación de datos por fechas

Se generó una serie de tiempo a partir de los datos originales, agrupando los registros por fecha y sumando los valores correspondientes a la columna TOTAL. Esta transformación permitió consolidar la información diaria en una estructura adecuada para el análisis temporal. Como resultado, se obtuvo una serie compuesta por 1185 registros, donde cada fila refleja el total recaudado en un día específico a lo largo del período observado.

Antes de proceder con el entrenamiento de los modelos, fue necesario realizar un tratamiento adicional sobre la serie con el fin de reducir el impacto de valores atípicos que aún persistían. Para ello, se aplicó la técnica de *Winsorización*, utilizando como umbral el percentil 99. Esta estrategia consistió en identificar los valores más extremos de la serie y sustituirlos por el valor del percentil, ya que podían distorsionar las predicciones si eran considerados durante el ajuste del modelo. En total, se sustituyeron 12 observaciones que superaban dicho umbral.

Una vez tratada la serie, se procedió con su división en dos subconjuntos, respetando el orden cronológico de los datos para evitar fugas de información: los primeros 948 registros fueron utilizados como conjunto de entrenamiento y los 237 restantes como conjunto de validación.

4.3. Resultados de los entrenamientos

Una vez procesados y preparados los datos, se procedió con el entrenamiento de los modelos. A continuación, se detallan las configuraciones utilizadas durante el entrenamiento, así como los resultados obtenidos en términos de desempeño y precisión del modelo.

4.3.1. Resultados de la LSTM

Las tres gráficas en la figura 4-1, representan la evolución de la función de pérdida durante el entrenamiento del modelo LSTM para distintos valores de time_steps (1, 6 y 26). En particular, los modelos entrenados con time_steps 6 y 26 muestran comportamientos muy similares: ambas curvas presentan una disminución sostenida y progresiva de la pérdida, con leves fluctuaciones, lo que sugiere

que el modelo logra capturar relaciones temporales significativas a lo largo del entrenamiento. Esto podría deberse a que tanto 6 como 26 pasos permiten al modelo incorporar información contextual suficiente para aprender patrones semanales o mensuales.

En contraste, el modelo entrenado con time_steps igual a 1 muestra una curva con una caída más rápida al inicio, pero que se estabiliza de manera prematura y con mayor variabilidad relativa. Este comportamiento indica que, con solo un paso temporal, el modelo tiene un contexto limitado para aprender relaciones temporales más complejas. Este patrón se mantuvo de forma consistente a lo largo de varias ejecuciones, lo que refuerza la idea de que un mayor número de pasos temporales permite capturar mejor la estructura de dependencia temporal de los datos. Sin embargo, como se analizará más adelante, este modelo, a pesar de su simplicidad y de no aprovechar plenamente la dinámica temporal de la serie, logra superar a otras configuraciones en métricas de error absoluto. Esto sugiere que, en ciertos contextos, un enfoque más simple puede ser suficiente o incluso más robusto frente a la variabilidad y el ruido de los datos.

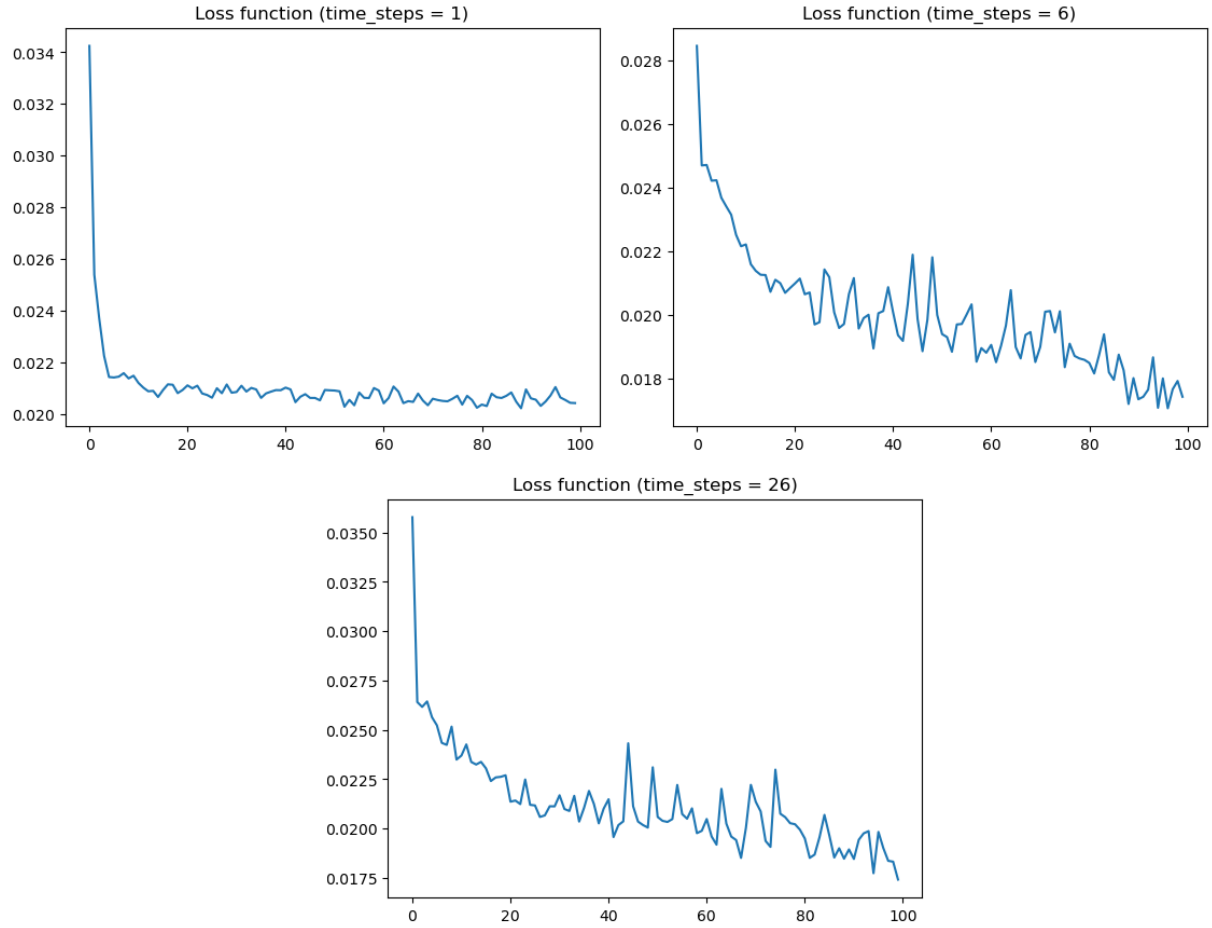


Figura 4-1: Funciones de pérdida con distintos time_steps

Fuente: Elaboración propia, 2025

4.3.2. Resultados del Prophet

La función **plot_components()** de Prophet, permite visualizar los distintos componentes que el modelo identificó en la serie temporal tras el entrenamiento. Estos componentes son fundamentales para interpretar el comportamiento subyacente de los datos, ya que separan efectos de tendencia, estacionalidades y eventos especiales.

La figura 4-2 corresponde a la tendencia, donde se observa una disminución progresiva en los valores estimados desde abril de 2024 hasta enero de 2025. Esto indica que, en general, la recaudación presenta una caída sostenida durante el periodo analizado.

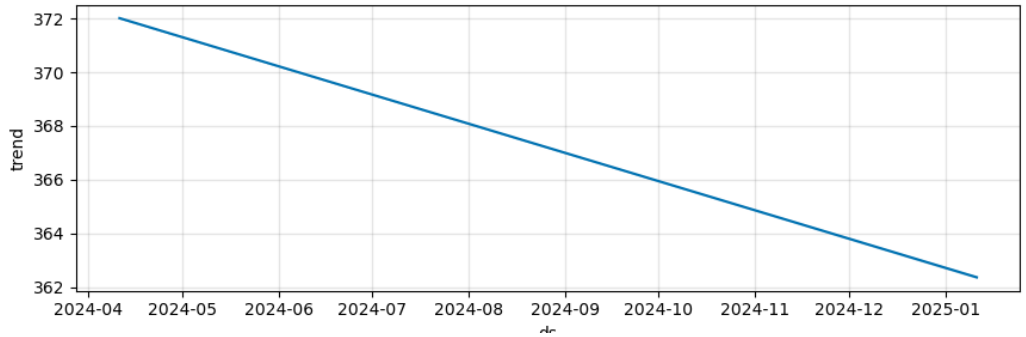


Figura 4-2: Tendencia identificada por Prophet

Fuente: Elaboración propia, 2025

La figura 4-3 refleja el efecto de los *holidays*. En este caso, se configuró el 5 de diciembre como una fecha clave, con un margen de cinco días antes y después para capturar su efecto. El gráfico evidencia variaciones abruptas hacia finales de diciembre, con valores tanto positivos como negativos, lo que sugiere un comportamiento inusual en esos días. Esto se corresponde con el análisis previo que indicaba que muchos usuarios tienden a cancelar sus deudas acumuladas en el último mes del año.

