PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE HÁBITAT, INFRAESTRUCTURA Y CREATIVIDAD

MAESTRÍA EN SISTEMAS DE INFORMACIÓN MENCIÓN DATA SCIENCE



TEMA:

DESARROLLO DE UN MODELO PREDICTIVO PARA LA ESTIMACIÓN DE REQUERIMIENTOS EN UNA MESA DE SERVICIOS DE SOPORTE DE TI

AUTOR:

Catherine Rossana Rivas Ortiz

DIRECTOR:

Henry Nelson Roa Marin, PhD.

Quito, agosto de 2026

DEDICATORIA

Este proyecto de fin de titulación está dedicado a mi héroe, mi mentor, al que siempre me motivó a superarme profesional y personalmente que, aunque ya no este físicamente, su amor, sabiduría y guía continúan iluminando mi camino cada día. Gracias Papa Rigo por ser mi ejemplo de perseverancia y por enseñarme el valor del trabajo arduo y la dedicación, sé que me estás mirando desde arriba con orgullo mientras alcanzo este logro.

A mis queridos padres Jenny y Mario, que con su amor incondicional y sus palabras de aliento han sido un faro de luz en mi vida, esta titulación es un testimonio de su amor y apoyo inquebrantable, siempre estaré agradecida por el cariño y sabiduría.

A mi querido hijo, el motor de mi vida y la razón que me impulsa a superarme cada día. Cada esfuerzo, cada noche de estudio y cada meta alcanzada tienen como propósito brindarte un mejor futuro. Tu sonrisa y amor han sido la fuerza que necesitaba para no rendirme.

A mis hermanos Liz y Kevin, por el amor, apoyo y la inspiración que me han brindado en cada paso del camino, a mi hermosa familia agradezco profundamente su amor, comprensión y apoyo constante, este logro también es de ustedes, y lo comparto con gratitud y alegría.

AGRADECIMIENTO

Agradezco a Dios por la dicha de la vida y por haberme concedido la fortaleza, la sabiduría y la perseverancia necesaria para completar este proyecto de fin de titulación.

A mi director de proyecto PhD. Henry Roa, agradezco profundamente su orientación y su dedicación constante, su guía y liderazgo han sido fundamentales para alcanzar este logro.

A mi querida familia por confiar en mí, su confianza fue mi fortaleza.

Un agradecimiento especial a la Familia Mosquera Perugachi, por todo su apoyo en este proceso.

RESUMEN

Los servicios de soporte técnico enfrentan el desafío de una demanda variable de solicitudes, lo que complica la planificación de recursos y puede aumentar los tiempos de respuesta cuando la gestión se realiza únicamente de forma reactiva. Mediante la ciencia de datos y el aprendizaje automático, el presente estudio tuvo como objetivo desarrollar un modelo predictivo para estimar el número de solicitudes de un servicio de soporte técnico.

La investigación se llevó a cabo utilizando el método CRISP-DM en las fases de comprensión del negocio, comprensión de los datos, preparación de los datos, modelado y evaluación. Se recopilaron datos históricos de 910 observaciones diarias del servicio de soporte de la organización, desde el 25 de enero de 2023 hasta el 22 de julio de 2025. La preparación de los datos se realizó mediante la limpieza, la normalización de fechas, la construcción de una serie temporal continua y la generación de variables temporales para entrenar los modelos predictivos.

Se utilizaron tres modelos predictivos: SARIMA, Prophet y XGBoost, los cuales fueron evaluados en términos de MAE, RMSE, MAPE y WAPE. XGBoost obtuvo los MAE, RMSE y WAPE más bajos, mientras que SARIMA obtuvo el MAPE más bajo. Sin embargo, la diferencia entre los dos modelos fue pequeña, lo que indicó que el rendimiento fue similar en los datos que se analizaron. Además, se encontró que los tres modelos son buenos predictores de la tendencia y de la estacionalidad semanal de la demanda, pero tienen dificultades para predecir eventos extraordinarios que no se reflejan en el comportamiento histórico de la serie temporal.

En conclusión, se ha demostrado que los modelos predictivos son una herramienta de apoyo para la planificación operativa de un servicio de soporte técnico, al proporcionar proyecciones precisas del comportamiento de la demanda en condiciones normales de operación.

Palabras clave: mesa de servicios, series temporales, ciencia de datos, machine learning, SARIMA, Prophet, XGBoost.

ÍNDICE

1. Introducción 11
1.1. Antecedentes 11
1.2. Planteamiento del Problema 12
1.3. Justificación 13
1.4. Objetivos 14
1.4.1. Objetivo General 14
1.4.2. Objetivos Específicos 15
1.5. Alcance 15
2. Marco Teórico 16
2.1. Ciencia de datos 16
2.2. Mesa de Servicios de Tecnología de la información 16
2.3. Minería de Datos y metodología CRISP-DM 16
2.4. Machine Learning 17
2.5. Series temporales 17
2.6. Modelos predictivos (SARIMA, Prophet y XGBoost) 18
2.7. Evaluación de modelos predictivos 18
3. Metodología CRISP DM 19
3.1. Fases de la metodología CRISP-DM 19
3.1.1. Comprensión del negocio 19
3.1.2. Comprensión de los datos 20
3.1.3. Preparación de los datos 20
3.1.4. Modelados 20
3.1.5. Evaluación 20
3.1.6. Despliegue 20
4. Desarrollo de la metodología 21
4.1. Comprensión del negocio 21
4.1.1. Contexto de la mesa de servicios. 21
4.1.2. Objetivos del análisis. 21
4.1.3. Definición del problema. 22
4.2. Comprensión de los datos 22
4.2.1. Descripción del dataset. 22
4.2.2. Variables. 22
4.2.3. Calidad de los datos. 22
4.2.4. Análisis inicial. 23
4.3. Preparación de los datos 23

4.3.1. Limpieza. ... 23
4.3.2. Tratamiento de valores nulos. ... 25
4.3.3. Conversión de fechas. ... 27
4.3.4. Agregación semanal. ... 28
4.3.5. Construcción de la serie temporal. ... 28
4.4. Análisis exploratorio de datos ... 29
4.4.1. Estadísticos descriptivos. ... 30
4.4.2. Distribuciones. ... 30
4.4.3. Tendencia. ... 32
4.4.4. Estacionalidad. ... 33
4.4.5. Visualizaciones. ... 34
4.5. Desarrollo de los modelos predictivos ... 34
4.5.1. SARIMA. ... 34
4.5.2. Prophet. ... 36
4.5.3. XGBoost. ... 38
4.6. Evaluación y comparación de modelos ... 40
4.6.1. MAE ... 40
4.6.2. RMSE. ... 41
4.6.3. MAPE. ... 41
4.6.4. Comparación. ... 42
4.6.5. Selección del mejor modelo. ... 43
5. Análisis de Resultados ... 44
5.1. Análisis de la preparación de los datos ... 44
5.1.1. Evaluación de la calidad del conjunto de datos ... 44
5.1.2. Análisis del proceso de limpieza ... 47
5.1.3. Análisis de la transformación y construcción de la serie temporal ... 49
5.1.4. Impacto de la preparación de los datos en el desempeño de los modelos ... 52
5.2. Análisis del comportamiento de la serie temporal ... 55
5.2.1. Evolución temporal de los requisitos ... 55
5.2.2. Análisis de Tendencias ... 56
5.2.3. Análisis de la estacionalidad ... 57
5.2.4. Variabilidad y comportamiento residual ... 58
5.2.5. Implicaciones para el modelado predictivo. ... 59
5.3. Análisis del desempeño de los modelos predictivos ... 60
5.3.1. Análisis del desempeño del modelo SARIMA ... 60
Fundamentación del modelo. ... 60
Interpretación del entrenamiento. ... 61
Análisis de las predicciones ... 63

Interpretación de las métricas. 64
Fortalezas y limitaciones. 66
5.3.2. Análisis del desempeño del modelo Prophet 67
Comportamiento del modelo. 67
Interpretación de las predicciones. 69
Intervalos de confianza. 70
Análisis de las métricas. 71
Ventajas y limitaciones 71
5.3.3. Análisis del desempeño del modelo XGBoost 72
Construcción de variables. 72
Optimización de hiperparámetros. 74
Variables predictoras 76
Interpretación de las predicciones. 77
Análisis de las métricas. 77
Fortalezas y limitaciones. 78
5.4. Comparación de resultados 79
5.4.1. Comparación de las métricas de evaluación 80
5.4.2. Comparación del comportamiento predictivo 81
5.4.3. Análisis complementario considerando días laborables 82
5.4.4. Análisis complementario de predicción por categoría con Prophet 83
5.4.5. Selección del modelo 84
5.5. Discusión de resultados y cumplimiento de objetivos 85
5.5.1. Discusión de los resultados obtenidos 85
5.5.2. Cumplimiento del objetivo general 87
5.5.3. Cumplimiento de los objetivos específicos 88
5.5.4. Aporte de la investigación 89
5.5.5. Limitaciones de la investigación 91
5.5.6. Implicaciones para la organización 91
6. Conclusiones y Recomendaciones 92
6.1. Conclusiones 92
6.2. Recomendaciones 93
Bibliografía 94

ÍNDICE DE FIGURA

Figura 1 Metodología CRISP-DM. 19

Figura 2 Inspección inicial del archivo de tickets antes del proceso de limpieza. 24

Figura 3 Limpieza profunda y librería csv nativa. 25

Figura 4 Verificación de valores nulos por variable. 26

Figura 5 Conversión de la variable Fecha de registro al tipo datetime. 27

Figura 6 Construcción de la serie temporal y división cronológica en conjuntos de entrenamiento y prueba. 29

Figura 7 Serie temporal de tickets diarios registrados (enero de 2023 – julio de 2025). 30

Figura 8 Distribución de tickets por categoría. 31

Figura 9 Distribución de tickets por prioridad. 32

Figura 10 Optimización de hiperparámetros y predicción con el modelo XGBoost. 39

Figura 11 Comparación del MAE obtenido por los modelos SARIMA, Prophet y XGBoost 41

Figura 12 Construcción de la serie temporal continua a partir de los registros históricos. 46

Figura 13 Evaluación de los métodos de lectura del conjunto de datos. 47

Figura 14 Preparación de la serie temporal para el entrenamiento y validación de los modelos 53

Figura 15 Selección automática de parámetros del modelo SARIMA mediante AutoARIMA. 61

Figura 16 Entrenamiento del modelo Prophet y componentes del pronóstico. 68

Figura 17 Intervalos de confianza del modelo Prophet. 70

Figura 18 Variables construidas para el entrenamiento del modelo XGBoost. 73

Figura 19 Optimización de hiperparámetros mediante RandomizedSearchCV. 75

Figura 20 Construcción de variables predictoras para el modelo XGBoost 76

Figura 21 Pronóstico generado por el modelo XGBoost seleccionado. 90

ÍNDICE DE GRÁFICO

Gráfico 1 Proceso de agregación temporal de los tickets registrados. 28

Gráfico 2 Componente de tendencia de la serie temporal. 33

Gráfico 3 Componente estacional de la serie temporal. 33

Gráfico 4 Predicciones del modelo SARIMA sobre el conjunto de prueba. 36

Gráfico 5 Predicciones del modelo Prophet sobre el conjunto de prueba. 37

Gráfico 6 Comparación de las predicciones generadas por SARIMA, Prophet y XGBoost..... 43

Gráfico 7 Construcción de la serie temporal diaria a partir de los registros históricos. 50

Gráfico 8 Evolución temporal de los requerimientos registrados. 56

Gráfico 9 Componente de tendencia de la serie temporal. 57

Gráfico 10 Componente estacional de la serie temporal. 58

Gráfico 11 Componente residual de la serie temporal 59

Gráfico 12 Entrenamiento del modelo SARIMA y generación de predicciones en el conjunto de prueba. 62

Gráfico 13 Comparación entre los valores reales y las predicciones generadas por Prophet. .. 69

Gráfico 14 Comparación de las predicciones generadas por SARIMA, Prophet y XGBoost.... 81

ÍNDICE DE TABLAS

Tabla 1 Resume los resultados obtenidos. 42

Tabla 2 Resumen de la calidad y depuración del conjunto de datos 45

Tabla 3 Principales actividades desarrolladas durante el proceso de limpieza de datos 48

Tabla 4 Características de la serie temporal utilizada para el modelado 51

Tabla 5 Relación entre la preparación de los datos y el desempeño de los modelos 54

Tabla 6 Métricas de rendimiento del modelo SARIMA. 65

Tabla 7 Métricas de desempeño del modelo Prophet 71

Tabla 8 Métricas de desempeño del modelo XGBoost 78

Tabla 9 Comparación de las métricas de desempeño de los modelos predictivos 80

Tabla 10 Resultados del análisis complementario considerando días laborables 83

Tabla 11 Desempeño de Prophet por categoría 84

Tabla 12 Comparación de las métricas de desempeño de los modelos predictivos 87

Tabla 13 Cumplimiento de los objetivos específicos 89

1. Introducción

1.1. Antecedentes

Las organizaciones dependen cada vez más de las Tecnologías de la Información (TI) para garantizar la continuidad de sus procesos de negocio. En este contexto, las mesas de servicio constituyen un componente fundamental para la gestión de incidentes y requerimientos, ya que centralizan la atención a los usuarios, coordinan la resolución de problemas y contribuyen al cumplimiento de los acuerdos de nivel de servicio (Service Level Agreements, SLA). Sin embargo, el incremento continuo del volumen de solicitudes plantea desafíos para la asignación eficiente de recursos, la planificación operativa y la reducción de los tiempos de respuesta.

La gestión tradicional de una mesa de servicios suele basarse en la experiencia del personal y en el análisis descriptivo de los registros históricos. Aunque estos enfoques permiten comprender el comportamiento pasado de la operación, presentan limitaciones para anticipar la demanda futura y para apoyar la toma de decisiones estratégicas. En este sentido, el uso de técnicas de ciencia de datos y de aprendizaje automático permite identificar patrones temporales y construir modelos predictivos capaces de estimar el comportamiento futuro de los requerimientos, lo que facilita la planificación de recursos y mejora la calidad del servicio (Han, Kamber & Pei, 2012).

En los últimos años, el análisis predictivo ha adquirido un papel relevante en la gestión de servicios de TI debido a la disponibilidad de grandes volúmenes de datos generados por herramientas de gestión de incidencias. Estos datos permiten aplicar modelos de series temporales y algoritmos de aprendizaje automático para pronosticar la carga de trabajo, optimizar la asignación del personal y reducir el impacto de los picos de demanda. Según Hyndman y Athanasopoulos (2021), los modelos de pronóstico constituyen herramientas esenciales para apoyar la planificación operativa cuando existen patrones de tendencia, estacionalidad o variaciones temporales en los datos.