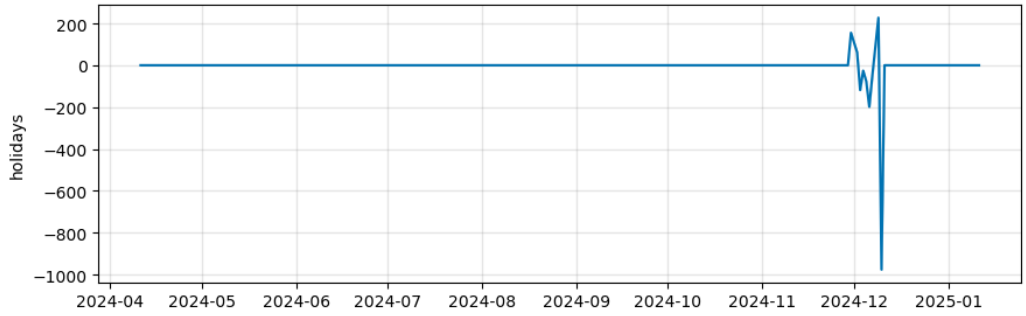


Figura 4-3: Efecto de los *holidays* identificado por Prophet

Fuente: Elaboración propia, 2025

En cuanto la figura 4-4, se muestra la estacionalidad semanal (weekly), se observa que durante los días de la semana se presenta un impacto positivo más acentuado en la recaudación, mientras que los fines de semana, en particular los domingos, tienen una influencia negativa. Este patrón semanal está relacionado con los horarios y días de atención al cliente, ya que los sábados solo se atiende hasta medio día y los domingos no hay atención.

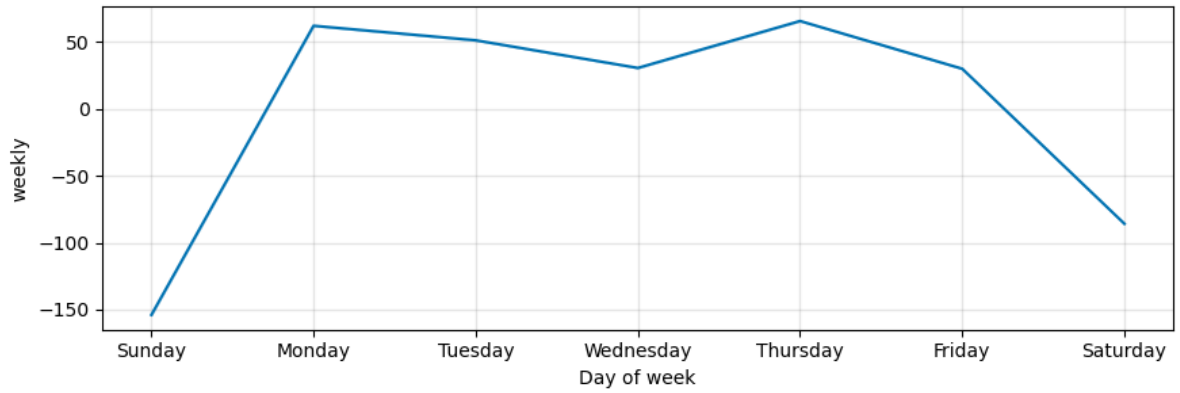


Figura 4-4: Estacionalidad semanal identificada por Prophet

Fuente: Elaboración propia, 2025

La figura 4-5, que corresponde a la estacionalidad anual (yearly) muestra una serie de fluctuaciones a lo largo del año, con picos y valles bien definidos. El modelo logró capturar estos comportamientos complejos gracias al uso de un mayor número de términos de Fourier (yearly_seasonality=50), lo cual permite una mayor flexibilidad en la representación de variaciones estacionales que ocurren de forma cíclica cada año. El pico más alto se presenta en diciembre, nuevamente reforzando la hipótesis de pagos acumulados en ese periodo.

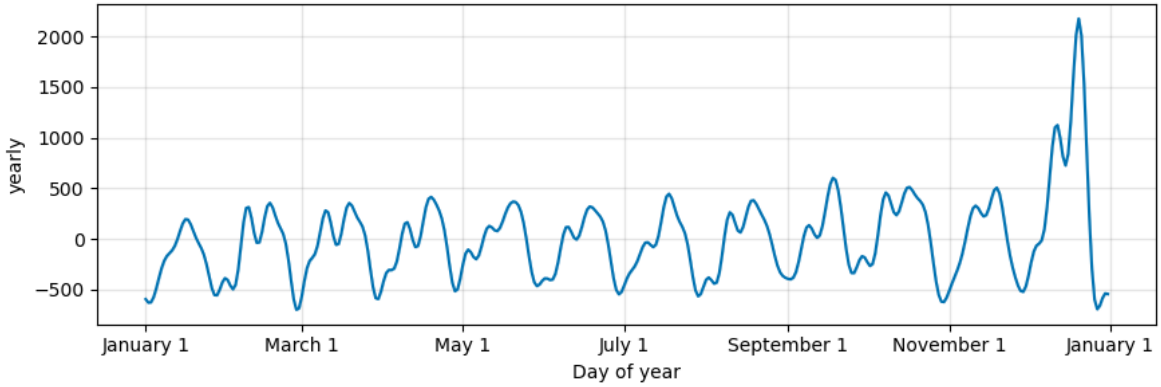


Figura 4-5: Estacionalidad anual identificada por Prophet

Fuente: Elaboración propia, 2025

Por último, en la figura 4-6 se muestra la estacionalidad mensual personalizada (monthly) con un período de 30.5 días y un orden de Fourier de 15. Este componente revela oscilaciones recurrentes dentro de cada mes, que no serían capturadas adecuadamente por las estacionalidades predeterminadas. La incorporación de esta componente permite que el modelo se ajuste con mayor precisión a fluctuaciones periódicas de menor escala, propias del comportamiento mensual de los usuarios.

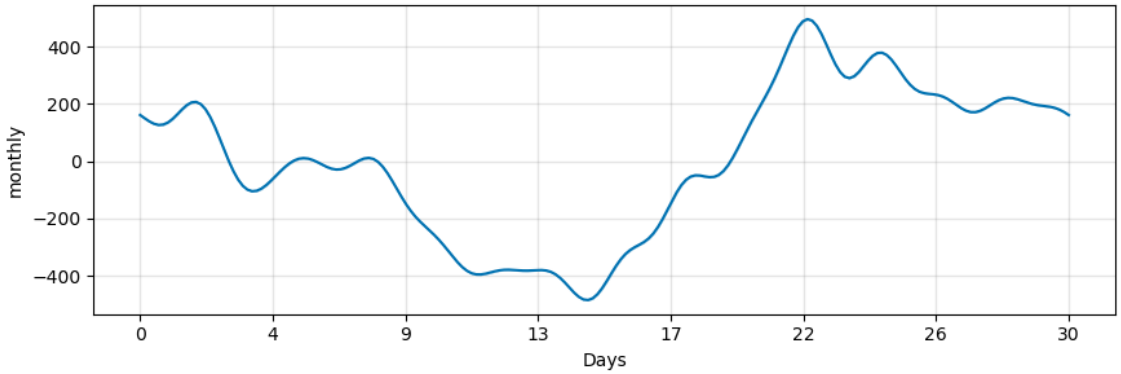


Figura 4-6: Estacionalidad mensual identificada por Prophet

Fuente: Elaboración propia, 2025

4.4. Resultados de las predicciones

Tras completar el proceso de entrenamiento, se aplicó los modelos al conjunto de validación para realizar las predicciones sobre la recaudación diaria. En esta sección se presentan los resultados obtenidos, evaluando la precisión del modelo mediante las métricas RMSE, MAE y MAPE; además, analizando visualmente la correspondencia entre los valores predichos y los valores reales.

Al analizar las métricas de error mostradas en la tabla 4-1, se observa que el modelo LSTM con time_steps igual a 1 obtiene el menor RMSE (342.58) y MAE (231.96), lo que indica que, en términos absolutos, sus predicciones están más próximas a los valores reales del conjunto de prueba. Además, también presenta el MAPE más bajo entre las configuraciones LSTM (29.92%), lo que sugiere que no solo es más preciso en magnitud, sino también proporcionalmente más exacto en relación con los valores reales. El LSTM con time_steps = 6 muestra un leve aumento en el RMSE (348.23) y el MAE (237.66), acompañado de un MAPE mayor (31.33%), lo cual evidencia una ligera pérdida de precisión tanto absoluta como relativa al ampliar la ventana temporal. Por su parte, el modelo LSTM con time_steps = 26, aunque intenta capturar patrones de más largo plazo, obtiene el peor desempeño entre las configuraciones LSTM, con un RMSE de 396.50, un MAE de 251.18 y un MAPE de 33.16%, reflejando un sacrificio en precisión al extender el horizonte de entrada. Finalmente, el modelo Prophet, que no se basa en ventanas temporales, presenta un RMSE de 401.20 y un MAE de 258.12, superiores a los de los modelos LSTM, pero logra el MAPE más bajo entre todos los modelos evaluados (17.82%). Este resultado indica una alta precisión relativa en sus predicciones, lo que lo convierte en una alternativa destacable cuando el objetivo es minimizar el error porcentual, especialmente en contextos con variaciones de escala significativas.