El presente trabajo propone desarrollar un modelo predictivo para estimar el número de requerimientos registrados en una mesa de servicios de soporte de TI a partir de la información histórica de los tickets. Para ello, se implementa un proceso metodológico basado en CRISP-DM (Cross-Industry Standard Process for Data Mining), ampliamente reconocido para proyectos de minería de datos y analítica predictiva, que comprende las

fases de comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue (Chapman et al., 2000).

De acuerdo con el notebook desarrollado para esta investigación, el proceso metodológico incluye la importación y depuración del conjunto de datos, el tratamiento de valores faltantes y registros inconsistentes, la transformación de variables temporales, el análisis exploratorio de los datos y la construcción de modelos predictivos basados en SARIMA, Prophet y XGBoost, con el propósito de comparar su desempeño mediante métricas objetivas de evaluación. La selección de estos modelos responde a que representan enfoques complementarios para el pronóstico de series temporales: los modelos estadísticos tradicionales, los modelos aditivos para series temporales y los algoritmos de aprendizaje automático.

Los resultados obtenidos permitirán identificar el modelo con la mejor capacidad predictiva para estimar el comportamiento futuro de los requerimientos de la mesa de servicios, lo que proporcionará información útil para apoyar la planificación de recursos humanos, la gestión de la capacidad operativa y la toma de decisiones en el área de soporte tecnológico.

1.2. Planteamiento del Problema

Los servicios de soporte técnico desempeñan un papel fundamental en la continuidad de una organización al recibir, registrar, clasificar y gestionar las solicitudes e incidentes de los clientes. La eficiencia con la que se gestionan estas solicitudes afecta directamente la disponibilidad de los servicios tecnológicos, la productividad de los usuarios y la calidad del soporte ofrecido. En este sentido, la gestión de tickets es uno de los procesos más importantes en las operaciones de soporte técnico, ya que permite organizar el flujo de trabajo y cumplir con los niveles de servicio.

Sin embargo, el uso cada vez mayor de la tecnología ha resultado en un aumento de solicitudes, que ahora son gestionadas por los servicios de soporte. Junto con el crecimiento de la demanda, hay variaciones temporales en los tickets que dificultan la planificación de los recursos humanos y la distribución eficiente del trabajo. Por esta razón, la capacidad disponible no es suficiente para atender las solicitudes que se registran.

Como resultado de esta variabilidad en la demanda, las organizaciones se ven afectadas por la saturación de solicitudes, la asignación ineficiente del personal de soporte y

tiempos de respuesta y de resolución más largos. Estas condiciones son perjudiciales para el servicio de soporte y para los usuarios, quienes experimentan retrasos en la atención de sus solicitudes y una menor calidad del servicio recibido.

En cuanto al plan de tesis, se ha encontrado que una de las principales limitaciones es la falta de herramientas para predecir el comportamiento de la demanda en el futuro. Si la planificación se realiza únicamente en función del número de tickets ya registrados, la gestión del servicio es reactiva y no puede asignar recursos con anticipación ni reaccionar con mayor rapidez ante situaciones de mayor carga operativa.

En esta situación, la ciencia de datos y los modelos predictivos constituyen una alternativa para transformar la información histórica de los tickets en conocimiento útil para la toma de decisiones. Los registros históricos pueden utilizarse para identificar patrones de comportamiento, tendencias y estacionalidad, a fin de estimar la demanda futura de solicitudes. Esto puede apoyar la planificación de recursos, la gestión de la capacidad operativa y la toma de decisiones en el servicio de soporte.

A partir de este problema, se propone desarrollar un modelo predictivo para estimar el número de solicitudes registradas en un servicio de soporte técnico, aplicando la metodología CRISP-DM e implementando modelos basados en SARIMA, Prophet y XGBoost. La comparación objetiva del rendimiento de estos modelos ayudará a identificar la alternativa con la mejor capacidad predictiva, a fin de apoyar la planificación operativa y contribuir a una gestión más eficiente de las solicitudes de soporte tecnológico, de acuerdo con el alcance definido en la investigación.

1.3. Justificación

Esta investigación está motivada por la necesidad de herramientas que apoyen la planificación de la demanda de requisitos en un servicio de mesa de ayuda de TI. La variabilidad del número de solicitudes registradas dificulta la asignación de recursos y la planificación de la capacidad operativa cuando las decisiones se basan en la información disponible en el momento del servicio. El desarrollo de un modelo predictivo basado en datos históricos es una alternativa para estimar el comportamiento esperado de la demanda y proporcionar información que contribuya a la planificación de las actividades de soporte.

Como metodología, la investigación se basa en el método CRISP-DM, que proporciona una estructura organizada para el desarrollo de proyectos de ciencia de datos en las fases

de comprensión del negocio, comprensión de los datos, preparación de los datos, modelado y evaluación. Este método permitió desarrollar un enfoque sistemático para recopilar la información histórica de la mesa de ayuda, obtener trazabilidad de las actividades y reproducir el procedimiento utilizado en la construcción de los modelos predictivos. Según el alcance de esta investigación, el desarrollo abarca las etapas hasta la evaluación de los modelos y se propone, como dirección futura, la implementación del modelo seleccionado en el entorno operativo de la organización.

El estudio también incorpora series temporales y algoritmos de aprendizaje automático mediante los modelos SARIMA, Prophet y XGBoost para comparar el rendimiento predictivo de distintos modelos sobre el mismo conjunto de datos. Este estudio proporciona una visión del rendimiento de los modelos estadísticos tradicionales y de los algoritmos de aprendizaje automático para prever los requisitos de una mesa de ayuda, y ayuda a identificar alternativas que puedan apoyar la planificación operativa de dichas organizaciones.

Así, desde una perspectiva organizacional, los resultados obtenidos son útiles para comprender el comportamiento esperado de la demanda, prepararse para los recursos humanos y la programación, y compartir la carga de trabajo de la mesa de ayuda. La investigación no tiene como objetivo optimizar la gestión del servicio ni automatizar la toma de decisiones, sino generar un modelo predictivo que pueda utilizarse como herramienta de apoyo para la planificación operativa basada en información histórica.

Finalmente, la investigación contribuye a la comunidad académica al ilustrar el uso de métodos de ciencia de datos y de modelos predictivos en la gestión de los departamentos de servicios de TI. El procedimiento y los resultados obtenidos en este trabajo serán útiles para futuras investigaciones sobre la previsión de los requisitos de la mesa de ayuda, así como para incorporar nuevas variables explicativas u otros modelos predictivos en entornos organizacionales similares.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un modelo predictivo del número de requerimientos que proporcione estimaciones para apoyar la planificación operativa y la toma de decisiones en la mesa de servicio de una empresa de soporte IT.

1.4.2. Objetivos Específicos

- Colectar datos históricos de requerimientos de la mesa de servicio
- Preprocesar los datos recopilados
- Desarrollar modelos predictivos basados en metodologías de ciencia de datos y en modelos de machine learning.

1.5. Alcance

En esta investigación se desarrolló y evaluó un modelo predictivo para estimar el comportamiento de los requisitos registrados en un servicio de mesa de ayuda de TI, a partir de datos históricos de tickets. Se utilizó la metodología CRISP-DM para evaluar y comprender el negocio, así como para la comprensión y preparación de datos, el modelado y la evaluación (incluyendo limpieza, transformación y análisis exploratorio). Luego se implementan y comparan los modelos SARIMA, Prophet y XGBoost y se elige el mejor modelo predictivo para proporcionar a la mesa de ayuda el mayor poder predictivo mientras se prepara para las operaciones. El análisis de la información histórica en el conjunto de datos es limitado, y no se consideró factores externos, como cambios organizacionales, tecnológicos o eventos extraordinarios no registrados.

El estudio se centra en la construcción y evaluación de modelos predictivos, sin considerar su implementación en el entorno de producción ni su integración con plataformas ITSM. Esto es consistente con la metodología de minería de datos CRISP-DM, que establece que los modelos predictivos deben desarrollarse a partir de datos de calidad y mediante un proceso de evaluación sistemático antes de su despliegue (Chapman et al., 2000).

2. Marco Teórico

2.1. Ciencia de datos

La ciencia de datos es un campo multidisciplinario que combina conocimientos de estadística, informática, matemáticas y del dominio para extraer información útil a partir de grandes volúmenes de datos. La ciencia de datos tiene como objetivo convertir los datos en conocimiento para la toma de decisiones mediante métodos descriptivos, predictivos y prescriptivos (Provost & Fawcett, 2013).

La ciencia de datos se ha aplicado en las organizaciones, ya que permite el descubrimiento de patrones y la predicción de comportamientos, así como la optimización de procesos mediante el aprendizaje automático y la minería de datos. En las empresas modernas, la ciencia de datos ha ayudado a mejorar la eficiencia operativa mediante el análisis de datos, la automatización de procesos y el entrenamiento de modelos predictivos para la planificación estratégica (Han, Kamber & Pei, 2012).

La ciencia de datos es la metodología que permite analizar registros históricos de tickets de servicio y desarrollar modelos para predecir el comportamiento futuro de las solicitudes.

2.2. Mesa de Servicios de Tecnología de la información

Un service desk es el único punto de contacto entre los usuarios y el proveedor de servicios de TI, y es responsable de gestionar incidentes, solicitudes de servicio y requerimientos relacionados con la infraestructura tecnológica. Se describe en ITIL 4 que el service desk es una parte esencial de la entrega y el soporte de servicios, ya que es clave en el proceso de comunicación con los usuarios y ayuda a cumplir con los acuerdos de nivel de servicio (AXELOS, 2019). La información histórica generada por las herramientas de ITSM se utiliza para analizar el comportamiento de los requisitos a lo largo del tiempo. Estos son los registros para utilizar técnicas de análisis predictivo y hacer más eficiente la planificación de recursos, mejorar los tiempos de respuesta y optimizar la operación del service desk.

2.3. Minería de Datos y metodología CRISP-DM

La minería de datos es el proceso de encontrar patrones, relaciones y conocimiento útil a partir de grandes conjuntos de datos mediante métodos estadísticos y algoritmos de aprendizaje automático (Han et al., 2012).

En este contexto, CRISP-DM (Proceso Estándar de la Industria para la Minería de Datos) es uno de los modelos de referencia más utilizados en proyectos de minería de datos por su enfoque detallado y su independencia de la tecnología empleada. CRISP-DM se compone de seis etapas: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue (Chapman et al., 2000).

En este estudio, estas etapas guían el desarrollo del proceso analítico en el notebook, desde la limpieza del conjunto de datos hasta la formación y comparación de modelos predictivos.

2.4. Machine Learning

El aprendizaje automático (ML) es una rama de la inteligencia artificial que se encarga de aprender patrones a partir de datos para hacer predicciones o tomar decisiones, sin necesidad de un algoritmo programado para esa tarea (Mitchell, 1997). Entre sus principales enfoques se encuentran el aprendizaje supervisado, el no supervisado y el por refuerzo. En la presente aplicación, se utilizó el aprendizaje supervisado porque el objetivo es predecir el número de necesidades futuras a partir de los datos disponibles previamente del servicio de asistencia.

El aprendizaje automático puede aplicarse al análisis de grandes volúmenes de datos operativos para identificar patrones y mejorar la toma de decisiones en tecnología de la información, por lo que este estudio también introduce algoritmos de aprendizaje automático en el servicio de asistencia técnica de TI. El algoritmo XGBoost se utiliza como método de aprendizaje automático para comparar su capacidad predictiva con los modelos de series temporales SARIMA y Prophet, a fin de identificar el mejor modelo para estimar la demanda del servicio de asistencia técnica de TI.

2.5. Series temporales

Una serie temporal es un conjunto de datos recopilados en orden cronológico. A partir de series temporales, se pueden estudiar tendencias, estacionalidad y comportamiento cíclico para realizar predicciones futuras (Hyndman & Athanasopoulos, 2021).

Los modelos de series temporales se aplican ampliamente en la planificación de la demanda, la economía, la salud, la logística y la tecnología de la información. En este estudio, los datos históricos de tickets constituyen una serie temporal porque representan el número de solicitudes registradas a intervalos regulares. Con base en los datos

recopilados durante el período de la serie temporal, es posible construir modelos de demanda futura del servicio de asistencia.

2.6. Modelos predictivos (SARIMA, Prophet y XGBoost)

El modelo SARIMA incluye componentes autorregresivos, promedios móviles y diferenciación estacional, y puede usarse para modelar series temporales con patrones repetitivos y comportamiento estacional (Box et al., 2016).

Prophet, desarrollado por Meta, es un modelo aditivo que puede generar pronósticos robustos de series temporales con tendencia y estacionalidad, con la mínima intervención del usuario (Taylor & Letham, 2018).

Por otro lado, XGBoost es un algoritmo de aprendizaje automático basado en árboles de decisión, mejorado mediante boosting por gradiente. Su capacidad para capturar relaciones no lineales y manejar grandes volúmenes de datos lo convierte en una alternativa ampliamente utilizada en problemas predictivos (Chen & Guestrin, 2016).

2.7. Evaluación de modelos predictivos

Los modelos predictivos se evalúan en términos de precisión mediante medidas cuantitativas que comparan los valores estimados con los valores reales. Las métricas más comúnmente utilizadas para comparar el rendimiento de diferentes modelos son el error absoluto medio (MAE), el error cuadrático medio (RMSE) y el error porcentual absoluto medio (MAPE) (Hyndman & Koehler, 2006).

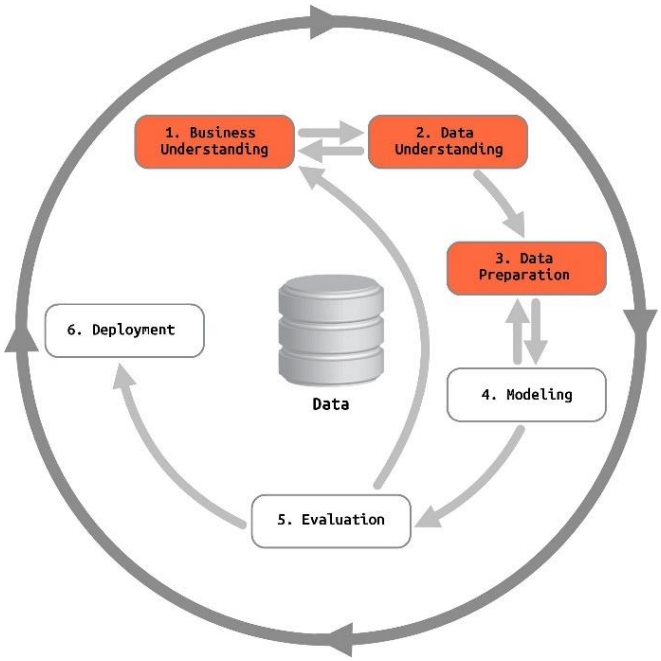
En este trabajo, estas métricas ayudarán a determinar cuál de los modelos presenta el mejor rendimiento para estimar el comportamiento futuro de los requisitos registrados en el servicio de atención.