MODELO	TIME STEPS	RMSE	MAE	MAPE
LSTM	1	342.58	231.96	29.92
LSTM	6	348.23	237.66	31.33
LSTM	26	396.50	251.18	33.16
Prophet	-	401.20	258.12	17.82

Tabla 4-1: Métricas de las predicciones correspondientes

Fuente: Elaboración propia, 2025

Por último, se visualizan las predicciones junto a los valores reales con el fin de evidenciar las diferencias previamente identificadas.

Hablando de las predicciones con LSTM, al compararlos con diferentes valores de time_steps, se puede apreciar que cada configuración responde de manera distinta a los patrones presentes en la serie temporal. En particular, el modelo entrenado con time_steps igual a 1, que se encuentra en la figura 4-7, muestra una alta precisión para capturar pequeñas fluctuaciones. Las predicciones siguen de cerca los valores reales cuando las variaciones no son muy pronunciadas, lo que indica que este modelo es muy sensible a los cambios locales. Sin embargo, al enfrentarse a fluctuaciones más abruptas, aunque logra detectar su ocurrencia, tiende a subestimar su magnitud.

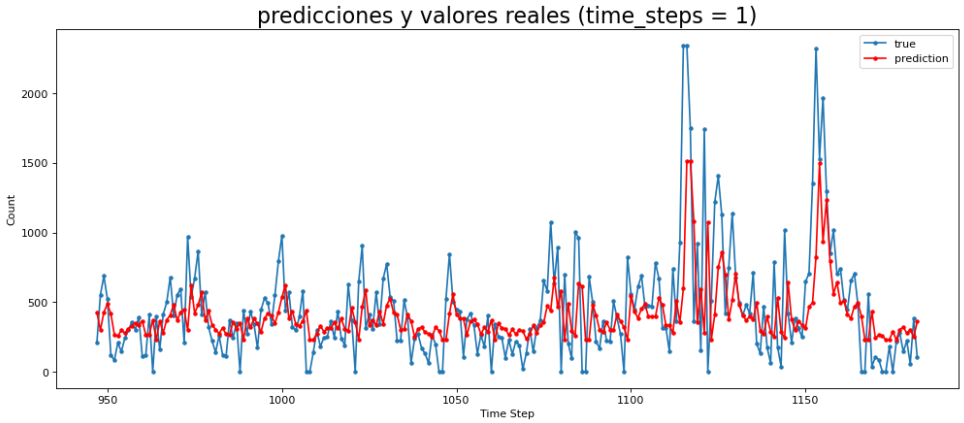


Figura 4-7: Predicción del LSTM con time_steps = 1

Fuente: Elaboración propia, 2025

En contraste, el modelo entrenado con time_steps igual a 26, mostrado en la figura 4-8, exhibe un comportamiento opuesto, sus predicciones representan de forma más precisa las grandes fluctuaciones, acercándose mejor a los valores reales cuando se producen cambios bruscos. No obstante, pierde precisión frente a pequeñas variaciones, generando una línea de predicción más suavizada que no replica con exactitud las ondulaciones menores.

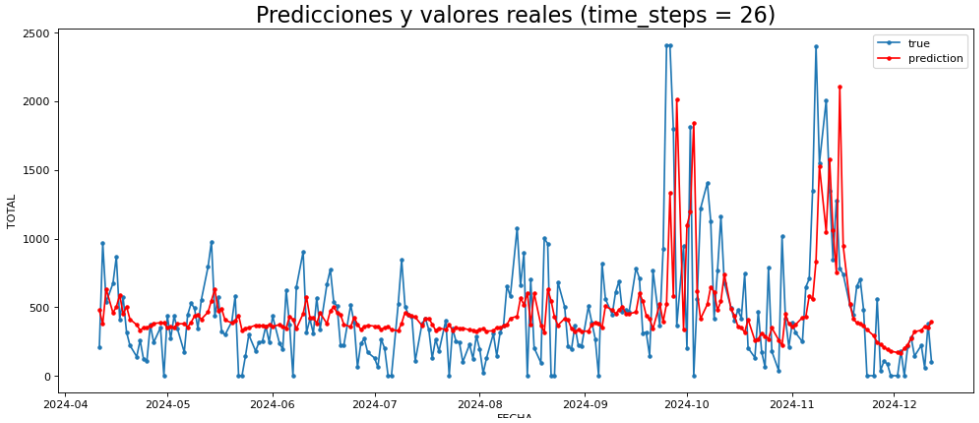


Figura 4-8: Predicción del LSTM con time_steps = 26

Fuente: Elaboración propia, 2025

El modelo con time_steps igual a 6, que se muestra en la figura 4-9, se comporta como un punto medio entre ambos extremos, mostrando una capacidad razonable tanto para seguir fluctuaciones pequeñas, de mejor manera que el modelo con time_steps igual a 26, pero no con tanta precisión como el modelo con time_steps igual a 1. Respecto a las grandes fluctuaciones, su comportamiento es muy similar al modelo con time_steps igual a 1.

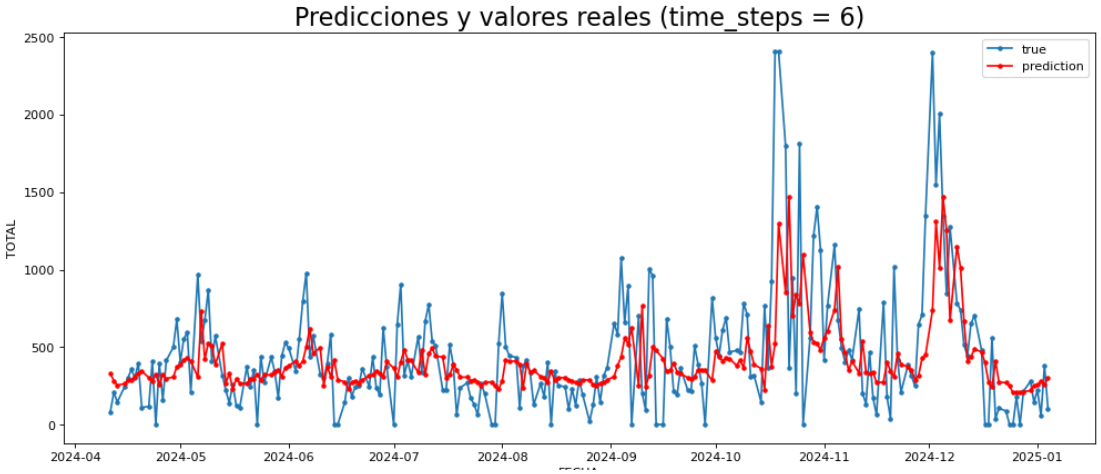


Figura 4-9: Predicción del LSTM con time_steps = 6

Fuente: Elaboración propia, 2025

De igual manera, las métricas estadísticas de evaluación respaldan estas observaciones. El modelo con time_steps igual a 1 alcanzó los mejores resultados entre las distintas configuraciones del LSTM, lo que indica que en promedio cometió los menores errores absolutos. A medida que se incrementa el tamaño de la secuencia temporal, tanto RMSE como MAE aumentan progresivamente. Esto sugiere que, si bien los modelos con time_steps mayores son útiles para capturar tendencias más amplias, su error promedio en la predicción también se incrementa, especialmente en los detalles de menor escala.

La predicción realizada por el modelo Prophet, como se muestra en la figura 4-10, presenta una representación que no se ajusta idóneamente a los datos reales, pero la tendencia general es coherente con su comportamiento. También se puede observar que este modelo tiende a subestimar los picos más pronunciados y a sobreestimar algunos valles, mostrando un desfase leve en la detección de cambios bruscos. Esta característica es esperada, ya que Prophet está diseñado principalmente para capturar tendencias y estacionalidades de forma robusta, sacrificando cierta precisión en eventos extremos o atípicos a favor de una interpretación más estable de la serie.

Al comparar los modelos, se observa que las redes LSTM muestran una mayor sensibilidad a las fluctuaciones de la serie temporal. Como se aprecia en las Figuras 4-7 y 4-8, sus predicciones tienden a seguir con mayor precisión la forma de la serie real, replicando tanto los picos como las caídas más pronunciadas. Esta capacidad para adaptarse a variaciones rápidas constituye una de las principales fortalezas de las redes neuronales recurrentes. En contraste, Prophet descompone la serie en componentes de tendencia, estacionalidad y *holidays*, como se mostró en la sección 4.3.2, y la combinación de estos elementos da lugar a su predicción final

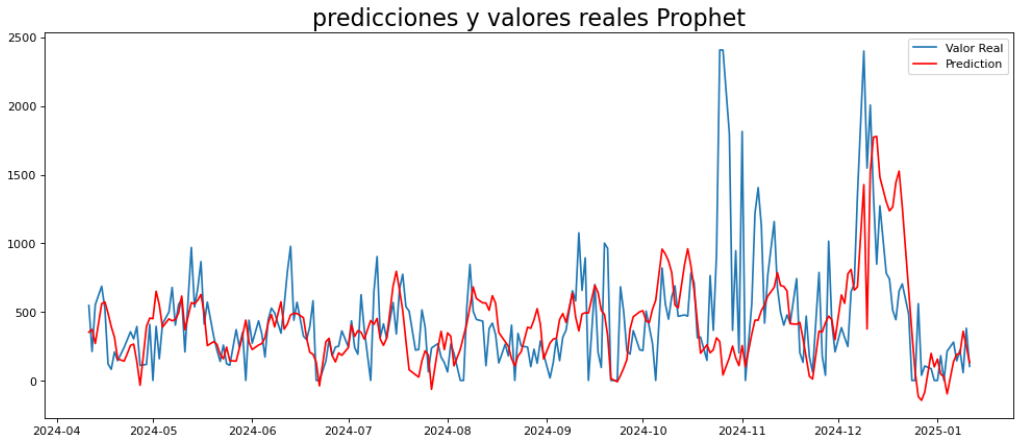


Figura 4-10: Predicción de Prophet

Fuente: Elaboración propia, 2025

4.5. Discusión de resultados

En esta sección se realiza una comparación entre los resultados obtenidos en este trabajo y los reportados en otro estudio relacionado que también emplea redes neuronales LSTM y Prophet. El objetivo es contextualizar el desempeño de los modelos, identificar similitudes y analizar las ventajas o limitaciones observadas en comparación con otras aproximaciones existentes.

4.5.1. Descripción del proyecto

El proyecto que se tomo es MODELO DE PREDICCIÓN DE DEMANDA DE PRODUCTOS PARA LA EMPRESA "TODO CALAMINAS", producido por Hébert Juan de Dios Delgadillo Fernández que se desarrolló para el diplomado de Estadística aplicada a la toma de decisiones, en su segunda versión, del postgrado de la facultad de ciencias y tecnología de la universidad Mayor de San Simón.

El proyecto se centró en desarrollar modelos estadísticos y de Machine Learning para predecir la demanda de productos de la empresa "Todo Calaminas", con el fin de optimizar la gestión de inventario y reducir pérdidas. Para ello, se aplicaron técnicas de limpieza y análisis de datos, y se implementaron los modelos ARIMA, Prophet y LSTM. Estos se entrenaron con más de 9 años de datos y se validaron con 5 meses. Las métricas utilizadas para evaluar el desempeño fueron MSE, RMSE, MAE, R^2 y MAPE. Se analizaron tres productos clave, y en todos los casos, el modelo de redes neuronales obtuvo los mejores resultados, con errores MAPE aceptables según el contexto.

La elección de este trabajo como punto de comparación para el presente estudio se basa en tres razones principales. En primer lugar, aunque el objetivo de predicción no es exactamente el mismo, los datos utilizados en el proyecto comparado presentan un comportamiento temporal similar al de los datos analizados en este trabajo. En segundo lugar, ambos estudios emplean redes neuronales LSTM y Prophet como modelo de predicción de series de tiempo, lo que permite establecer una base metodológica común. Finalmente, ambos trabajos utilizan la métrica de evaluación MAPE, lo que facilita una comparación directa del rendimiento predictivo entre ambos enfoques.

4.5.2. Datos

Este trabajo intenta predecir la demanda de varios productos que ofrece esta empresa, pero por razones de practicidad y simplicidad, se realizara la comparación solo con el producto más vendido, la plancha de calamina Zincalum #28. La figura 4-11 muestra las ventas históricas de este producto.

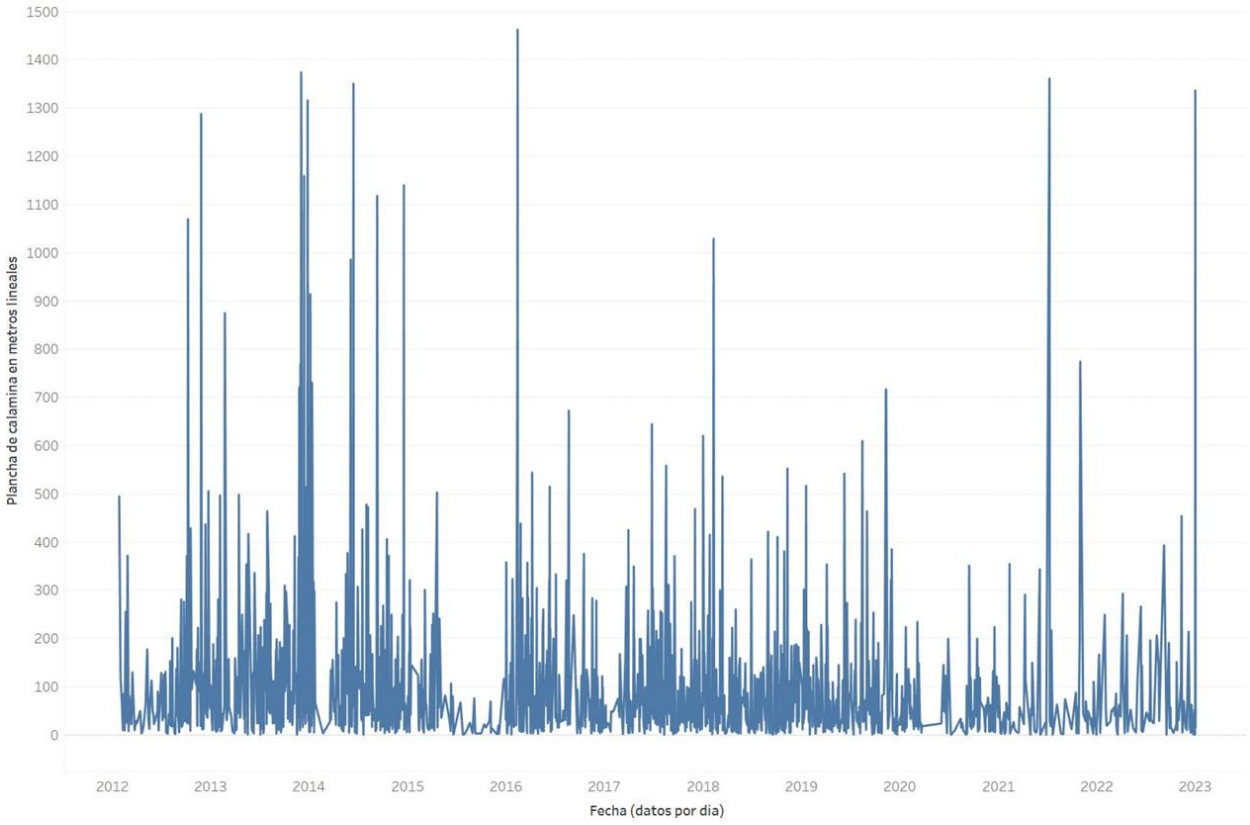


Figura 4-11: Ventas diarias del Zincalum #28

Fuente: Delgadillo, 2024

De acuerdo con el autor del proyecto, las ventas diarias del producto presentan una alta variabilidad y un comportamiento marcadamente aleatorio, lo que representa un desafío considerable para obtener predicciones precisas a nivel diario. Esta inestabilidad en los datos podría dificultar la captura de patrones consistentes por parte del modelo. Por esta razón, se consideró más adecuado agrupar las ventas en una frecuencia mensual, ya que esta transformación permite suavizar las fluctuaciones diarias y resaltar tendencias generales, facilitando así un análisis más estable y una predicción más fiable.

En la figura 4-12, se puede observar el comportamiento de los datos que resultaron de la limpieza y el agrupamiento de los datos diarios. En cuanto al comportamiento de la secuencia analizada, se puede apreciar a simple vista que se trata de una serie temporal estacionaria, sin una estacionalidad o patrón cíclico claramente definido. Esta característica es comparable con la serie de tiempo utilizada en el presente proyecto, la cual también muestra un comportamiento relativamente estable a lo largo del tiempo y carece de una estructura temporal marcada, como ciclos anuales o estacionales evidentes.

Figura 4-12: Ventas mensuales del Zincalum #28

Fuente: Delgadillo, 2024

4.5.3. Preparación y entrenamiento del LSTM

Es importante destacar que el proyecto con el que se realiza la comparación siguió un enfoque metodológico muy similar al del presente trabajo, tanto en el preprocesamiento como en la construcción del modelo. En primer lugar, ambos proyectos aplicaron la normalización Min-Max a los datos de entrada, escalando los valores al rango [0, 1]. En cuanto a la arquitectura del modelo LSTM, también se observa una alta similitud, el modelo comparado está compuesto por cuatro capas LSTM secuenciales, cada una con 250 unidades, intercaladas con capas Dropout. Respecto al entrenamiento, ambos modelos utilizaron batch de tamaño 32, durante 100 épocas, sin validación cruzada y con shuffle=False para preservar la secuencia temporal de los datos. Esta coincidencia en la preparación de los datos y en la arquitectura de los modelos permite establecer una comparación técnica idónea entre los resultados obtenidos en ambos trabajos.

4.5.4. Preparación y entrenamiento de Prophet

En comparación con el enfoque utilizado en el presente trabajo, donde se configuró el modelo Prophet incorporando información adicional como los *holidays* y se ajustaron ciertos parámetros para mejorar el ajuste del modelo, el estudio con el que se contrasta aplicó Prophet utilizando únicamente su configuración por defecto. En dicho trabajo, se mantuvieron los parámetros preestablecidos del algoritmo, asumiendo una estacionalidad anual de 12 meses y un agrupamiento mensual, sin realizar ajustes específicos durante el proceso de creación o entrenamiento del modelo. Si bien esta elección puede ser válida cuando los datos se alinean naturalmente con los supuestos de Prophet, limita la capacidad del modelo para adaptarse a particularidades del dominio o de la serie temporal en cuestión.