3. Metodología CRISP DM

El enfoque CRISP-DM (Proceso Estándar de la Industria para la Minería de Datos) es uno de los estándares más utilizados en proyectos de minería de datos y de análisis predictivo. Fue desarrollado para proporcionar un enfoque estructurado y flexible de minería de datos, independiente de la tecnología utilizada, y para aprovechar los datos en la toma de decisiones (Chapman et al., 2000). Puede iterarse hacia fases anteriores si se identifican oportunidades de mejora o cuando se requiere mejorar el modelo, y permite desarrollar soluciones robustas que pueden adaptarse a diferentes dominios de aplicación.

CRISP-DM consta de seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y despliegue. Aunque típicamente se conceptualiza como secuencial, en realidad el proceso es iterativo, ya que los resultados obtenidos pueden requerir modificaciones en fases anteriores para mejorar el rendimiento del modelo (Chapman et al., 2000).

Figura 1 Metodología CRISP-DM.



Fuente: Adaptado de Chapman et al. (2000).

3.1. Fases de la metodología CRISP-DM

3.1.1. Comprensión del negocio

Esta fase consiste en identificar el problema que motiva la investigación, establecer los objetivos del proyecto y definir los criterios para evaluar el éxito de la solución propuesta. En esta etapa, se consideran el contexto organizacional y las necesidades del usuario para

asegurar que el modelo predictivo se construya para abordar el problema real y aportar valor a la organización (Chapman et al., 2000).

3.1.2. Comprensión de los datos

En esta etapa se recopilan los datos y se realiza un análisis inicial para comprender su estructura, calidad y contenido. También se identifican posibles inconsistencias, valores faltantes, registros duplicados y otros aspectos que pueden afectar el desarrollo del modelo. Esta etapa permite comprender el comportamiento de la información antes de iniciar el proceso de preparación de datos.

3.1.3. Preparación de los datos

La preparación de datos implica la limpieza, transformación, integración y selección de las variables necesarias para el modelado. Esto se hace para garantizar un conjunto de datos consistente y adecuado para entrenar modelos predictivos que reduzcan el impacto de errores, datos incompletos o información irrelevante.

3.1.4. Modelados

En esta fase se utilizaron distintos modelos de predicción para estimar el comportamiento futuro de los requerimientos registrados en el service desk. Con base en el desarrollo realizado en el notebook, los modelos SARIMA, Prophet y XGBoost son enfoques estadísticos y de aprendizaje automático para la previsión de series temporales. Los modelos se entrenaron con los datos históricos recopilados en las fases anteriores para comparar sus predicciones.

3.1.5. Evaluación

El propósito de la evaluación es verificar que los modelos desarrollados cumplan con los objetivos establecidos en la fase de comprensión del negocio. Como tal, se analizan diferentes métricas de rendimiento y se comparan los resultados obtenidos para seleccionar el modelo con la mayor capacidad predictiva y el mejor ajuste a los datos.

3.1.6. Despliegue

La fase de despliegue es la en la que el modelo seleccionado se implementa en un entorno operativo para apoyar la toma de decisiones. Dependiendo del alcance del proyecto, esto podría incluir la integración del modelo con sistemas de información existentes, la automatización de procesos y, en general, la previsión para mejorar la gestión organizacional.

4. Desarrollo de la metodología

En este capítulo se describe la metodología que se aplicó para desarrollar el modelo predictivo de requisitos de un servicio de mesa de ayuda de TI. La metodología se desarrolla tras las etapas de CRISP-DM, en las que se describe el proceso de comprensión del problema y luego se evalúa los modelos predictivos. La redacción se basa en las actividades del notebook de Python y los resultados del procesamiento de datos.

4.1. Comprensión del negocio

4.1.1. Contexto de la mesa de servicios.

El service desk es el punto de contacto principal para gestionar incidentes y solicitudes de soporte tecnológico en una organización empresarial. Su función principal es registrar, clasificar y atender las solicitudes de los usuarios para que se cumplan los acuerdos de nivel de servicio y se mantenga la continuidad operativa.

Los datos utilizados para la investigación corresponden a tickets históricos de enero de 2023 a julio de 2025, lo que cubre más de dos años y medio de operación del service desk. Los datos de esos tickets contienen información sobre las solicitudes recibidas y constituyen la base para identificar patrones temporales y modelos de pronóstico.

Analizar esta información puede ayudar a predecir el comportamiento futuro de la demanda, mejorar la asignación de recursos y la gestión del personal, y apoyar los procesos de toma de decisiones basados en datos.

4.1.2. Objetivos del análisis.

El propósito del análisis fue construir un modelo predictivo capaz de estimar el número de solicitudes registradas en el service desk a partir de información histórica. Para este propósito, se estableció el siguiente flujo de trabajo: inspeccionar y comprender la estructura de los datos; evaluar la calidad y la corrección de los datos; transformar los datos en una serie temporal para su análisis; analizar el comportamiento de las solicitudes y las visualizaciones; construir modelos predictivos con SARIMA, Prophet y XGBoost; evaluar el rendimiento de los modelos con herramientas de evaluación objetivas.

De esta manera, el modelo predictivo se construyó a partir de información consistente que refleja el comportamiento histórico del service desk.

4.1.3. Definición del problema.

El aumento del número de solicitudes complica la planificación operativa cuando las decisiones se basan en el comportamiento histórico observado. Cuando no es posible anticipar la demanda, los recursos se asignan de manera deficiente y los tiempos de respuesta y la calidad del servicio se reducen.

En este contexto, se propone desarrollar un modelo predictivo para estimar el número de solicitudes futuras a partir de los registros históricos de tickets. Para ello, se ha utilizado análisis de series temporales y técnicas de aprendizaje automático para identificar el modelo que ofrezca la mayor precisión en la estimación de la demanda.

4.2. Comprensión de los datos

4.2.1. Descripción del dataset.

El conjunto de datos utilizado es un archivo tipo csv que contiene el historial de los tickets registrados por el servicio de asistencia desde enero de 2023 hasta julio de 2025. El notebook describe primero la inspección del archivo en bruto para detectar posibles problemas estructurales antes de realizar cualquier cambio.

En esta etapa, se observa que el archivo presentaba inconsistencias de formato derivadas del proceso de exportación. Por ejemplo, había delimitadores adicionales al final de cada línea y la estructura del archivo era distinta. Se utilizaron dos estrategias de lectura: un primer intento con una limpieza ligera usando pandas y un segundo intento con una limpieza profunda usando la biblioteca csv de Python. Tras comparar ambos métodos, se optó por el segundo, ya que recuperó correctamente la estructura del archivo y evitó inconsistencias durante la carga de datos.

4.2.2. Variables.

Una vez cargado el conjunto de datos, el primer análisis mostró que la base de datos estaba compuesta por 12,191 registros y 12 variables que inicialmente se interpretaron como datos de texto (objeto). Estos son campos relacionados con el identificador del ticket, el tipo de solicitud, la prioridad, las fechas de registro, etc. Esta revisión permitió identificar columnas que requerían transformación antes de modelar.

4.2.3. Calidad de los datos.

La primera inspección detectó problemas estructurales que podrían afectar el análisis. Entre ellos estaban la columna de Prioridad, diferentes formatos de fecha de registro y

valores especiales en la columna Expirado, debido al sistema de gestión utilizado para registrar los tickets. Estas observaciones llevaron a una limpieza, luego se inició el análisis exploratorio.

4.2.4. Análisis inicial.

Como parte de la comprensión de los datos, se verificó la estructura general del conjunto de datos, el tipo de información almacenada en cada variable y la presencia de posibles anomalías. Esta revisión estableció una línea de base de la calidad de la información y definió las estrategias de preparación para obtener un conjunto de datos coherente para el modelado predictivo.

4.3. Preparación de los datos

La fase de preparación de datos tenía como objetivo contar con un conjunto de datos coherente y adecuado para entrenar modelos predictivos. Según CRISP-DM, la fase de preprocesamiento consiste en seleccionar, limpiar, transformar e integrar datos antes de modelar, y es una de las fases principales que consumen más tiempo en un proyecto de minería de datos. En el notebook, el preprocesamiento se realizó en una secuencia de pasos. Primero se leyó el archivo y luego se construyeron las series temporales con los modelos SARIMA, Prophet y XGBoost.

4.3.1. Limpieza.

Como primer trabajo, se examinó el archivo original exportado por el servicio de asistencia, con la esperanza de detectar alguna inconsistencia estructural antes de cargarlo en la memoria. Se inició revisando el contenido del archivo en formato de texto para verificar el número total de líneas, el encabezado y los registros representativos del principio, del medio y del final del archivo. Esto permitió identificar los caracteres especiales y verificar la estructura del archivo durante la preparación del proceso de limpieza.

Figura 2 Inspección inicial del archivo de tickets antes del proceso de limpieza.

```
# Abrir el archivo en modo lectura con encoding latin-1 para soportar tildes y ñ
# readlines() carga todas las líneas del archivo en una lista
nombre_archivo = list(uploaded.keys())[0]  # toma el nombre real automáticamente
with io.StringIO(uploaded[nombre_archivo].decode("latin-1")) as f:
    lineas = f.readlines()

# Mostrar el total de líneas del archivo (incluye encabezado)
print(f"Total de líneas: {len(lineas)}")

# Inspeccionar las primeras 3 líneas para ver el encabezado y los primeros registros
print("\n--- PRIMERAS 3 ---")
for l in lineas[:3]:
    print(repr(l))  # repr() muestra los caracteres invisibles como \n, \t, comillas, etc.

# Inspeccionar 3 líneas del medio para detectar si el formato cambia a lo largo del archivo
print("\n--- MITAD ---")
mitad = len(lineas) // 2  # // divide y redondea hacia abajo para obtener el índice del centro
for l in lineas[mitad:mitad+3]:
    print(repr(l))

# Inspeccionar las últimas 3 líneas para verificar cómo termina el archivo
print("\n--- ÚLTIMAS 3 ---")
for l in lineas[-3:]:
    print(repr(l))
```

Fuente: Elaboración propia a partir del notebook

Luego, en el notebook se aplicó un proceso de limpieza estructural para asegurar que la información pudiera ser leída correctamente por las bibliotecas de análisis de datos. En este proceso, se consideraron las alternativas de lectura de archivos y se seleccionó la que mejor recuperaría los registros y las variables del conjunto de datos para el análisis. Esta validación evitó que los errores derivados del formato del archivo afectaran el resto del análisis.

Como resultado, se recopiló un conjunto de datos consistente, que contenía 12,191 registros en 12 variables y conservaba toda la información relevante para el estudio. La limpieza de los datos obtenidos al exportarlos eliminó inconsistencias y preservó la representatividad de los datos utilizados para el modelado.

Figura 3 Limpieza profunda y librería csv nativa.

```
import csv

# Lista vacía para guardar las líneas limpias del intento 2
cleaned_lines_v2 = []

# Limpieza profunda: ';;;', comillas externas y dobles dobles
for line in uploaded[nombre_archivo].decode("latin-1").splitlines():
    l = line.strip()
    if l.endswith(';;;'):
        l = l[:-3]
    if l.startswith('"') and l.endswith('"'):
        l = l[1:-1]
        l = l.replace('""', '"')
    cleaned_lines_v2.append(l)

# Parsear con la librería csv nativa
filas_parseadas = []
encabezado = None

reader = csv.reader(cleaned_lines_v2, quotechar='"', delimiter=',', skipinitialspace=True)
for i, fila in enumerate(reader):
    if i == 0:
        encabezado = fila
    elif len(fila) == len(encabezado):
        filas_parseadas.append(fila)

df_intento2 = pd.DataFrame(filas_parseadas, columns=encabezado)

print(f"Filas cargadas:     {len(df_intento2)}")
print(f"Filas en archivo:   {len(cleaned_lines_v2) - 1}")
print(f"Filas descartadas:  {len(cleaned_lines_v2) - 1 - len(df_intento2)}")
```

```
Filas cargadas:     12191
Filas en archivo:   12357
Filas descartadas:  166
```

Fuente: Elaboración propia a partir del notebook

La diferencia entre las 12.193 filas reportadas en el primer intento y los 12.191 registros recuperados mediante el segundo procedimiento se produjo durante la limpieza estructural, ya que este último conservó únicamente las filas cuya cantidad de campos coincidía con las 12 columnas definidas en el encabezado; por tanto, los 12.191 registros corresponden al conjunto correctamente estructurado utilizado para las validaciones posteriores.

4.3.2. Tratamiento de valores nulos.

Una vez realizada la limpieza estructural, se verificó que había valores faltantes en cada una de las variables del conjunto de datos. El propósito del ejercicio era comprobar si se utilizarían registros incompletos para entrenar los modelos predictivos. Se hizo calculando el número de valores nulos en cada columna y cómo estos afectan la calidad de la información.

Figura 4 Verificación de valores nulos por variable.

```
# Lista vacía para guardar las líneas limpias del intento 1
cleaned_lines_v1 = []
# Limpieza ligera: solo eliminar los ';;;' del final
for line in uploaded[nombre_archivo].decode("latin-1").splitlines():
    cleaned_line = line.strip()
    if cleaned_line.endswith(';;;'):
        cleaned_line = cleaned_line[:-3]
    cleaned_lines_v1.append(cleaned_line)
# Unir las líneas en un texto único
cleaned_data_v1 = "\n".join(cleaned_lines_v1)

# Cargar con pandas usando el motor de Python
df_intento1 = pd.read_csv(
    io.StringIO(cleaned_data_v1),
    sep=',',
    quotechar='"',
    engine='python',
    on_bad_lines='skip'
)
print(f"Filas cargadas:     {len(df_intento1)}")
print(f"Filas en archivo:   {len(cleaned_lines_v1) - 1}")
print(f"Filas descartadas:  {len(cleaned_lines_v1) - 1 - len(df_intento1)}")

print("VERIFICACIÓN DEL INTENTO 1")
print(f"\nPrimera fila:")
print(df_intento1.iloc[0])
print(f"\nNulos por columna:")
print(df_intento1.isnull().sum())
```

Fuente: Elaboración propia a partir del notebook

Aunque pandas reporta haber cargado 12,193 filas, la inspección de la primera fila revela que el parseo falló silenciosamente: toda la fila se almacenó en la columna 'Código', mientras que las 11 columnas restantes contienen exclusivamente valores nulos (12,193 NaN en cada una).

Esto ocurre porque al no eliminar las comillas dobles externas que envuelven cada fila, pandas las interpreta como un único valor protegido y no detecta las comas internas como separadores de columnas. El resultado es un DataFrame estructuralmente correcto, pero funcionalmente inservible, lo que justifica probar una estrategia de limpieza más exhaustiva en el Intento 2.

La verificación confirma que el Dataset se cargó correctamente. Cada una de las 12 columnas contiene el dato esperado en la primera fila: el código del ticket, la fecha de registro, la prioridad, el título, el solicitante, la descripción, el técnico, la categoría, el

grupo y el SLA. Adicionalmente, ninguna columna presenta valores nulos derivados del parseo, lo que indica que la información se distribuyó sin pérdidas en las filas válidas.