4.5.5. Comparación de resultados

En la tabla 4-2, se muestra los resultados obtenidos por los modelos que propuso el autor.

Indicador	ARIMA	SARIMA	LSTM	Prophet
MSE	315369.37	801827.50	79561.98	205530.07
MAE	440.58	590.17	191.01	423.04
RMSE	561.58	895.45	282.07	453.35
MAPE	57.56	65.82	18.31	74.18
R2	-0.12	-1.84	0.72	0.27

Tabla 4-2: Resultados de las predicciones de los distintos modelos
Fuente: Delgadillo, 2024

En base a estos resultados, el autor concluye que el mejor modelo para la predicción fue el LSTM, que se puede ver claramente que tiene los mejores resultados en las métricas, con clara diferencia en comparación a los demás modelos.

Al observar los gráficos de predicción, es evidente una diferencia notable en el desempeño de los modelos LSTM y Prophet. En la figura 4-13, el modelo LSTM muestra una capacidad mucho mayor para seguir el comportamiento real de la serie, replicando con precisión tanto la tendencia general como las fluctuaciones que se presentan a lo largo del tiempo.

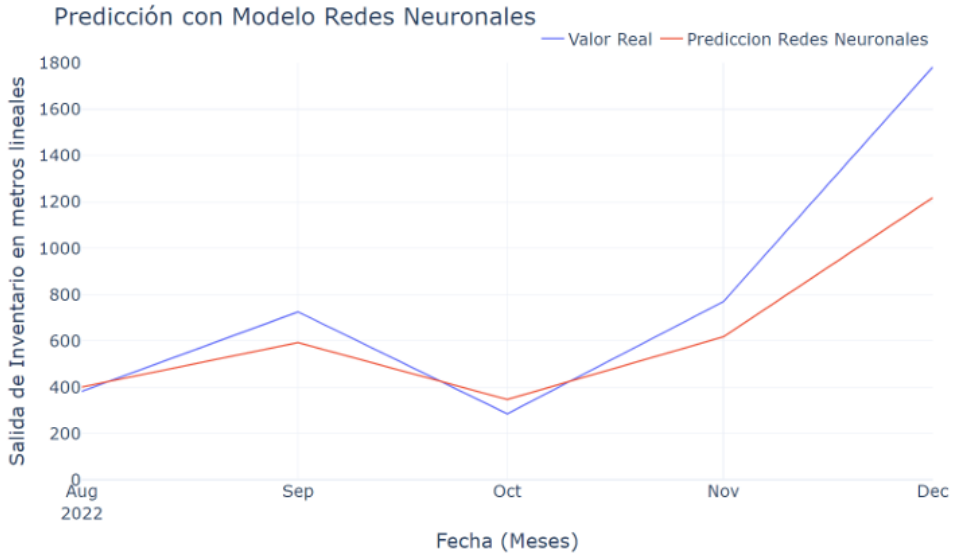


Figura 4-13: Comparación de los valores predichos contra los valores reales con LSTM
Fuente: Delgadillo, 2024

En cambio, la predicción generada por Prophet que se observa en la figura 4-14, se aleja considerablemente de los valores reales. Esto resalta la ventaja de las redes neuronales recurrentes como LSTM cuando se requiere un seguimiento más exacto de la evolución temporal y de las variaciones locales de los datos. Es posible que el modelo Prophet hubiera logrado un mejor ajuste si se hubieran modificado los parámetros predeterminados por valores más adecuados a las características específicas de la serie temporal.

Figura 4-14: Comparación de los valores predichos contra los valores reales con Prophet
Fuente: Delgadillo, 2024

Sin embargo, según los resultados del MAPE del LSTM del trabajo comparado (18.31%) obtuvo un mejor resultado que el de este trabajo con la mejor configuración, que fue con time_steps igual a 1 (29.92%). Existen factores clave que explican esta diferencia. En primer lugar, el proyecto comparado contó con una mayor cantidad de datos históricos, iniciando en 2012, mientras que los datos del presente trabajo comienzan en 2021. Esto significa que el modelo de referencia tuvo más datos para aprender patrones complejos y generalizar mejor. En segundo lugar, el horizonte de predicción también fue diferente, el proyecto comparado realizó predicciones para cinco meses (aproximadamente 150 días), con datos agregados mensualmente, mientras que este trabajo realizó predicciones diarias para un total de 237 días. Esta diferencia tanto en granularidad como en duración del período predicho implica una mayor complejidad y variabilidad, lo cual naturalmente afecta el valor del MAPE. Además, otro posible motivo, es que la serie que se intentó predecir presentaba una menor cantidad de fluctuaciones abruptas, lo que facilita obtener una mejor precisión relativa. Por tanto, aunque el error porcentual es mayor en este trabajo, el resultado sigue siendo competitivo y adecuado para el contexto específico en el que fue desarrollado.

Por otro lado, al comparar el rendimiento del modelo Prophet, se observa una diferencia favorable hacia el presente trabajo. En el proyecto comparado, las predicciones generadas por Prophet no lograron ajustarse de manera adecuada a la serie temporal real, presentando resultados alejados de los datos reales. En cambio, el modelo Prophet desarrollado en el presente trabajo obtuvo métricas considerablemente mejores, logrando un MAPE de 16.34%, lo cual indica una mayor precisión relativa en sus predicciones. Incluso, al compararlo con los modelos LSTM, Prophet presentó un mejor desempeño en términos de MAPE, superando tanto al del presente trabajo como al del trabajo comparado. Estos resultados reflejan que, al configurar correctamente los parámetros y considerar componentes relevantes como los *holidays* y estacionalidades adicionales, Prophet puede ser una herramienta eficaz incluso frente a modelos más complejos como LSTM, especialmente cuando se busca interpretar de forma clara los patrones subyacentes de la serie temporal.

5. Conclusiones

El presente trabajo tuvo como objetivo desarrollar un modelo de predicción basado en series de tiempo, utilizando técnicas de aprendizaje automático, con el fin de estimar la recaudación de ingresos por cobro de facturas en una EPSA. Este enfoque buscó proporcionar una herramienta útil para la planificación financiera y la toma de decisiones estratégicas, contribuyendo así a una gestión más eficaz del recurso y a la sostenibilidad económica de la entidad.

Uno de los aspectos más importantes en este trabajo fue el tratamiento riguroso de los datos, el cual implicó la eliminación de una cantidad considerable de registros y columnas que presentaban inconsistencias, nulos o valores atípicos, procesos que llevaron de tener 19.889 registros a 16.500 registros. Sin embargo, esta depuración no afectó negativamente la validez del análisis ni la efectividad del modelo, ya que los datos eliminados no aportaban valor relevante al entrenamiento o no eran representativos del comportamiento que se buscaba modelar. Esto refuerza la idea de que una reducción significativa del conjunto de datos puede ser válida y hasta beneficiosa, siempre que los datos restantes conserven la estructura, la calidad y la representatividad necesarias para el objetivo del proyecto.

Posteriormente, se procedió a la implementación de los modelos de predicción. Entre las distintas configuraciones evaluadas del modelo LSTM, la que utilizó un valor de time_steps igual a 1 fue la que obtuvo los mejores resultados en términos de error absoluto y cuadrático, alcanzando los valores más bajos de RMSE (342.58) y MAE (231.96). Esto indica que, en cuanto a precisión numérica pura, esta configuración logró realizar predicciones más cercanas a los valores reales. No obstante, al compararlo con el modelo Prophet, se observa que este último obtuvo un MAPE considerablemente menor (17.82%), lo que sugiere que, en proporción a los valores reales, sus predicciones fueron más estables y relativamente más precisas. Esta diferencia refleja dos enfoques distintos: mientras LSTM con time_steps igual a 1 es capaz de ajustarse rápidamente a las variaciones locales de los datos, Prophet sobresale al capturar patrones de estacionalidad y tendencia más amplios, especialmente cuando se incorporan adecuadamente componentes como los *holidays* y configuraciones específicas de estacionalidad. No obstante, debido a que el objetivo de este trabajo es predecir ingresos, un fenómeno que suele presentar variaciones bruscas por factores como la demanda, el comportamiento del cliente o eventos puntuales, se consideró más adecuado priorizar un modelo que pueda capturar esas fluctuaciones. En este sentido, el LSTM demostró una mayor capacidad para adaptarse a los cambios abruptos en la serie, lo que motivó su elección como modelo final.

Antes de seleccionar a las redes neuronales LSTM y a Prophet como los modelos que se presentan en este trabajo, se exploró inicialmente el uso del modelo ARIMA. Sin embargo, los resultados de este modelo fueron significativamente alejados de los valores reales, por lo que fue descartado. Dada la marcada diferencia en el desempeño, no se consideró pertinente incluirlo en la comparación con el modelo LSTM y Prophet.