Esto valida que la limpieza profunda combinada con la librería csv nativa era la estrategia adecuada para este archivo.

4.3.3. Conversión de fechas.

La variable Fecha de Registro fue el atributo principal para generar la serie temporal, por lo que fue necesario convertirla a tipo datetime. Durante la inspección, se detectó diferentes formatos de fecha en el mismo conjunto de datos, por lo que no se pudo realizar la operación cronológica.

Figura 5 Conversión de la variable Fecha de registro al tipo datetime.

```
# Crear nuevo DataFrame para la conversión de fechas
df_fechas = df_limpio.copy()

# Convertir con dos formatos distintos
# Primer intento: formato con guiones DD-MM-YYYY HH:MM:SS
fechas_guiones = pd.to_datetime(
    df_fechas['Fecha de registro'],
    format='%d-%m-%Y %H:%M:%S',
    errors='coerce'
)

# Segundo intento: formato con slashes D/M/YYYY H:MM
fechas_slashes = pd.to_datetime(
    df_fechas['Fecha de registro'],
    dayfirst=True,
    format='mixed',  # permite múltiples variantes
    errors='coerce'
)

# Combinar: usar el formato guiones si existe, si no usar slashes
df_fechas['Fecha de registro'] = fechas_guiones.fillna(fechas_slashes)

# Verificar
print(f"Tipo: {df_fechas['Fecha de registro'].dtype}")
print(f"Fechas inválidas: {df_fechas['Fecha de registro'].isna().sum()}")
print(f"\nRango: {df_fechas['Fecha de registro'].min()} a {df_fechas['Fecha de registro'].max()}")
```

```
Tipo: datetime64[ns]
Fechas inválidas: 0
```

Fuente: Elaboración propia a partir del notebook

La columna 'Fecha de registro' fue convertida exitosamente a tipo datetime64. Al detectarse 9 patrones distintos de formato en el dataset, se aplicó una estrategia de conversión en dos pasos:

1. Primer intento con el formato estricto %d-%m-%Y %H:%M:%S para el patrón mayoritario (7,871 filas con guiones).

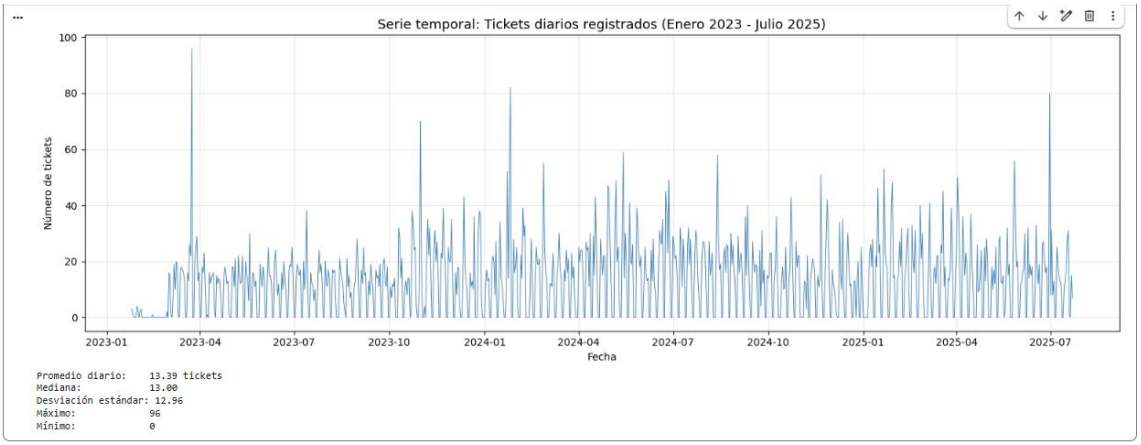
2. Segundo intento con format='mixed' y dayfirst=True para los patrones con slashes (4,318 filas restantes).

Los resultados se combinaron usando fillna(), lo que dejó 0 fechas inválidas. El dataset cubre el período del 25 de enero de 2023 al 22 de julio de 2025 (910 días), un horizonte temporal suficiente para entrenar modelos de pronóstico y capturar patrones estacionales semanales y mensuales.

4.3.4. Agregación semanal.

Para entender el comportamiento temporal de los requisitos, los registros individuales se agruparon por fecha de registro. Esto permitió convertir los tickets individuales en una lista de observaciones agregadas, lo que permitió identificar patrones de comportamiento y realizar predicciones. El notebook muestra que, para modelar la serie diaria, se consideraron los días sin registros para que la serie no se interrumpiera en el tiempo.

Gráfico 1 Proceso de agregación temporal de los tickets registrados.



Fuente: Elaboración propia a partir del notebook

La agregación temporal permitió reducir la variabilidad inherente a los registros individuales y proporcionó una representación más adecuada del comportamiento de la demanda a lo largo del tiempo. Esta transformación constituye uno de los pasos fundamentales para la aplicación de modelos de series temporales, ya que estos requieren observaciones ordenadas cronológicamente y sin interrupciones.

4.3.5. Construcción de la serie temporal.

La serie temporal final utilizada para el entrenamiento de los modelos predictivos se seleccionó durante el proceso de preparación de datos. Para asegurar la validez de la evaluación, la serie de datos se dividió en un 80% para entrenamiento y un 20% para

pruebas, evitando la mezcla aleatoria de observaciones y preservando el orden temporal de los registros. Esto es consistente con la mejor práctica en el análisis de series temporales, en la que la información futura no debe utilizarse durante el entrenamiento de los modelos.

Figura 6 Construcción de la serie temporal y división cronológica en conjuntos de entrenamiento y prueba.

```
# Crear nuevo DataFrame para la fase de modelado
df_modelo = df_atributos.copy()

# Construir la serie diaria final
serie_diaria = df_modelo.groupby(df_modelo['Fecha de registro'].dt.date).size()
serie_diaria.index = pd.to_datetime(serie_diaria.index)
serie_diaria = serie_diaria.asfreq('D').fillna(0)
serie_diaria.name = 'tickets'

# División train/test (80% / 20%)
# Debido a que es serie temporal no sse hace shuffle
punto_corte = int(len(serie_diaria) * 0.8)
train = serie_diaria.iloc[:punto_corte]
test = serie_diaria.iloc[punto_corte:]

print(f"Total de días:      {len(serie_diaria)}")
print(f"Días entrenamiento: {len(train)} ({len(train)/len(serie_diaria)*100:.1f}%)")
print(f"Días prueba:        {len(test)} ({len(test)/len(serie_diaria)*100:.1f}%)")
print(f"\nFecha inicio train: {train.index[0].date()}")
print(f"Fecha fin train:    {train.index[-1].date()}")
print(f"Fecha inicio test:  {test.index[0].date()}")
print(f"Fecha fin test:     {test.index[-1].date()}")
```

Fuente: Elaboración propia a partir del notebook

4.4. Análisis exploratorio de datos

El análisis exploratorio de datos (EDA) se realizó para comprender el comportamiento de los requisitos registrados en el servicio de atención antes de construir modelos predictivos. Esta etapa proporcionó información de series temporales, tendencias, componentes estacionales y distribución de las principales variables en el conjunto de datos, para poder elegir y configurar modelos de pronóstico.

Basado en el enfoque CRISP-DM, el análisis exploratorio se desarrolló después de la preparación de datos y la extracción de atributos temporales de la variable Fecha de Registro. Estos incluyen año, mes, día, día de la semana, número de semana e indicador de fin de semana, y se utilizaron posteriormente para el análisis descriptivo y la construcción de los modelos SARIMA, Prophet y XGBoost.

4.4.1. Estadísticos descriptivos.

Como primera actividad del análisis exploratorio, se construyó la serie temporal diaria a partir de los tickets registrados. Para este propósito, se categorizó las solicitudes por fecha de registro y se asignó 0 a los días sin actividad para asegurar que la serie siguiera un orden cronológico. Esto se debe a que los días sin tickets constituyen información válida sobre el proceso de trabajo del servicio de atención y no omiten los registros correspondientes.

A continuación, se calculó las estadísticas descriptivas de la serie temporal para comprender el comportamiento general de la demanda. El notebook reportó un período de análisis de 910 días, que abarca desde enero de 2023 hasta julio de 2025, con un promedio de 13.39 tickets por día y un máximo de 96 tickets registrados en un solo día. Estos resultados indican una variabilidad significativa en la carga de trabajo y la importancia de modelos capaces de capturar la demanda fluctuante en el tiempo.

Figura 7 Serie temporal de tickets diarios registrados (enero de 2023 – julio de 2025).

```
# Construir la serie diaria de tickets
tickets_diarios = df_atributos.groupby(df_atributos['Fecha de registro'].dt.date).size()
tickets_diarios.index = pd.to_datetime(tickets_diarios.index)
tickets_diarios = tickets_diarios.asfreq('D').fillna(0)

# Gráfico de la serie temporal completa
fig, ax = plt.subplots(figsize=(15, 5))
ax.plot(tickets_diarios.index, tickets_diarios.values, color='steelblue', linewidth=0.7)
ax.set_title('Serie temporal: Tickets diarios registrados (Enero 2023 - Julio 2025)', fontsize=13)
ax.set_xlabel('Fecha')
ax.set_ylabel('Número de tickets')
ax.grid(alpha=0.3)
plt.tight_layout()
plt.show()

# Estadísticas básicas
print(f"Promedio diario:    {tickets_diarios.mean():.2f} tickets")
print(f"Mediana:            {tickets_diarios.median():.2f}")
print(f"Desviación estándar: {tickets_diarios.std():.2f}")
print(f"Máximo:             {tickets_diarios.max():.0f}")
print(f"Mínimo:             {tickets_diarios.min():.0f}")
```

Fuente: Elaboración propia a partir del notebook

4.4.2. Distribuciones.

Las distribuciones se utilizaron para determinar la composición de la recopilación de datos e identificar aquellas con alta demanda de soporte. Primero, se evalúo la distribución de días sin solicitudes y se encontró 295 días sin tickets, lo que significa el 32.42% de los datos. De estos, alrededor del 85% corresponden a sábados y domingos;

por lo tanto, la falta de registros en esos días se debe al calendario operativo de la organización y no a la captura de datos. Por esta razón, el notebook indica que estos registros deben registrarse en la serie temporal, ya que se trata de un aspecto estructural del negocio.

Ahora se considera la distribución de las categorías de solicitudes. Los resultados muestran que cinco categorías en conjunto representan el 97% del volumen total de tickets.

Categorías:

- */Computadora/Otras Solicitudes* cubre el 65% de los registros;
- /Computador, el 15%;
- /Computadora/Configuración, el 8%;
- */Computadora/Instalación_desinstalación_actualización*, el 7%;
- */Vinculación – Promoción*, el 2%.

Las 63 categorías restantes sirven cada una para menos del 2% del total, lo cual es muy significativo al considerar un número reducido de servicios.

Las prioridades también estaban muy desequilibradas. La prioridad P2 representa el 83% de los tickets registrados, y la P3, el 12%. Todas las demás son mucho menores. El notebook mismo indica que la razón principal es que P2 es la prioridad predeterminada del sistema y que clasifica la mayoría de las solicitudes generadas durante la operación normal del servicio de asistencia.

Figura 8 Distribución de tickets por categoría.

```
Total de categorías distintas: 69

--- Top 15 categorías por volumen ---
Categoría
/Computador/Otros Requerimientos                       7915
/Computador                                            1821
/Computador/Configuración                              1005
/Computador/Instalación_desinstalación_actualización    838
/Vinculación - Promoción                                242
/Computador/Nueva computadora                           154
/Acceso a Aplicaciones                                   27
/Puntos de Red                                           17
/Computador/Requerimientos Masivos                       16
/Asesores                                                14
/Reclamos y Requerimientos                               11
/Logiflow                                                11
/Cardholder                                               7
/Instalación ATM                                          7
/Páginas web - Proxy                                      6
/Eflow                                                    5
/Bizagi                                                   4
```

Fuente: Elaboración propia a partir del notebook

Figura 9 Distribución de tickets por prioridad.

```
# Volver a empezar desde df original
df_limpio = df.copy()

# Lista de prioridades válidas (incluyendo N/A que el sistema asigna a tickets sin prioridad)
prioridades_validas = ['P0', 'P1', 'P2', 'P3', 'P4', 'P5', 'N/A']

# Identificar las filas con datos desplazados (cualquier cosa que NO sea válida ni vacía)
filas_desplazadas = df_limpio[~df_limpio['Prioridad'].isin(prioridades_validas + [''])]

print(f"Filas con datos desplazados a descartar: {len(filas_desplazadas)}")
print(filas_desplazadas[['Código', 'Prioridad']])

# Descartar las filas desplazadas
codigos_a_descartar = filas_desplazadas['Código'].tolist()
df_limpio = df_limpio[~df_limpio['Código'].isin(codigos_a_descartar)].copy()

# Normalizar las filas con prioridad vacía a "Sin prioridad"
df_limpio.loc[df_limpio['Prioridad'] == '', 'Prioridad'] = 'Sin prioridad'

# Verificar resultado
print(f"\nFilas en df original:  {len(df)}")
print(f"Filas en df_limpio:    {len(df_limpio)}")
print(f"\nDistribución de prioridad:")
print(df_limpio['Prioridad'].value_counts())
```

```
Filas con datos desplazados a descartar: 2
                                                  Código  \
8689   INC 2023-073452,14-12-2023 17:36:41,P3,Error d...
11283  INC 2023-018898,12/5/2023 14:28,P0,Error de co...

                                               Prioridad
8689   por lo tanto se procederá a realizar una re-pr...
11283   y el excel se vuelve sumamente lento y se cierra
```

Fuente: Elaboración propia a partir del notebook

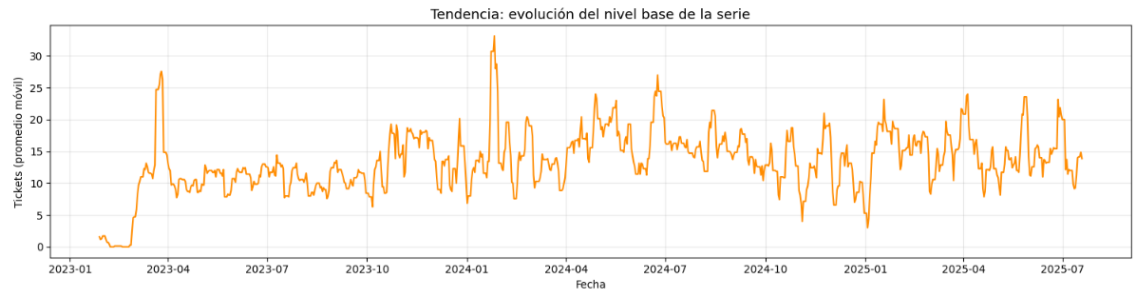
4.4.3. Tendencia.

El análisis de tendencias se ha realizado para determinar si el número de solicitudes ha cambiado sistemáticamente con el tiempo. El resumen de EDA del notebook muestra que la serie temporal presenta una tendencia al alza. Se estima que hay 10 tickets por día en 2023 y de 20 a 25 tickets diarios en 2025 como línea base. Esta tendencia indica el crecimiento operativo de la organización y la demanda de soporte tecnológico.

Que este patrón semanal sea no estacionario de naturaleza estacional también es importante y esto es algo a considerar durante la construcción del modelo SARIMA. Se necesita diferenciación estacional para tener en cuenta este comportamiento y hacer que el modelo sea capaz de explicar el ciclo semanal en la serie temporal.

Por lo tanto, el análisis de tendencia y estacionalidad tendrá información valiosa para la próxima configuración del modelo SARIMA.

Gráfico 2 Componente de tendencia de la serie temporal.



Fuente: Elaboración propia a partir del notebook

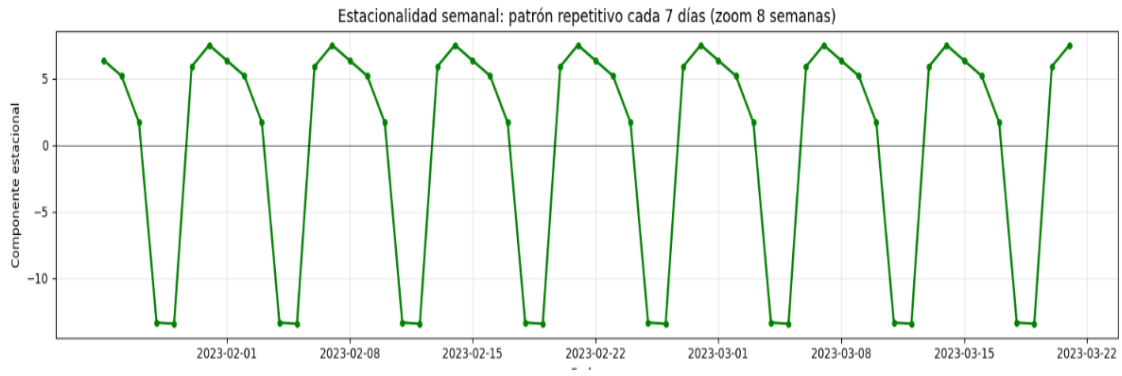
4.4.4. Estacionalidad.

El reconocimiento de patrones estacionales es uno de los objetivos más importantes del análisis exploratorio, ya que incide directamente en los modelos predictivos. Los resultados muestran dos patrones estacionales claramente diferenciados.

El primero corresponde a una estacionalidad semanal, que se consideró el componente principal de la serie. El martes es el día con mayor demanda en promedio (20.91 tickets por día) y los fines de semana son prácticamente nulos (0.06 tickets por día). La amplitud de este componente es de aproximadamente 19 tickets, y el calendario laboral es la principal causa de la variabilidad de la demanda.

El segundo corresponde a una estacionalidad anual de menor intensidad. Los meses de febrero y diciembre tienen las solicitudes promedio más bajas, mientras que enero, abril, mayo y noviembre están por encima del promedio total. Estos resultados muestran que la demanda está influida no solo por el ciclo operativo semanal, sino también por la variación del calendario anual.

Gráfico 3 Componente estacional de la serie temporal.



Fuente: Elaboración propia a partir del notebook

4.4.5. Visualizaciones.

Las visualizaciones del análisis exploratorio confirmaron los hallazgos obtenidos mediante las estadísticas descriptivas y permitieron comprender el comportamiento temporal de los requerimientos. En particular, la descomposición aditiva de la serie permitió separar la tendencia, la estacionalidad y los componentes residuales, lo que mostró que la mayor parte de la estructura temporal puede explicarse por una tendencia creciente y una estacionalidad semanal estable.

Se encontró que el comportamiento residual es aproximadamente simétrico en torno a cero y que estos eventos corresponden a picos de operaciones extraordinarias. El notebook muestra claramente que estos eventos no pueden explicarse por el comportamiento pasado de la serie y constituyen la principal fuente de error en los modelos predictivos. Por lo tanto, se concluye que los modelos podrán predecir con precisión el comportamiento de los días normales y que los eventos excepcionales requerirán variables adicionales (como horarios de mantenimiento o eventos organizacionales) que no se consideran aquí.

La descomposición aditiva con periodicidad semanal (s=7) confirmó:

- **Tendencia creciente** no estacionaria → requiere diferenciación
- **Estacionalidad semanal fuerte y estable** → componente principal
- **Residuo simétrico alrededor de cero** con picos puntuales correspondientes a eventos operativos no predecibles desde la serie

4.5. Desarrollo de los modelos predictivos

4.5.1. SARIMA.

El primer modelo desarrollado fue el de Promedio Móvil Integrado Autoregresivo Estacional (SARIMA), ya que el análisis exploratorio mostró la presencia de una tendencia y una marcada estacionalidad semanal en la serie temporal.

Como paso preliminar del entrenamiento, el notebook verificó la estacionariedad de la serie mediante la prueba de Dickey-Fuller aumentada. Los resultados llevaron a la conclusión de que no era necesario el diferenciado regular (d = 0), pero sí el diferenciado estacional (D = 1) para capturar el ciclo semanal de 7 días. Esta decisión metodológica se anotó explícitamente en el notebook antes del análisis de autocorrelación.

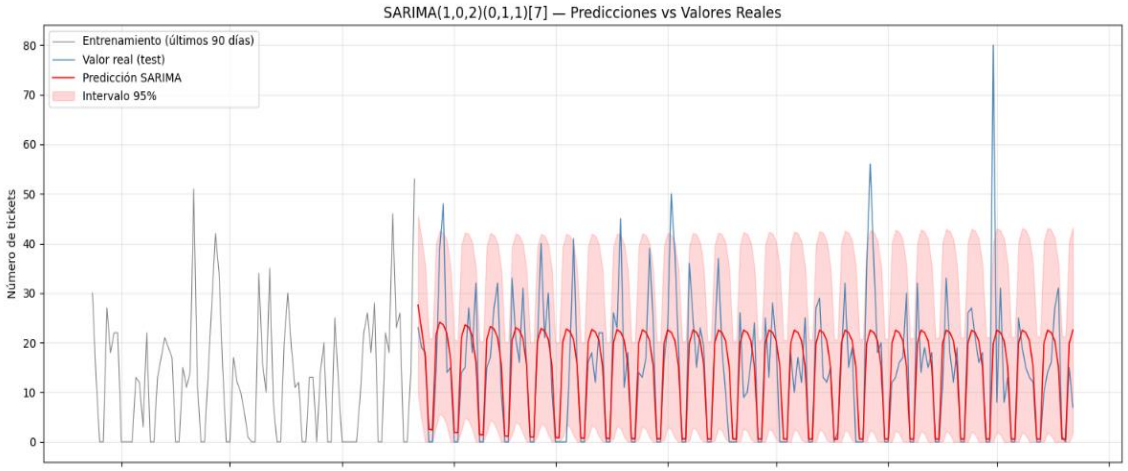
Luego, se analizaron las funciones de autocorrelación (ACF) y autocorrelación parcial (PACF) para identificar la dependencia temporal de las observaciones y luego determinar el orden autorregresivo y de promedio móvil del modelo.

El siguiente paso fue buscar automáticamente los parámetros mediante la función auto_arima de la biblioteca pmdarima. La función evaluó diferentes combinaciones de parámetros del modelo a través de una búsqueda escalonada y utilizó el Criterio de Información de Akaike (AIC) para identificar la mejor configuración. En esta búsqueda, d = 0 porque se encontró que la prueba aumentada de Dickey-Fuller no era necesaria para la diferenciación regular; D = 1 para la diferenciación estacional; y s = 7 para la periodicidad semanal de la serie. También se consideraron los valores p y q entre 0 y 3; P y Q entre 0 y 2.

Como resultado de este procedimiento, se seleccionó el modelo SARIMA(1,0,2)(0,1,1)[7]. (1,0,2) representa las partes no estacionales del modelo: p = 1 indica que hay un término autorregresivo en los datos y, como tal, el valor actual de la serie depende de los valores anteriores de los datos; d = 0 indica que no se realizó diferenciación regular; y q = 2 incluye dos términos de promedio móvil de los errores de predicción anteriores. (0,1,1) representa las partes estacionales: P = 0 significa que no hay un término autorregresivo estacional en los datos; D = 1 significa que hay una diferenciación estacional; Q = 1 incluye un término de promedio móvil estacional; y [7] especifica un período de siete días, que es el patrón semanal observado en los requisitos.

En la práctica, esta configuración permite que el modelo considere tanto la dependencia temporal reciente del número de requisitos como el patrón de comportamiento que se repite semanalmente. Por esta razón, se utiliza SARIMA(1,0,2)(0,1,1)[7] para el entrenamiento y la predicción. Una vez ajustado el modelo, se generaron predicciones en el conjunto de prueba y se compararon con los valores reales. SARIMA reprodujo correctamente el patrón de demanda semanal y mostró un comportamiento estable ante variaciones normales en la serie, aunque tiende a suavizar los picos de demanda extraordinarios.

Gráfico 4 Predicciones del modelo SARIMA sobre el conjunto de prueba.



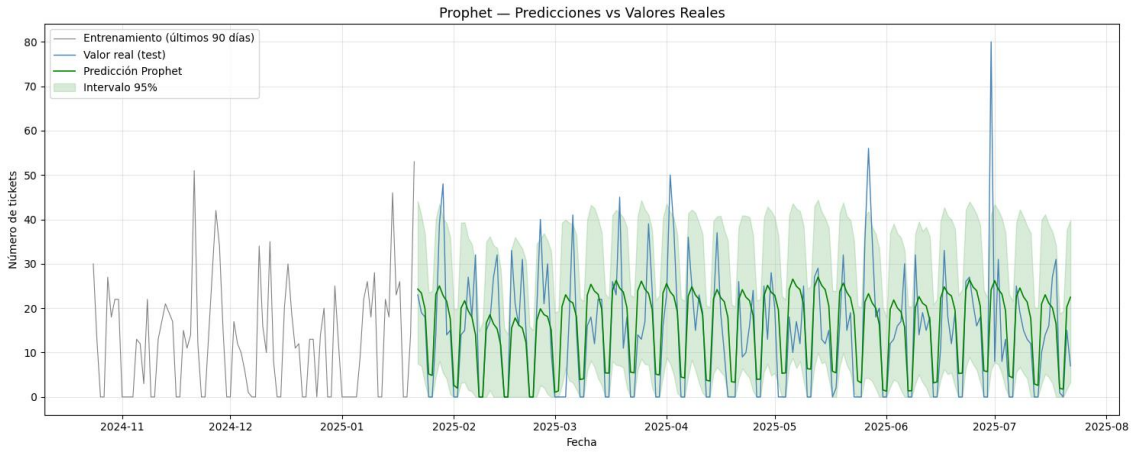
Fuente: Elaboración propia a partir del notebook

4.5.2. Prophet.

El segundo modelo fue Prophet, un modelo desarrollado por Meta para predecir series temporales con tendencias y componentes estacionales. Según los comentarios en el notebook, se seleccionó porque Prophet es capaz de modelar automáticamente tendencias y estacionalidades sin necesidad de especificar manualmente los parámetros estacionales. Luego, el conjunto de datos se entrenó con las columnas ds (fecha) e y (número de boletos diarios) como lo requiere esta biblioteca. Para predecir las fechas del período de prueba, se utilizó make_future_dataframe() para generar las predicciones. Nuevamente, se usó valores no negativos para mantener el problema consistente.

Se encontró que Prophet reprodujo con precisión el comportamiento general de la serie temporal y la estacionalidad semanal. El notebook indica que el rendimiento de Prophet no fue tan bueno como el rendimiento de SARIMA en las tres métricas principales (MAE, RMSE y MAPE). La razón de esto es que la temporada era muy regular y había pocos eventos especiales presentes, y un modelo estadístico más especializado como SARIMA es mucho más eficiente. Sin embargo, Prophet es una buena opción para la previsión de categorías porque funciona bien cuando hay menos datos históricos.

Gráfico 5 Predicciones del modelo Prophet sobre el conjunto de prueba.



Fuente: Elaboración propia a partir del notebook

El gráfico muestra los últimos 90 días del entrenamiento (gris), los valores reales del conjunto de prueba (azul), las predicciones del modelo Prophet (verde) y el intervalo de confianza del 95% (banda verde).

- **Hallazgos visuales:**
  - **Captura del patrón semanal:** Prophet reproduce correctamente el ciclo semanal, con picos en los días laborables y valles en los fines de semana. La fidelidad del patrón es comparable a la de un modelo SARIMA.
  - **Detección de la tendencia creciente:** se observa que las predicciones de Prophet aumentan progresivamente hacia los meses finales del test (mayo-julio 2025), lo que refleja la tendencia alcista identificada en el EDA. Esta es una ventaja teórica de Prophet sobre SARIMA, que asume un nivel medio constante en sus pronósticos a futuro. Sin embargo, este beneficio no fue suficiente para superar las métricas globales de SARIMA.
  - **Subestimación moderada de los picos:** los picos predichos alcanzan valores cercanos a 25-27 tickets, frente a picos reales de 40-50 tickets. Al igual que SARIMA, Prophet predice el valor más probable y no los extremos.
  - **Intervalo de confianza más amplio:** la banda del 95% de Prophet es visiblemente más ancha que la de SARIMA, lo que refleja una mayor incertidumbre en sus pronósticos. Esto puede ser beneficioso para la planificación operativa, ya que cuantifica de manera más conservadora el rango de demanda posible.

- **Eventos extraordinarios no capturados:** el pico de 80 tickets del 1 de julio queda completamente fuera del intervalo de confianza, lo que confirma que los outliers operativos no son predecibles a partir de la serie temporal con ninguno de los modelos evaluados.

4.5.3. XGBoost.

El tercer modelo desarrollado fue XGBoost, un algoritmo de aprendizaje automático basado en árboles de decisión mejorado mediante el boosting por gradiente. A diferencia de SARIMA y Prophet, este modelo requiere construir variables explicativas a partir de la serie temporal. Como se describe en el notebook, primero se construyó un conjunto de 12 variables predictoras relacionadas con el calendario y con los rezagos de la serie temporal. Luego, con el objetivo de mejorar el rendimiento del modelo, se añadieron 25 variables: variables cíclicas (funciones seno y coseno del día de la semana y del mes), rezagos adicionales, así como estadísticas móviles (desviación estándar, máximos, mínimos y promedios móviles), lo que permitió una descripción más rica de la dinámica de las series y mejoró el poder predictivo del modelo.