Adicionalmente, se implementó un proceso de validación que incluyó la comparación con otro proyecto de características similares, el cual utilizó también una arquitectura LSTM para predecir las ventas de un producto. Aunque el modelo comparado obtuvo un MAPE menor (18.31%), es importante considerar que contaba con un mayor volumen de datos históricos, registrados desde el año 2012, mientras que el presente trabajo dispuso de datos únicamente desde 2021. Asimismo, la predicción en el proyecto de referencia abarcaba cinco meses (aproximadamente 150 días) con datos agregados mensualmente, mientras que en este trabajo se realizaron predicciones diarias durante 237 días, lo que representa un mayor grado de detalle y complejidad para el modelo. A esto se suma que la serie de tiempo del proyecto comparado presentaba una menor variabilidad, lo que naturalmente favorece la obtención de un MAPE más bajo. En cuanto a la comparación entre los modelos Prophet, el presente trabajo logró mejores resultados gracias a un ajuste más cuidadoso de los parámetros del modelo.

6. Recomendaciones

A partir del desarrollo de este proyecto, se identificaron diversas oportunidades de mejora tanto en el manejo de los datos como en el diseño e implementación de modelos predictivos. Uno de los aspectos más relevantes para fortalecer la calidad de futuras predicciones es la disponibilidad de datos históricos amplios y consistentes. Por ello, se recomienda mantener y mejorar los procesos de recolección y almacenamiento de datos, asegurando que se realicen de forma estructurada, periódica y con el menor nivel de pérdida o inconsistencia posible. Cuantos más datos se tengan disponibles, especialmente si cubren múltiples ciclos económicos, variaciones estacionales o cambios en el comportamiento de los usuarios, mayor será la capacidad del modelo para generar predicciones útiles y confiables en contextos reales.

En el análisis de las diferentes configuraciones del modelo LSTM, se observó que la variante con time_steps igual a 1 presentó una destacada capacidad para capturar las fluctuaciones locales de la serie temporal, logrando un seguimiento más cercano a los cambios diarios en los ingresos. Esta sensibilidad resulta especialmente valiosa en el contexto de una EPSA, donde los ingresos pueden verse afectados por factores puntuales como campañas de cobranza, cortes, reconexiones o eventos externos que alteran temporalmente el comportamiento de pago de los usuarios. Aunque configuraciones con mayor número de pasos como time_steps igual a 26 mostraron un mejor seguimiento de cambios más amplios, su menor precisión en las variaciones sutiles las hace menos adecuadas para este tipo de fenómeno. Por tanto, se recomienda la elección del modelo con time_steps igual a 1 como el más apropiado, ya que prioriza la detección de fluctuaciones, aspecto crucial cuando se trata de predecir ingresos diarios con el fin de optimizar la planificación financiera y operativa. Esta capacidad de respuesta ante cambios a corto plazo supera el enfoque más tendencial de modelos como Prophet, y lo convierte en una herramienta más eficaz para este tipo de aplicación.

Finalmente, se sugiere que para futuros trabajos se considere ampliar la cantidad de datos históricos, experimentar con otras arquitecturas como GRU o modelos híbridos, y explorar predicciones a distintas escalas temporales que puedan facilitar la comparación entre modelos. También es recomendable automatizar el tratamiento de valores extremos y vacíos, así como incorporar mecanismos de validación cruzada más robustos, que permitan una mejor generalización de los resultados obtenidos.

Bibliografía

Alpaydin, E. (2020). *Introduction to Machine Learning* (4 ed.). London, England: MIT Press.

Barrera Cordero, J. (2009). La guerra del agua en Cochabamba: un caso de palabras que hablan mal. *Biblat*, 91-100. Obtenido de https://biblat.unam.mx/es/revista/investigacion-ambiental-ciencia-y-politica-publica/articulo/la-guerra-del-agua-en-cochabamba-un-caso-de-palabras-que-hablan-mal

Brownlee, J. (2019). *Long Short-Term Memory Networks with Python.* Machine Learning Mastery.

Cabañas, R. (4 de noviembre de 2021). *Redes Neuronales Artificiales*. Obtenido de Ab Datum: https://abdatum.com/tecnologia/redes-neuronales-artificiales

Cabrera, J. E. (2018). Fragmentación urbana por medio de redes de agua: el caso de Cochabamba, Bolivia. *SciELO*. Obtenido de http://dx.doi.org/10.12804/revistas.urosario.edu.co/territorios/a.6313

Carballo, Y. (enero de 2024). Audiencia de Rendición Pública de Cuentas Final 2023 del Ministerio de Medio Ambiente y Agua.

Contreras Morales, E. F., Ferreira Correa, F. M., & Mauricio, V. (2017). *DISEÑO DE UN MODELO PREDICTIVO DE FUGA DE CLIENTES UTILIZANDO ÁRBOLES DE DECISIÓN.* Santiago de Chile: Universidad de Valparaiso. doi:https://revistas.ubiobio.cl/index.php/RI/article/view/3055/3075

Cryer, J., & Chan, K.-S. (2008). *Time Series Analysis With Applications in R.* New York: Springer.

Decreto Supremo Nº 0071. (9 de abril de 2009). *Crear las Autoridades de Fiscalización y Control Social*.

Delgadillo Fernandez, H. J. (2024). *MODELO DE PREDICCION DE DEMANDA DE PRODUCTOS PARA LA EMPRESA "TODO CALAMINAS".* Cochabamba: Diplomado de la facultad de Ciencias y Tecnologia de la Universidad Mayor de San Simon. doi:http://ddigital.umss.edu/handle/123456789/43504

Graupe, D. (2013). *Principles of Artificial Neural Networks.* Singapore: World Scientific Publishing Co.

IBM. (s.f.). *Conceptos básicos de ayuda de CRISP-DM*. Obtenido de IBM SPSS Modeler: https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

Instituto Nacional de Estadistica. (2016). Informe anual 2016. La Paz.

Kleiber, C., & Zeileis, A. (2008). *Applied Econometrics with R.* Nueva York: Springer.

Korstanje, J. (2021). *Advanced Forecasting with Python: With State-of-the-Art-Models Including LSTMs, Facebook's Prophet, and Amazon's DeepAR.* Maisons Alfort, France: Apress.

Kruse, t. (2005). *La Guerra del Agua en Cochabamba, Bolivia: terrenos complejos, convergencias nuevas.* Buenos Aires, Argentina: CLACSO, Consejo Latinoamericano de Ciencias Sociales. Obtenido de https://core.ac.uk/download/pdf/35175037.pdf

Ley N° 2066. (11 de abril de 2000). *Ley de Prestación y utilización de Servicios de Agua Potable y Alcantarillado Sanitario*.

Liebowitz, J. (2020). *Data Analytics and AI.* Nueva York: CRC Press.

Maki, A. (2010). *Introduction to Etimatng Economic Models.* Nueva York: Routledge.

Marston, A. (2014). The Scale of Informality: Community-Run Water Systems in Peri-Urban Cochabamba, Bolivia. *Water Alternatives 7*, 72-88. Obtenido de https://www.water-alternatives.org/index.php/allabs/234-a7-1-5/file

Martinez Soto, J. G. (abril de 2025). Entrevista al presidente de la OTB Thika Khatu. (J. E. Martinez Barriga, Entrevistador)

Mercado Guzman, A. R., Cosio Grageda, C. X., & Copa Mitma, M. (2020). Eficiencia vinculada a la operacion y mantenimiento de pequeñas plantas de tratamiento de aguas residuales domesticas en Cochabamba, Bolivia. *Research Gate*.

Nieto, N. (2011). La gestión del agua: tensiones globales y latinoamericanas. *SciELO*. Obtenido de https://www.scielo.org.mx/scielo.php?pid=S0188-77422011000200007&script=sci_arttext

Observatorio Cochabamba Nos Une. (2015). *Informe 2015.* Cochabamba.

Peixeiro, M. (2022). *Time Series Forecasting in Python.* Shelter Island, New York: Manning.

Prophet official page. (s.f.). Obtenido de Facebook Open Source: https://facebook.github.io/prophet/

Riabani Mercado, F., Garcia Fernandez, W., & Herrera Acebey, J. A. (2016). Sistema de inteligencia artificial para la predicción temprana de heladas meteorológicas. *Acta Nova*.

Rios, G. (2008). *Series de Tiempo.* Chile: Universidad de Chile. Obtenido de https://gc.scalahed.com/recursos/files/r161r/w24113w/Semana%2011/Series_de_Tiempo.pdf

Salomon, D. (2007). *Data Compression.* Londres: Springer.

sambsv. (27 de marzo de 2023). *Implementing Neural Network LSTM*. Obtenido de Medium: https://medium.com/@sambsv/implementing-neural-network-lestm-b7ec03ba2b5

Sanjines Tudela, G. N. (2011). Análisis y pronóstico de la demanda de potencia eléctrica en Bolivia: una aplicación de redes neuronales. *Universidad Católica Boliviana "San Pablo"*.

Siegel, E. (2013). *Predictive Analiytics.* Hoboken, New Jersey: Jhon Wily & Sons, Inc.

Sivanandam, P. (2009). *Introduction to Artificial Neural Networks.* Nueva Delhi: Vikas Publishing House PVT.

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep Learning with PyTorch.* Shelter Island, NY: Manning Publications Co.