Se utilizó la optimización de hiperparámetros para mejorar el rendimiento del modelo XGBoost. La configuración fue de 200 estimadores (n_estimators = 200), profundidad máxima de 3 (max_depth = 3), tasa de aprendizaje de 0.03 (learning_rate = 0.03), submuestreo de observaciones de 0.85 (subsample = 0.85), proporción de características por árbol de 0.70 (colsample_bytree = 0.70) y peso mínimo de hijo de 1 (min_child_weight = 1). Se considera los parámetros subsample y colsample_bytree por separado porque el primero controla la proporción de observaciones utilizadas en la construcción de cada árbol, mientras que el segundo controla la proporción de características. Esta configuración se utilizará para hacer las predicciones y evaluar su rendimiento.

También señala que las variables basadas en el calendario y las estadísticas móviles permitieron al modelo representar mejor los picos de demanda moderados, lo que constituye una ventaja frente a un modelo puramente estadístico.

Figura 10 Optimización de hiperparámetros y predicción con el modelo XGBoost.

```
from sklearn.model_selection import TimeSeriesSplit, RandomizedSearchCV

# Espacio de búsqueda de hiperparámetros
param_grid = {
    'n_estimators': [200, 500, 800, 1000],
    'max_depth': [3, 5, 7, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.85, 1.0],
    'colsample_bytree': [0.7, 0.85, 1.0],
    'min_child_weight': [1, 3, 5]
}

# Validación cruzada temporal (respeta el orden cronológico)
tscv = TimeSeriesSplit(n_splits=5)

# Modelo base
modelo_base = XGBRegressor(
    objective='reg:squarederror',
    random_state=42,
    n_jobs=-1
)

# Búsqueda aleatoria de hiperparámetros
print("Buscando los mejores hiperparámetros... (puede tardar 3-7 minutos)\n")

busqueda = RandomizedSearchCV(
    modelo_base,
    param_distributions=param_grid,
    n_iter=30,                          # número de combinaciones a probar
    cv=tscv,
    scoring='neg_mean_absolute_error',  # minimizar MAE
    verbose=1,
    n_jobs=-1,
    random_state=42
`


 busqueda.fit(X_train, y_train)

 print(f"\n{'='*60}")
 print("MEJORES HIPERPARÁMETROS ENCONTRADOS")
 print(f"{'='*60}")
 for param, valor in busqueda.best_params_.items():
     print(f"  {param}: {valor}")
 print(f"\nMAE promedio en validación cruzada: {-busqueda.best_score_:.4f}")
```

```
Buscando los mejores hiperparámetros... (puede tardar 3-7 minutos)

Fitting 5 folds for each of 30 candidates, totalling 150 fits

============================================================
MEJORES HIPERPARÁMETROS ENCONTRADOS
============================================================
  subsample: 0.85
  n_estimators: 500
  min_child_weight: 1
  max_depth: 7
  learning_rate: 0.01
  colsample_bytree: 0.7

MAE promedio en validación cruzada: 5.8898
```

```
# Entrenar el modelo final con los mejores hiperparámetros
modelo_xgb_optimo = busqueda.best_estimator_
```

Fuente: Elaboración propia a partir del notebook

4.6. Evaluación y comparación de modelos

Una vez entrenados los modelos SARIMA, Prophet y XGBoost, las predicciones se evaluaron en el conjunto de prueba. Según el plan de investigación, se evalúo con las métricas que se utilizan a menudo en problemas de pronóstico de series temporales, y estas se emplearon para determinar la diferencia entre los valores reales y las predicciones de cada modelo.

El notebook explica que la comparación se realizó utilizando las métricas Error Absoluto Medio (MAE), Raíz del Error Cuadrático Medio (RMSE), Error Porcentual Absoluto Medio (MAPE) y Error Porcentual Absoluto Ponderado (WAPE). Estas métricas permitieron evaluar los modelos desde diferentes ángulos: error absoluto promedio, penalización de errores grandes, error porcentual y error ponderado en función del volumen total de la demanda.

4.6.1. MAE

Se utilizó el error absoluto medio (MAE) para evaluar la diferencia promedio en el número de boletos entre los valores observados y las predicciones de cada modelo. Esta métrica permite interpretar el error al situarlo en la misma unidad que la variable en estudio.

Los resultados muestran que el XGBoost optimizado tuvo el MAE más bajo (6.14 boletos) y SARIMA(1,0,2)(0,1,1)[7] estuvo muy cerca de él (6.23 boletos). Prophet tuvo el error absoluto medio más alto (7.17 boletos), lo que refleja un nivel de precisión inferior en esta serie temporal.

La diferencia entre SARIMA y XGBoost es inferior a una décima de boleto por día, lo que indica que ambos modelos representan adecuadamente el comportamiento general de la serie. Sin embargo, XGBoost se benefició de variables del calendario, rezagos y estadísticas móviles.

Figura 11 Comparación del MAE obtenido por los modelos SARIMA, Prophet y XGBoost

```
# Construir la tabla comparativa con todas las métricas
tabla_comparativa = pd.DataFrame([metricas_sarima, metricas_prophet, metricas_xgb_optimo])
tabla_comparativa = tabla_comparativa.set_index('modelo')

# Identificar el mejor modelo por cada métrica
mejores = {col: tabla_comparativa[col].idxmin() for col in tabla_comparativa.columns}

print("-" * 60)
print("TABLA COMPARATIVA FINAL DE MODELOS")
print("-" * 60)
print(tabla_comparativa.round(2))

print("\n" + "-" * 60)
print("GANADOR POR MÉTRICA")
print("-" * 60)
for metrica, ganador in mejores.items():
    print(f"  {metrica}: {ganador}")
```

```
------------------------------------------------------------
TABLA COMPARATIVA FINAL DE MODELOS
------------------------------------------------------------
                          MAE  RMSE   MAPE   WAPE
modelo
SARIMA(1,0,2)(0,1,1)[7]  6.23  9.56  40.94  41.90
Prophet                  7.17  9.99  45.20  48.16
XGBoost optimizado       6.14  9.46  43.68  41.00

------------------------------------------------------------
GANADOR POR MÉTRICA
------------------------------------------------------------
  MAE: XGBoost optimizado
  RMSE: XGBoost optimizado
  MAPE: SARIMA(1,0,2)(0,1,1)[7]
  WAPE: XGBoost optimizado
```

Fuente: Elaboración propia a partir del notebook

4.6.2. RMSE.

Se utilizó la métrica RMSE para evaluar el impacto de errores de mayor magnitud, ya que penaliza más fuertemente las diferencias significativas entre los valores observados y los predichos. Esto nuevamente muestra el mejor rendimiento del XGBoost optimizado con 9.46 tickets, del SARIMA con 9.56 tickets y del Prophet con 9.99 tickets. Estos resultados demuestran que los tres modelos son capaces de reproducir adecuadamente el comportamiento general de la serie, mientras que XGBoost presenta una ligera ventaja al aproximar mejor los picos de demanda moderados. (ver Figura 12).

4.6.3. MAPE.

La medida MAPE calcula el porcentaje de error entre las predicciones y los valores reales, lo que permite comparar el rendimiento de los modelos sin importar el tamaño de la serie. A diferencia de los números anteriores, el valor más bajo fue obtenido por

SARIMA(1,0,2)(0,1,1)[7], con un MAPE de 40.94%, mientras que el XGBoost, con mejor rendimiento, tuvo un MAPE de 43.68% y Prophet tuvo el mayor porcentaje de error (45.20%). Esto significa que SARIMA es más preciso en términos de la estructura porcentual de la serie temporal, debido a la marcada estacionalidad semanal observada en el análisis exploratorio (ver Figura 12).

4.6.4. Comparación.

Con el propósito de seleccionar el modelo más adecuado para la estimación de requerimientos, se compararon las métricas obtenidas por cada uno de los algoritmos evaluados.

Tabla 1 Resume los resultados obtenidos.

Modelo	MAE	RMSE	MAPE	WAPE
SARIMA (1,0,2) (0,1,1) [7]	**6,23**	9,56	**40,94 %**	41,90 %
Prophet	7,17	9,99	45,20 %	48,16 %
XGBoost optimizado	**6,14**	**9,46**	43,68 %	**41,00 %**

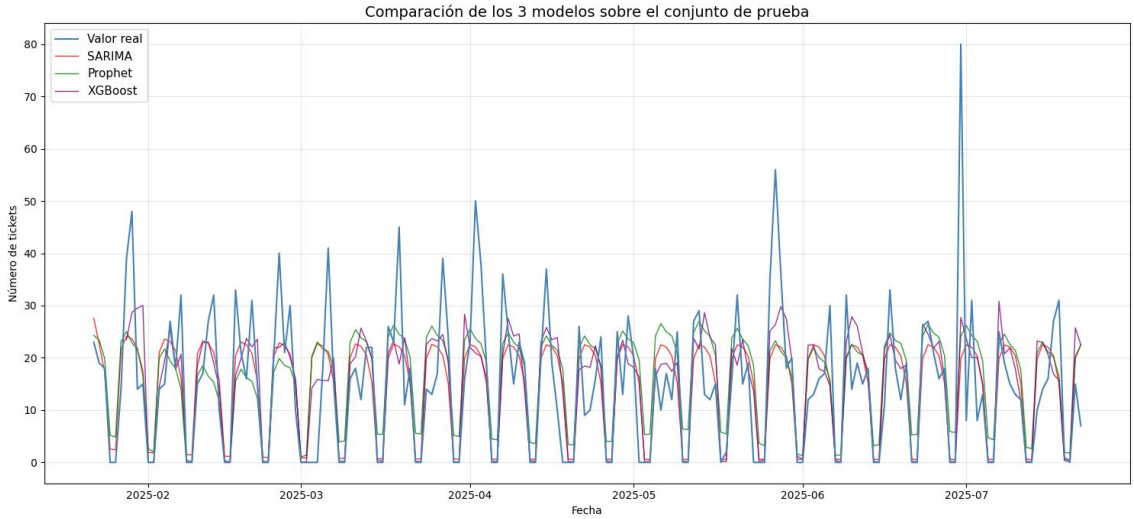
Fuente: Elaboración propia a partir del notebook

Los resultados muestran que XGBoost optimizado fue el mejor en general, ya que presentó los MAE, RMSE y WAPE más bajos, mientras que SARIMA obtuvo el MAPE más bajo. Prophet mostró los valores más altos en las cuatro métricas evaluadas, lo que indica un rendimiento deficiente en series temporales.

El análisis cualitativo en el notebook también indica que los tres modelos capturaron perfectamente la tendencia y la estacionalidad semanal, pero todos mostraron dificultades para anticipar eventos extraordinarios con demandas de entre 45 y 80 boletos diarios, ya que no pueden inferirse únicamente del comportamiento histórico de la serie.

Además, el notebook muestra que XGBoost representó picos de demanda moderados de manera más amplia, que SARIMA mostró un comportamiento más estable e interpretable y que Prophet aún tuvo un rendimiento medio, aunque capturó bien la fuerte caída de los fines de semana.

Gráfico 6 Comparación de las predicciones generadas por SARIMA, Prophet y XGBoost.



Fuente: Elaboración propia a partir del notebook

4.6.5. Selección del mejor modelo.

Según los resultados obtenidos, el modelo XGBoost optimizado es el que presenta el mejor rendimiento general para predecir el número diario de solicitudes en el servicio de asistencia. Esto se debe a que el modelo obtuvo los menores valores de MAE, RMSE y WAPE, lo que indica que es más preciso en términos de error absoluto promedio y de error acumulado en comparación con el número total de tickets. También ha utilizado variables del calendario, desfases temporales y estadísticas móviles, así como la optimización de hiperparámetros mediante RandomizedSearchCV y TimeSeriesSplit para representar mejor la dinámica de la serie temporal. También se encontró que SARIMA es muy competitivo y presenta el MAPE más bajo, lo que indica que la serie temporal tiene una estructura temporal claramente definida y una estacionalidad semanal.

Por lo tanto, la diferencia entre ambos modelos es mínima y la decisión final se basará en las necesidades de la organización. Si la prioridad es la precisión del pronóstico y la posibilidad de incorporar nuevas variables explicativas, el XGBoost optimizado es el más adecuado. Por otro lado, si el enfoque está en la interpretabilidad del modelo y la explicación de la estructura de la serie temporal, SARIMA es igualmente adecuado. En el notebook, ambos modelos se consideran la mejor opción para el problema estudiado.

5. Análisis de Resultados

5.1. Análisis de la preparación de los datos

5.1.1. Evaluación de la calidad del conjunto de datos

La calidad de los datos es un factor importante para garantizar la fiabilidad de un modelo predictivo. En la investigación de series temporales, los registros incompletos, los formatos inconsistentes o cualquier error en la estructura de los datos pueden afectar en gran medida tanto el proceso de identificación de patrones como la capacidad predictiva del modelo. Por lo tanto, al evaluar la calidad de los datos en este contexto, los datos históricos del servicio de asistencia técnica pudieron utilizarse para el análisis exploratorio, el entrenamiento y la validación de los modelos predictivos.

El conjunto de datos estudiado se basa en los registros históricos de solicitudes del servicio de asistencia técnica de TI desde el 25 de enero de 2023 hasta el 22 de julio de 2025. Cabe destacar que el notebook también señala que, al inicio del análisis, el archivo exportado del sistema de gestión de tickets fue revisado minuciosamente para determinar su estructura, identificar cualquier inconsistencia y verificar la integridad de la información. Esto se hizo para identificar cualquier diferencia en el formato de almacenamiento, los caracteres especiales y los diferentes formatos de fecha que impedirían que los registros fueran procesados automáticamente sin limpieza.

La inspección inicial también reveló que la información contenía un gran número de atributos relacionados con el proceso de gestión de solicitudes. Sin embargo, no todas las variables resultaron relevantes para el objetivo de la investigación. Por lo tanto, al preparar los datos, solo se utilizaron las variables necesarias para el análisis temporal, con el fin de reducir la complejidad del conjunto de datos y eliminar información redundante que no contribuía al modelo de predicción. Este enfoque permitió centrarnos en las variables que describen la fecha de registro, la categoría de la solicitud y otros atributos asociados al comportamiento temporal de la demanda.

Uno de los principales hallazgos de la evaluación de calidad es la consistencia de la información tras la limpieza. Una vez corregidos los problemas de formato y normalizadas las variables de fecha, el conjunto de datos estaba compuesto por 12,191 registros distribuidos en 12 variables, preservando toda la información necesaria para el estudio. Este hecho demuestra que las inconsistencias se debieron principalmente al

formato de exportación del archivo y no a una pérdida significativa de información durante el registro de las solicitudes.

Tabla 2 Resumen de la calidad y depuración del conjunto de datos

Característica evaluada	**Resultado obtenido**
Registros correctamente estructurados	12.191
Variables iniciales	12
Registros con datos desplazados identificados	2
Registros posteriores a la depuración estructural	12.189
Variable temporal principal	Fecha de registro
Variable categórica analizada	Categoría

Fuente: Elaboración propia a partir del notebook.

La tabla 2 resume las principales verificaciones que se realizaron antes y después de la evaluación de la calidad de los datos. El procedimiento de segunda lectura permitió verificar la estructura correcta de 12,191 registros, inicialmente distribuidos en 12 variables. Dos registros con datos desplazados entre columnas fueron descartados en verificaciones posteriores porque no seguían la estructura que se esperaba. Después de la limpieza, se obtuvieron 12,189 registros para continuar con las etapas posteriores de preparación y transformación de datos.

Otro aspecto interesante de la evaluación fue la ausencia de días sin registros de tickets. El análisis del notebook mostró que estos días corresponden principalmente a los fines de semana y no presentan valores faltantes ni errores de captura. Por lo tanto, se mantuvo los días sin solicitudes en la serie temporal en cero, de modo que se pudo transmitir correctamente el comportamiento operativo de la organización y no introducir sesgos en el entrenamiento de los modelos.

Figura 12 Construcción de la serie temporal continua a partir de los registros históricos.

```
# Crear nuevo DataFrame para la fase de modelado
df_modelo = df_atributos.copy()

# Construir la serie diaria final
serie_diaria = df_modelo.groupby(df_modelo['Fecha de registro'].dt.date).size()
serie_diaria.index = pd.to_datetime(serie_diaria.index)
serie_diaria = serie_diaria.asfreq('D').fillna(0)
serie_diaria.name = 'tickets'

# División train/test (80% / 20%)
# Debido a que es serie temporal no sse hace shuffle
punto_corte = int(len(serie_diaria) * 0.8)
train = serie_diaria.iloc[:punto_corte]
test = serie_diaria.iloc[punto_corte:]

print(f"Total de días:      {len(serie_diaria)}")
print(f"Días entrenamiento: {len(train)} ({len(train)/len(serie_diaria)*100:.1f}%)")
print(f"Días prueba:        {len(test)} ({len(test)/len(serie_diaria)*100:.1f}%)")
print(f"\nFecha inicio train: {train.index[0].date()}")
print(f"Fecha fin train:    {train.index[-1].date()}")
print(f"Fecha inicio test:  {test.index[0].date()}")
print(f"Fecha fin test:     {test.index[-1].date()}")
```

```
Total de días:      910
Días entrenamiento: 728 (80.0%)
Días prueba:        182 (20.0%)

Fecha inicio train: 2023-01-25
Fecha fin train:    2025-01-21
Fecha inicio test:  2025-01-22
Fecha fin test:     2025-07-22
```

Fuente: Elaboración propia a partir del notebook

La Figura 13 ha permitido verificar que la serie temporal fue generada en el orden cronológico de los registros y que los datos son consistentes entre sí en las observaciones. Esto es importante para los problemas de pronóstico porque se quiere que los modelos aprendan la evolución real de la demanda a lo largo del tiempo, no una secuencia fragmentada de eventos. Desde el punto de vista analítico, el análisis de calidad de los datos mostró que el proceso de preparación de la investigación no solo hizo más consistentes las estructuras, sino que también hizo que los resultados del análisis fueran más confiables.

La disponibilidad de la serie temporal en orden completo y consistente y sin errores permitió identificar los patrones de tendencia y estacionalidad, y los modelos SARIMA, Prophet y XGBoost fueron entrenados en el mismo escenario. Finalmente, los resultados han llevado a concluir que la calidad de los datos fue suficiente para desarrollar el modelo predictivo propuesto.

La consistencia lograda durante el proceso de preparación eliminó el riesgo de errores en el procesamiento de la información y sentó las bases sólidas para los modelos

considerados en esta investigación. Así, la calidad de los datos es uno de los factores que influyen en el rendimiento de los modelos predictivos y en el cumplimiento de los objetivos establecidos en el estudio.

5.1.2. Análisis del proceso de limpieza

El proceso de limpieza de datos fue un paso crucial para garantizar la calidad de la información utilizada en el entrenamiento de modelos predictivos. Se encontró que los principales problemas con el conjunto de datos no eran la falta de información, sino más bien las inconsistencias derivadas del proceso de exportación del sistema de gestión de tickets. Estas inconsistencias impedían que las herramientas de análisis de datos interpretaran correctamente el archivo, por lo que se necesitaba realizar una etapa de limpieza previa antes de comenzar la modelización.

Según los comentarios en el notebook, la primera lectura del archivo permitió observar la estructura distinta de los registros, así como los caracteres especiales y las fechas. Por lo tanto, leer el archivo mediante métodos de carga convencionales no producía un conjunto de datos coherente. Por lo tanto, se analizaron diferentes estrategias de lectura para la limpieza, a fin de encontrar una que preservara la estructura original de los registros. Este enfoque permitió recuperar toda la información relevante para el estudio y evitar la pérdida de datos durante la importación.

Figura 13 Evaluación de los métodos de lectura del conjunto de datos.

```
# Lista vacía para guardar las líneas limpias del intento 1
cleaned_lines_v1 = []
# Limpieza ligera: solo eliminar los ';;;' del final
for line in uploaded[nombre_archivo].decode("latin-1").splitlines():
    cleaned_line = line.strip()
    if cleaned_line.endswith(';;;'):
        cleaned_line = cleaned_line[:-3]
    cleaned_lines_v1.append(cleaned_line)
# Unir las líneas en un texto único
cleaned_data_v1 = "\n".join(cleaned_lines_v1)

# Cargar con pandas usando el motor de Python
df_intento1 = pd.read_csv(
    io.StringIO(cleaned_data_v1),
    sep=',',
    quotechar='"',
    engine='python',
    on_bad_lines='skip'
)
print(f"Filas cargadas:     {len(df_intento1)}")
print(f"Filas en archivo:   {len(cleaned_lines_v1) - 1}")
print(f"Filas descartadas:  {len(cleaned_lines_v1) - 1 - len(df_intento1)}")
```

```
Filas cargadas:     12193
Filas en archivo:   12357
Filas descartadas:  164
```

Fuente: Elaboración propia a partir del notebook

La Figura 13 muestra que la validación de diferentes métodos de lectura no solo fue una tarea técnica, sino también un medio para preservar los datos. El procedimiento de importación aseguró la preservación de todos los datos históricos y garantizó que las variables se observaran correctamente antes del trabajo exploratorio.

Después de cargar el archivo, se examinó los tipos de datos, los registros y la consistencia de las variables, y se encontró que el proceso de limpieza produjo una estructura de datos más consistente para la siguiente etapa del proyecto. Por ejemplo, la fecha de registro fue normalizada y todo el formato de fecha se unificó en un tipo de dato cronológico, lo que permitió construir la serie temporal y las variables derivadas que se utilizarán en modelos predictivos.

Una parte importante de la limpieza consistió en el manejo de valores nulos. El notebook mostró que las variables utilizadas para construir el modelo predictivo estaban completas. En este caso, no fue necesario utilizar métodos de imputación complejos, ya que la información disponible era suficiente para representar el comportamiento histórico de los requisitos. Tal condición redujo el riesgo de introducir sesgos derivados de la estimación artificial de datos faltantes y fortaleció la fiabilidad del proceso de modelado.

Tabla 3 Principales actividades desarrolladas durante el proceso de limpieza de datos

Actividad	**Resultado obtenido**	**Impacto en el análisis**
Validación de la estructura del archivo	Identificación de inconsistencias de formato	Permitió seleccionar el método de lectura adecuado
Normalización de fechas	Conversión a formato datetime	Facilitó la construcción de la serie temporal
Verificación de valores nulos	Variables de interés con información suficiente	Evitó procesos adicionales de imputación
Revisión de tipos de datos	Variables correctamente tipificadas	Garantizó la consistencia del análisis
Selección de variables	Conservación únicamente de los atributos relevantes	Redujo la complejidad del modelo

Fuente: Elaboración propia a partir del notebook

La Tabla 3 muestra que el proceso de limpieza no se limitó a errores de formato, sino que constituyó una etapa completa de validación del conjunto de datos. Todas las actividades estaban dirigidas a mejorar la consistencia de los datos y establecer la base de datos para el análisis predictivo.

La limpieza de datos también afectó el rendimiento de los modelos. Identificar correctamente las fechas, eliminar inconsistencias estructurales y probar la integridad de los registros permitieron construir una serie temporal continua y cronológicamente consistente. Así, los modelos SARIMA, Prophet y XGBoost fueron entrenados con la misma base de datos, y las diferencias en el rendimiento se debieron a las características de cada algoritmo y no a la calidad de los datos.

Finalmente, se concluye del proceso de limpieza que la preparación del conjunto de datos fue esencial para el éxito del proyecto. La información consistente permitió un análisis exploratorio confiable, el desarrollo de modelos predictivos sobre una base homogénea y los mismos resultados en la fase de evaluación. Por lo tanto, la limpieza de los datos es uno de los factores clave para mejorar la calidad del análisis y alcanzar los objetivos de la investigación.

5.1.3. Análisis de la transformación y construcción de la serie temporal

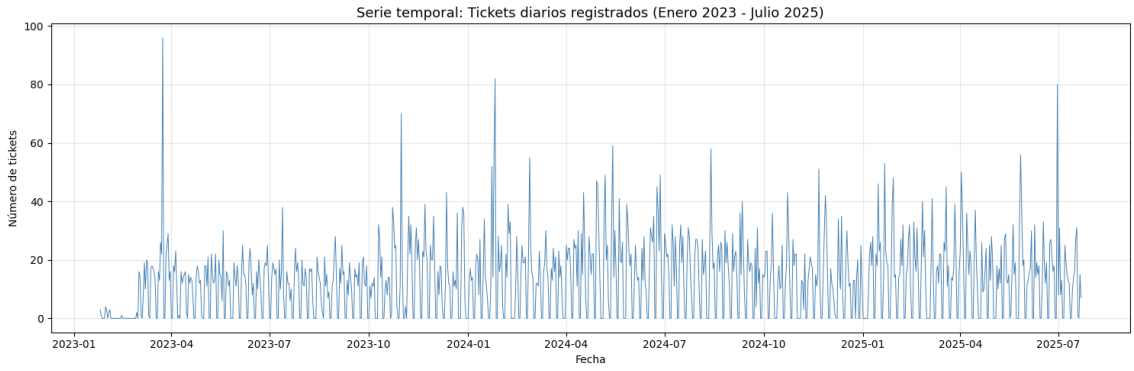
La transformación de datos fue un paso crucial en el desarrollo del modelo predictivo, ya que permitió convertir un conjunto de registros transaccionales del servicio de asistencia en una serie temporal estructurada que podía analizarse y utilizarse con algoritmos de pronóstico. Mientras que la fase de limpieza estaba destinada a asegurar la integridad y la consistencia de la información, la transformación de datos se dirigió a adaptar los datos a los modelos SARIMA, Prophet y XGBoost para mantener el orden en que ocurrieron los eventos registrados.

El análisis que se ha realizado muestra que el primer resultado importante de esta fase fue normalizar la variable "Fecha de registro" a un formato cronológico estandarizado. Esto llevó al ordenamiento de los registros y a una referencia temporal única para todas las observaciones. La correcta conversión de las fechas aclaró todas las ambigüedades de los diferentes formatos presentes en el archivo original y aseguró que cada requerimiento estuviera asociado al día exacto en que fue registrado. Se pudo aplicar la agregación temporal y crear una secuencia continua de observaciones, lo que constituye, por tanto, la clave para el modelado de series temporales.

Luego, transformaron cada registro mediante la agregación temporal de los requerimientos de cada día del período estudiado. Esto permitió reemplazar miles de registros por un único número que representa la demanda diaria del servicio de asistencia. Se descubrió que se podía identificar patrones de comportamiento a nivel global y reducir la complejidad del conjunto de datos sin perder información relevante para el propósito.

Una de las consideraciones clave en la creación de la serie temporal fue el tratamiento de los días no registrados. El notebook concluye que estos días corresponden mayormente a fines de semana y, por lo tanto, son solo parte de la actividad de la organización, no datos. Como resultado, se incluyó explícitamente estos días en la serie temporal como valores de cero. Esta decisión ayudó a mantener los datos en orden cronológico y a evitar que los modelos interpretaran la ausencia de registros como una ruptura de la serie. También permitió comprender mejor el patrón de comportamiento semanal observado durante el estudio exploratorio.

Gráfico 7 Construcción de la serie temporal diaria a partir de los registros históricos.



Fuente: Elaboración propia a partir del notebook

Promedio diario: 13.39 tickets

Mediana: 13.00

Desviación estándar: 12.96

Máximo: 96

Mínimo: 0

El gráfico 7 muestra el resultado del proceso de transformación, lo que indica que los registros individuales fueron consolidados en una serie temporal continua. Esta representación sirve como la principal entrada para las etapas subsecuentes de análisis exploratorio y modelado predictivo, ya que garantiza que la información conserve la

secuencia temporal necesaria para identificar tendencias, estacionalidad y variaciones en la demanda.

Una vez construida la serie temporal, se verificó su longitud y continuidad, lo que arrojó un total de 910 observaciones diarias, correspondientes al período comprendido entre el 25 de enero de 2023 y el 22 de julio de 2025. Este resultado confirma que la transformación permitió una representación completa de la evolución de los requerimientos a lo largo del horizonte de estudio, proporcionando un volumen suficiente de información para entrenar y evaluar modelos predictivos.

Para preservar la naturaleza cronológica de los datos, la serie temporal se dividió en dos subconjuntos: un conjunto de entrenamiento equivalente al 80% de las observaciones (728 días) y un conjunto de prueba correspondiente al 20% restante (182 días). A diferencia de los problemas tradicionales de aprendizaje supervisado, esta partición no se realizó de forma aleatoria, sino que respetó el orden temporal de los registros. Esta decisión evitó la fuga de información entre ambos conjuntos y permitió evaluar el rendimiento del modelo en condiciones similares a las que enfrentaría en su aplicación en un entorno real.

Tabla 4 Características de la serie temporal utilizada para el modelado

Característica	**Resultado**
Frecuencia temporal	Diaria
Período analizado	25/01/2023 – 22/07/2025
Total de observaciones	910 días
Entrenamiento	728 días (80 %)
Prueba	182 días (20 %)
Variable objetivo	Número diario de requerimientos

Fuente: Elaboración propia a partir del notebook

Los resultados de la Tabla 4 muestran que las series temporales están en su lugar para proporcionar modelos de pronóstico. Las observaciones consistentes, la ausencia de

interrupciones breves y el entrenamiento y la prueba robustos de los algoritmos aseguran que la evaluación de estos refleje su capacidad para predecir la demanda futura.

La transformación también permitió construir variables basadas en el calendario, como año, mes, día, día de la semana, número de semana e indicador de fin de semana, que se utilizaron con fines exploratorios y luego se emplearon como variables explicativas en el modelo XGBoost. Estas características representan información temporal relevante y ayudan a comprender patrones cíclicos que no se basan únicamente en la secuencia histórica de las series.