Taylor, S., & Letham, B. (2017). Forecasting at scale. *PeerJ Preprints*, 25. doi:https://doi.org/10.7287/peerj.preprints.3190v2

Toledo Medrano, R., & Amurrio Derpic, D. (2006). Evaluación de la calidad de las aguas del río Rocha en la jurisdicción de SEMAPA en la provincia Cercado de CochabambaBolivia. *Sistema OJS UCBSP*.

Udo Moffat, I., & Alphonus Akpan, E. (2019). White Noise Analysis: A Measure of Time Series Model Adequacy. *Applied Mathematics*. doi:https://doi.org/10.4236/am.2019.1011069

Unidad de Análisis de Políticas Sociales y Económicas (UDAPE), Comité Interinstitucional de las Metas de Desarrollo del Milenio (CIMDM). (2013). *Séptimo informe de progreso de los Objetivos de Desarrollo del Milenio en Bolivia.* La Paz.

Wei, W. (2005). *Time Series Analysis.* Pearson.

Zapata, S. (2015). Potencial de explotación de agua subterránea en Cliza - Cochabamba. *Facultad de Ingenieria civil, Universidad Mayor, Real Y Pontificia de San Francisco Xavier de Chuquisaca*.

Anexos

Anexo A. Base de datos

SOCIO	N_SOC	DIRECCION	NRO	FONO	RUC_CLI	FECHA	COD	CONCEPTO	IMPORTE
109	M-109	Thika Khatu				20210707		Cancelado	176
276	T-276	Thika Khatu		72258591		20221209		Cancelado	91
252	U-252	Thika Khatu				20210831		Cancelado	18
289	U-289	thika Khatu				20210503		Cancelado	208
4	A- 4	Thika Khatu		4313182		20210420		Cancelado	30
5	B-5	Thika Khatu		4313010		20210415		Cancelado	18
6	B-6	Thika Khatu		4313190		20210407		Cancelado	18
12	B-12	Thika Khatu				20210415		Cancelado	107
16	C-16	Thika Khatu				20210417		Cancelado	18
261	B-261	Thika Khatu				20210705		Cancelado	18
261	B-261	Thika Khatu				20210803		Cancelado	18
27	C-27	Thika Khatu			3324389 Cbba.	20210512		Cancelado	18
26	E-26	Thika Khatu				20210514		Cancelado	18
247	E-247	Thika Khatu		77449531		20210510		Cancelado	18
29	E-29	Thika Khatu		4313571		20210414		Cancelado	25
253	E-253	Thika Khatu				20210902		Cancelado	47
211	E-211	Thika Khatu				20210409		Cancelado	18
215	F-215	Thika Khatu				20210427		Cancelado	18
221	F-221	Thika Khatu				20210419		Cancelado	100

Nota: archivo completo en el CD: lecturas.xlsx

Anexo B. Definición de las variables de la base de datos

Nombre	Descripción	Nombre	Descripción
SOCIO	código de socio	MES_COBRO	mes de la factura consumida
N_SOCIO	código zona - código socio	LEC_ANTES	lectura del mes anterior (metros cúbicos)
DIRECCION	dirección del inmueble	LECT_HOY	lectura actual (metros cúbicos)
NRO	número de casa del socio (si aplica)	CREDITO	descartar
FONO	número de teléfono del socio	CONSUMO	diferencia entre LEC_HOY y LEC_ANTES (consumo real en m3)
RUC_CLI	nit del socio	ATRASO	cantidad de días de retraso después del plazo de vencimiento para pagar
FECHA	fecha de pago	TOTAL	total del importe a pagar
COD	descartar	LITE	total literal
CONCEPTO	factura cancelada o no	EMISION	descartar
IMPORTE	consumo de agua en Bs	MEDIDOR	número de medidor
VENCE	fecha de vencimiento de la factura	VENCE	fecha de vencimiento de la factura
ALUMBRADO	descartar	OTROS	descartar
BASURA	descartar	BENEMERITO	sí es benemérito
CERTIF	descartar	ZONA	zona de la OTB en la que vive el socio
TITULO	descartar	DESCBENEME	porcentaje de descuento por benemérito
INGRESO	fecha de ingreso del socio	ALCANTARI	descartar
BANDERA	descartar	ORDENES	descartar
VALOR	derecho de inscripción (dolares)	HEXA	descartar
CARNET	Nro. de carnet del socio	NUMFAC	número de factura
FLEC	fecha de lectura	FACTURA	número de factura
MINIMO	descartar	COPIA	descartar
CORTE	fue o no cortado el suministro de agua	REUNION	multas
MULTA	descartar	TRABAJOS	multas
RECONEX	costo de reconexión (bolivianos)	PASIBLE	fecha de posible corte de agua
FALTAS	costo de comprobante	LEIANTES	lectura anterior
BANDI	descartar	GESTION	gestión en la que se emitió la factura
CATEGORIA	tipo de consumidor	CPU	descartar

Anexo C. Tratamiento de datos nulos

```
df.isnull().sum()
```

```
print('valores nulos: ', df.isnull().sum().sum())
```

```
df.columns
```

```
df = df.drop(columns=['SOCIO', 'N_SOCIO', 'DIRECCION', 'NRO', 'FONO',
       'RUC_CLI', 'COD', 'CONCEPTO', 'CREDITO', 'LITE', 'EMISION', 'MEDIDOR',
        'ALUMBRADO', 'BASURA', 'CERTIF', 'TITULO', 'BANDERA', 'CARNET', 'MINIMO', 'CORTE', 'MULTAS',
        'RECONEX', 'BANDI', 'OTROS', 'ALCANTARI', 'ORDENES', 'HEXA',
       'NUMFAC', 'FACTURA', 'COPIA', 'REUNION', 'TRABAJOS', 'CPU'])
```

```
df.tail(30)
```

```
df.shape
```

```
df.isnull().sum()
```

```
df.columns
```

```
df = df.dropna(subset=['FECHA', 'IMPORTE', 'CATEGORIA', 'MES_COBRO', 'LEC_ANTES',
       'LECT_HOY', 'CONSUMO', 'ATRASO', 'TOTAL', 'VENCE', 'INGRESO', 'VALOR',
       'FLEC', 'FALTAS', 'BENEMERITO', 'ZONA', 'DESCBENEME', 'PASIBLE',
       'LEIANTES', 'GESTION'])
df.isnull().sum()
```

```
df.shape
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo D. Tratamiento de fechas

```
df.dtypes
```

```
fechas = df['FECHA'].value_counts().to_frame()
inconsistencias = fechas[fechas.index.str.len() < 8]
inconsistencias
```

```
filtered_df = df[(df['FECHA'].str.len() == 8) | (df['FECHA'].isna())].copy()
filtered_df.shape
```

```
filtered_df['FECHA'] = filtered_df['FECHA'].astype(str)
filtered_df['AÑO'] = filtered_df['FECHA'].str[:4]
filtered_df['MES'] = filtered_df['FECHA'].str[4:6]
filtered_df['DIA'] = filtered_df['FECHA'].str[6:8]
filtered_df['AÑO'] = filtered_df['AÑO'].astype('Int64')
filtered_df['MES'] = filtered_df['MES'].astype('Int64')
filtered_df['DIA'] = filtered_df['DIA'].astype('Int64')
```

```
anios_erroneas = filtered_df[(filtered_df['AÑO'] > 2026) | (filtered_df['AÑO'] < 1960)]
anios_erroneas
```

```
filtered_df = filtered_df[(filtered_df['AÑO'] < 2026) & (filtered_df['AÑO'] > 1960)].copy()
filtered_df.shape
```

```
meses_erroneos = filtered_df[(filtered_df['MES'] > 12) | (filtered_df['MES'] < 1)]
meses_erroneos
```

```
filtered_df = filtered_df[(filtered_df['MES'] <= 12) & (filtered_df['MES'] >= 1)].copy()
filtered_df.shape
```

```
dias_erroneos = filtered_df[(filtered_df['DIA'] > 31) | (filtered_df['DIA'] < 1)]
dias_erroneos
```

```
filtered_df = filtered_df.drop(columns=['AÑO', 'MES', 'DIA'])
filtered_df["FECHA"] = pd.to_datetime(filtered_df["FECHA"], format='%Y%m%d')
filtered_df
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo E. Tratamiento de valores atípicos

```
filtered_df[['TOTAL']].describe()
```

```
q1 = filtered_df['TOTAL'].quantile(q=0.25)
q3 = filtered_df['TOTAL'].quantile(q=0.75)
iqr = q3 - q1
print('cuartil 1: ', q1)
print('cuartil 3: ', q3)
print('rango intercuartilico: ', iqr)
```

```
print('limite inferior: ', q1 - 1.5 * iqr)
print('limite superior: ', q3 + 1.5 * iqr)
```

```
fuera = filtered_df[(filtered_df['TOTAL'] < (q1 - 1.5 * iqr)) | (filtered_df['TOTAL'] > (q3 + 1.5 * iqr))]
fuera
```

```
negativos = filtered_df[filtered_df['TOTAL']<0]
negativos
```

```
filtered_df = filtered_df[(filtered_df['TOTAL'] > 0) & (filtered_df['TOTAL'] < (q3 + 1.5 * iqr))]
filtered_df
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo F. Agrupamiento de los datos

```
grouped_df = total_df.groupby(['FECHA']).agg({'TOTAL': np.sum}).sort_values(['FECHA'], ascending=True)
grouped_df
```

```
custom_freq = CustomBusinessDay(weekmask='Mon Tue Wed Thu Fri Sat')
grouped_df = grouped_df.asfreq(custom_freq)
```

```
grouped_df.isna().sum()
```

```
grouped_df = grouped_df.fillna(1)
grouped_df
```

```
grouped_df["TOTAL"].plot(figsize=(20,8), title = "Total pagos diario")
plt.show()
```

```
anio = grouped_df.loc['2023-01-01':'2023-12-31']
anio["TOTAL"].plot(figsize=(20,5), title = "Total pagos diario")
plt.show()
```

```
anio_22 = grouped_df.loc['2022-01-01':'2022-12-31']
anio_22["TOTAL"].plot(figsize=(20,5), title = "Total pagos diario")
plt.show()
```

```
anio_24 = grouped_df.loc['2024-01-01':'2024-12-31']
anio_24["TOTAL"].plot(figsize=(20,5), title = "Total pagos diario")
plt.show()
```

```
mes = grouped_df.loc['2023-12-01':'2023-12-31']
mes["TOTAL"].plot(figsize=(20,5), title = "Total pagos diario")
plt.show()
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo G. Tratamiento de valores atípicos en la serie de tiempo

```
q1_serie = grouped_df['TOTAL'].quantile(q=0.25)
q3_serie = grouped_df['TOTAL'].quantile(q=0.75)
iqr_serie = q3_serie - q1_serie
print('q1: ', q1_serie)
print('q3: ', q3_serie)
print('iqr: ', iqr_serie)
plt.ylim(0, 3000)
plt.boxplot(grouped_df['TOTAL'])
```

```
lower_bound = q1_serie - 1.5 * iqr_serie
upper_bound = q3_serie + 1.5 * iqr_serie
print('limite inferior: ', lower_bound)
print('limite superior: ', upper_bound)
```

```
fig = plt.figure(figsize=(14, 8))
gs = gridspec.GridSpec(1, 2, width_ratios=[7, 1])
ax1 = plt.subplot(gs[0])
ax1.plot(grouped_df)
ax2 = plt.subplot(gs[1])
ax2.boxplot(grouped_df['TOTAL'])
ax1.axhline(upper_bound, color='red')
ax2.axhline(upper_bound, color='red',  label='outliers')
ax2.legend()
plt.tight_layout()
plt.show()
```

```
fuera = grouped_df[(grouped_df['TOTAL'] > upper_bound)]
fuera.count()
```

Winsorization

```
q99 = grouped_df['TOTAL'].quantile(q=0.99)
q99
```

```
fuera = grouped_df[(grouped_df['TOTAL'] > q99)]
fuera.count()
```

```
grouped_df.loc[grouped_df['TOTAL'] > q99, 'TOTAL'] = q99
grouped_df["TOTAL"].plot(figsize=(20,10), title = "Total pagos diario")
plt.show()
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo H. División de datos y análisis exploratorio

Division de los datos

```
size = int(len(grouped_df)*0.8)
train_df = grouped_df.iloc[:size]
test_df = grouped_df.iloc[size:]
```

```
train_df
```

```
test_df
```

```
test_df["TOTAL"].plot(figsize=(20,5), title = "Total pagos diario (datos validacion)")
plt.show()
```

Estacionariedad

```
sts.adfuller(grouped_df.TOTAL)
```

Estacionalidad

```
aditivo = seasonal_decompose(grouped_df.TOTAL, model="additive", period=26)
aditivo.plot()
plt.show()
```

Autocorrelacion

```
sgt.plot_acf(grouped_df.TOTAL, lags = 40, zero = False)
plt.ylim(-0.2, 0.5)
plt.show()
```

```
sgt.plot_pacf(grouped_df.TOTAL, lags = 40, zero = False, method = ('ols'))
plt.ylim(-0.2, 0.5)
plt.show()
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo I. Implementación del LSTM

```
 !pip install tensorflow
```

```
!pip install keras
```

```
train_max = train_df.max()
train_min = train_df.min()
train_set_scaled = (train_df - train_min)/(train_max - train_min)
test_set_scaled = (test_df - train_min)/(train_max - train_min)
test_set_scaled
```

```
def create_dataset(X, y, time_steps=1):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        v = X.iloc[i:(i + time_steps)].values
        Xs.append(v)
        ys.append(y.iloc[i + time_steps])
    return np.array(Xs), np.array(ys)
```

```
time_steps = 1

X_train_1, y_train_1 = create_dataset(train_set_scaled, train_set_scaled.TOTAL , time_steps)
X_test_1,  y_test_1  = create_dataset(test_set_scaled,  test_set_scaled.TOTAL, time_steps)
```

```
from keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dropout, Dense, Input

def lstm_architecture(X_data,rate_dropout):
    model = Sequential()

    model.add(Input(shape=(X_data.shape[1], X_data.shape[2])))

    model.add(LSTM(units = 250, return_sequences = True))
    model.add(Dropout(rate=rate_dropout))

    model.add(LSTM(units = 250, return_sequences = True))
    model.add(Dropout(rate=rate_dropout))

    model.add(LSTM(units = 250, return_sequences = True))
    model.add(Dropout(rate=rate_dropout))

    model.add(LSTM(units = 250, return_sequences = False))
    model.add(Dropout(rate=rate_dropout))

    model.add(Dense(units = 1))

    model.summary()

    return model
```

```
import datetime
print('Iniciando a las: ', datetime.datetime.now())
print("...")

model_1 = lstm_architecture(X_data = X_train_1, rate_dropout = 0.2)
model_1.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo J. Implementación de Prophet

```
!pip install prophet
```

```
from prophet import Prophet
```

```
df_prophet = train_df.copy()
```

```
df_prophet = df_prophet.reset_index()
df_prophet = df_prophet.rename(columns={'FECHA':'ds', 'TOTAL':'y'})
df_prophet.head()
```

```
test_prophet = test_df.copy()
```

```
test_prophet = test_prophet.reset_index()
test_prophet = test_prophet.rename(columns={'FECHA':'ds', 'TOTAL':'y'})
test_prophet.head()
```

```
fin_de_año = pd.DataFrame({
    'holiday': 'fin de año',
    'ds': pd.to_datetime(['2021-12-05', '2022-12-05', '2023-12-05', '2024-12-05']),
    'lower_window': -5,
    'upper_window': 5
})
```

```
prophet = Prophet(holidays=fin_de_año, changepoint_range=0.95, yearly_seasonality=50)
prophet.add_seasonality(name='monthly', period=30.5, fourier_order=15)
prophet.fit(df_prophet)
```

```
prophet_pred = prophet.predict(test_prophet)
prophet_pred
```

```
plt.figure(num=None, figsize=(15, 6), dpi=80, facecolor='w', edgecolor='k')
plt.plot(prophet_pred['ds'], test_prophet['y'], label="Valor Real")
plt.plot(prophet_pred['ds'], prophet_pred['yhat'], 'r', label="Prediction")
plt.xlabel('Fecha')
plt.title('predicciones y valores reales Prophet', size=20)
plt.legend()
plt.show()
```

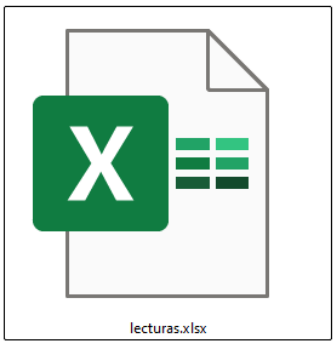
```
mse = mean_squared_error(test_prophet['y'], prophet_pred['yhat'])
rmse = np.sqrt(mse)
print(f'RMSE: ',rmse)

mape = mean_absolute_percentage_error(test_prophet['y'], prophet_pred['yhat'])
print(f'MAPE: ', mape)

mae = mean_absolute_error(test_prophet['y'], prophet_pred['yhat'])
print(f'MAE: ',mae)
```

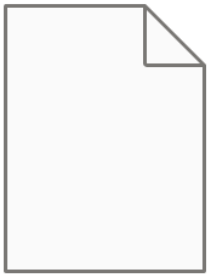
```
prophet.plot_components(prophet_pred)
```

Nota: El código completo de esta sección se encuentra en el archivo: prediccion de ingresos.ipynb

Anexo PRINCIPAL: CD

El CD almacena tres archivos:





- **MODELO DE PREDICCION DE INGRESOS POR LA COBRANZA DE FACTURAS DE LA EPSA “THIKA KHATU”.docx:** hace referencia a este proyecto
- **lecturas.xlsx:** es el archivo donde se encuentra la base de datos utilizados para este proyecto
- **prediccion de ingresos.ipynb:** es el notebook en el que se trabajó todo el proceso presentado, desde el tratamiento de datos hasta las predicciones

El siguiente enlace direcciona a un repositorio de GitHub en el que se encuentra los mismos archivos distribuidos de la misma manera:

https://github.com/javier0097/prediccion_serie_de_tiempo