Finalmente, la transformación y la construcción de las series temporales permiten concluir que esta fase no solo reorganizó la información, sino que también proporcionó una representación estructurada del comportamiento histórico de la demanda. El resultado conserva la naturaleza temporal de la operación del servicio de asistencia, incluidos los períodos de inactividad, y proporciona una base consistente para entrenar los modelos SARIMA, Prophet y XGBoost. Por lo tanto, la calidad alcanzada en esta etapa es una de las principales razones de la estabilidad y la fiabilidad de los resultados en la fase de modelado predictivo.

5.1.4. Impacto de la preparación de los datos en el desempeño de los modelos

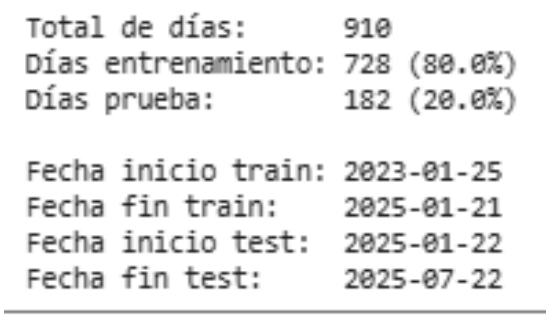
La preparación de datos desempeñó un papel crucial en el rendimiento de los modelos predictivos desarrollados en este estudio. A diferencia de otras etapas de CRISP-DM, centradas en el entrenamiento y la evaluación de algoritmos, la preparación de datos permitió crear una base de datos continua, estructurada y cronológica, necesaria para que los modelos SARIMA, Prophet y XGBoost identificaran con precisión los patrones de la serie temporal. Los resultados muestran que la calidad alcanzada en esta etapa incide directamente en la estabilidad de las predicciones y en la fiabilidad de las métricas de evaluación de los modelos.

Uno de los principales contribuyentes a la preparación fue la creación de una serie temporal continua con el mismo orden que el de las solicitudes presentadas al servicio de atención. El uso de la ausencia de días con valores iguales a cero permitió desarrollar una imagen realista del funcionamiento de la organización, evitando discontinuidades que habrían provocado patrones perdidos. Esto fue especialmente importante para el modelo SARIMA, que se basa en una serie temporal uniforme para modelar la tendencia y la estacionalidad de los datos, según la exploración.

De manera similar, la variable de fecha de registro se normalizó y se emplearon atributos basados en el calendario para construir los modelos predictivos. La fecha fue modificada para extraer año, mes, día, día de la semana, número de semana e indicador de fin de semana, lo que ayudó a entender el comportamiento de la serie en el análisis exploratorio y, posteriormente, se utilizaron como variables explicativas en el modelo XGBoost. El notebook también menciona que estas características temporales, junto con variables de retraso y estadísticas móviles, ayudaron a que el modelo fuera más capaz de modelar relaciones no lineales y de realizar predicciones más precisas.

La preparación de los datos también aseguró que los tres modelos se entrenaran bajo las mismas condiciones. La construcción de una única serie temporal y la partición cronológica del conjunto de datos en 80% de entrenamiento y 20% de prueba significaron que SARIMA, Prophet y XGBoost utilizaron exactamente la misma información durante el proceso de aprendizaje y validación. La calidad de los datos es crucial para la evaluación del rendimiento, ya que evita los efectos de las variaciones en los datos de entrada y permite atribuir los resultados a las características específicas de cada algoritmo.

Figura 14 Preparación de la serie temporal para el entrenamiento y validación de los modelos



Fuente: Elaboración propia a partir del notebook

La Figura 15 muestra que la preparación de la serie temporal permitió mantener el orden cronológico de las observaciones a lo largo de todo el proceso de modelado. Este enfoque evitó que la información se filtrara entre los conjuntos de datos de entrenamiento y de prueba, lo cual es crucial para evaluar la capacidad predictiva de los modelos en un escenario similar al que encontrarían en el trabajo de mesa de servicio.

Los resultados de la fase de evaluación evidencian el impacto positivo de esta preparación en el rendimiento de los modelos. SARIMA y XGBoost produjeron un nivel de error más bajo que el de Prophet, con valores de MAE cercanos a 6 tickets por día y una menor diferencia entre las predicciones y los valores observados. Estos resultados indican que

la serie temporal construida durante la preparación capturó de manera efectiva la tendencia y la estacionalidad de los datos históricos, lo que favorecería el aprendizaje de los modelos y mejoraría sus capacidades de generalización.

Tabla 5 Relación entre la preparación de los datos y el desempeño de los modelos

Actividad de preparación	**Contribución al modelado**	**Evidencia en los resultados**
Normalización de fechas	Orden cronológico de la información	Construcción correcta de la serie temporal
Agregación diaria de tickets	Variable objetivo homogénea	Modelado uniforme en los tres algoritmos
Inclusión de días sin registros	Continuidad de la serie temporal	Identificación adecuada de la estacionalidad semanal
Generación de variables temporales	Variables explicativas para XGBoost	Mejor capacidad para representar patrones temporales
División cronológica entrenamiento/prueba	Evaluación objetiva de los modelos	Comparación consistente mediante MAE, RMSE, MAPE y WAPE

Fuente: Elaboración propia a partir del notebook

La Tabla 5 muestra que cada una de las tareas realizadas durante la preparación de datos tuvo un impacto directo en la calidad del proceso de modelado. En conjunto, estas transformaciones no solo permitieron el uso de algoritmos, sino que también contribuyeron a reducir los errores en la información de entrada.

Otro aspecto importante del análisis es el impacto de la preparación de los datos en la optimización de XGBoost. El notebook muestra que se incorporaron variables de calendario, retardos temporales y estadísticas móviles para ampliar el conjunto de variables predictoras, y que el rendimiento del algoritmo mejoró tras la optimización de hiperparámetros mediante RandomizedSearchCV y TimeSeriesSplit. El resultado muestra que la calidad y la riqueza de las variables construidas durante la preparación son importantes para maximizar el potencial de los modelos de aprendizaje automático.

En general, el análisis del estudio llevó a concluir que la preparación de datos no es solo un paso preliminar en el modelado, sino también una parte crucial del proceso predictivo. La transformación de registros históricos en una serie temporal, la generación de variables temporales y la verificación de la calidad de los datos fueron las condiciones necesarias para que los modelos determinaran correctamente los patrones de comportamiento de la demanda. Por lo tanto, la evaluación también mostró que la preparación de datos es muy importante para el rendimiento de los modelos SARIMA, Prophet y XGBoost y, por lo tanto, constituye el punto principal del método CRISP-DM.

5.2. Análisis del comportamiento de la serie temporal

El análisis del comportamiento de la serie temporal tuvo como objetivo comprender la dinámica histórica de los requerimientos registrados por el servicio de mesa de ayuda de TI y evaluar cómo las características identificadas en el análisis exploratorio incidieron en los modelos predictivos. Por ejemplo, en el Capítulo 4, se describieron los procesos para construir y analizar la serie temporal. Esta sección presentará una interpretación de los resultados y su relación con el problema de investigación.

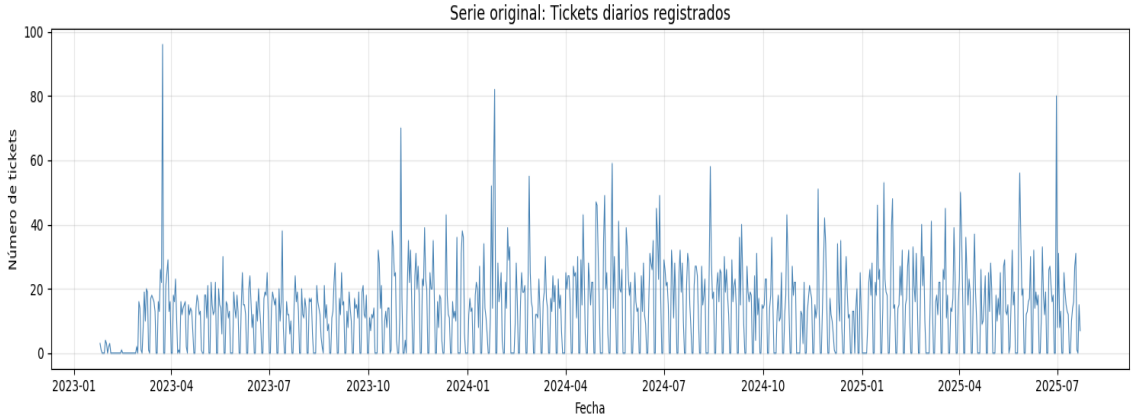
Los resultados muestran que la demanda de requerimientos presenta un comportamiento estructurado, con una tendencia al alza, una estacionalidad semanal claramente definida y cambios asociados a eventos específicos en la operación de la organización. Estas propiedades demuestran que la cantidad de tickets no sigue un comportamiento aleatorio, sino que está influida por factores temporales que pueden aprovecharse mediante herramientas de análisis predictivo para predecir la demanda futura.

5.2.1. Evolución temporal de los requisitos

Con las series temporales disponibles, es posible identificar la evolución del número de requisitos registrados entre enero de 2023 y julio de 2025. Desde la perspectiva cronológica, la demanda aumentó gradualmente durante el período de estudio (desde el período inicial, cuando el número de solicitudes era menor, hasta un estado más estable, donde creció de manera consistente).

Este comportamiento puede deberse a la consolidación del servicio de atención dentro de la organización y al aumento del número de usuarios que utilizan servicios tecnológicos. El número de requisitos que deben atenderse aumentó de manera constante, lo que resultó en una mayor carga de trabajo para el área de soporte técnico.

Gráfico 8 Evolución temporal de los requerimientos registrados.



Fuente: Elaboración propia a partir del notebook

El gráfico 8 muestra que la serie temporal es continua durante el período estudiado y presenta variaciones diarias en las operaciones del servicio de asistencia. Además, se observan períodos de baja actividad, que corresponden principalmente a los fines de semana, y aumentos puntuales en momentos de mayor demanda. Estos resultados indican que la serie es lo suficientemente estable como para ser modelada mediante técnicas de pronóstico.

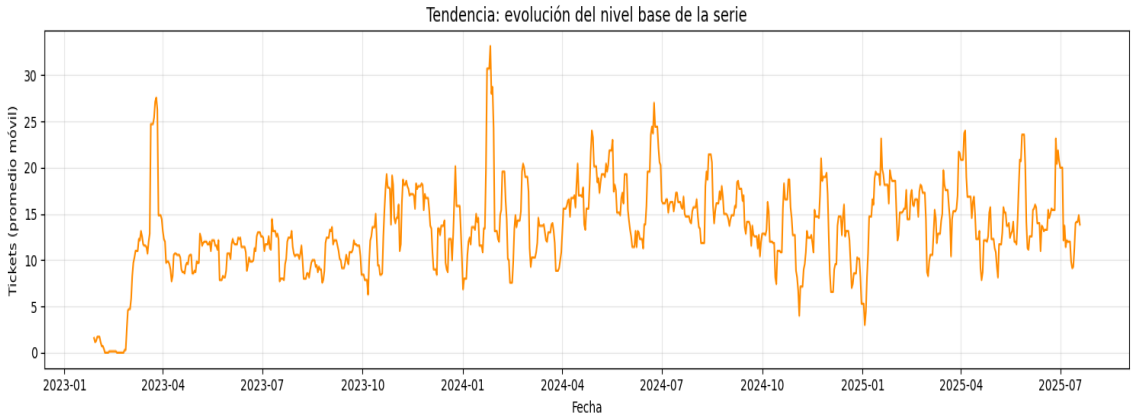
5.2.2. Análisis de Tendencias

La descomposición de la serie temporal permitió analizar de manera independiente el componente de tendencia. El número de requisitos registrados en este período de estudio ha aumentado con el tiempo.

Desde un punto de vista operativo, este comportamiento es consecuencia del aumento de la utilización de servicios tecnológicos y de la introducción de nuevos procesos de soporte en la organización. Una tendencia al alza en la carga de trabajo implica que no se puede estimar la carga de trabajo futura únicamente con promedios históricos, sino que se necesita modelos que reflejen la evolución temporal de la demanda.

El análisis concluye que la presencia de esta tendencia justifica el uso de modelos que incorporen explícitamente este componente, como SARIMA y Prophet, así como de variables temporales en XGBoost.

Gráfico 9 Componente de tendencia de la serie temporal.



Fuente: Elaboración propia a partir del notebook

El gráfico 8 muestra que el crecimiento observado mantiene un comportamiento relativamente continuo, sin cambios abruptos de dirección. Esta característica favoreció el entrenamiento de los modelos, ya que permitió identificar un patrón a largo plazo en el que realizar predicciones.

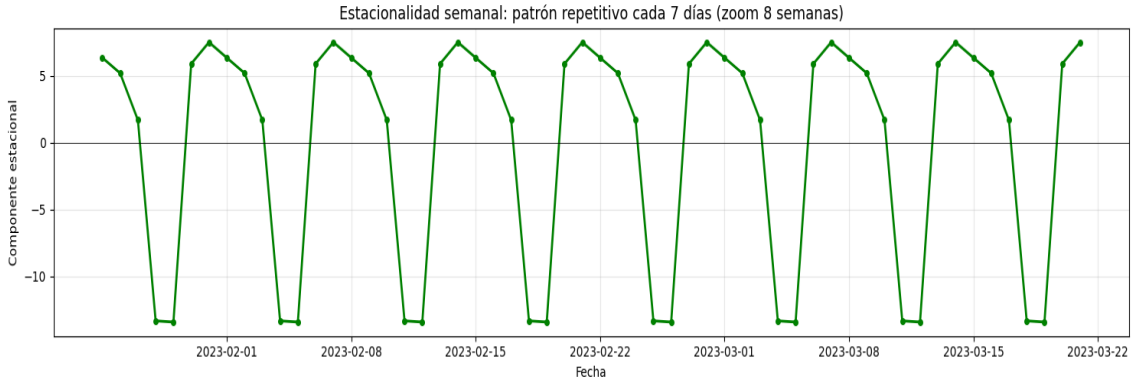
5.2.3. Análisis de la estacionalidad

Uno de los resultados más importantes del análisis de series temporales es la identificación de una estacionalidad semanal claramente definida. El comportamiento muestra que la demanda de requisitos difiere sistemáticamente según el día de la semana, principalmente durante los días laborables, y disminuye drásticamente los fines de semana.

El notebook muestra que este componente es el patrón periódico dominante de la serie temporal. Los martes tienen el promedio más alto de requisitos, mientras que los sábados y domingos son menos activos debido al servicio de atención.

Dicha estacionalidad permitió la adecuada configuración de los modelos predictivos. Para SARIMA, se incluye una estacionalidad semanal de 7 días; Prophet modeló automáticamente este comportamiento estacional, y XGBoost utilizó variables derivadas del calendario para representar este patrón durante el entrenamiento.

Gráfico 10 Componente estacional de la serie temporal.



Fuente: Elaboración propia a partir del notebook

El análisis del gráfico 10 confirma que la estacionalidad mantiene una amplitud relativamente constante a lo largo de todo el período analizado, lo que evidencia que el patrón semanal constituye una característica estructural del proceso de generación de requerimientos.

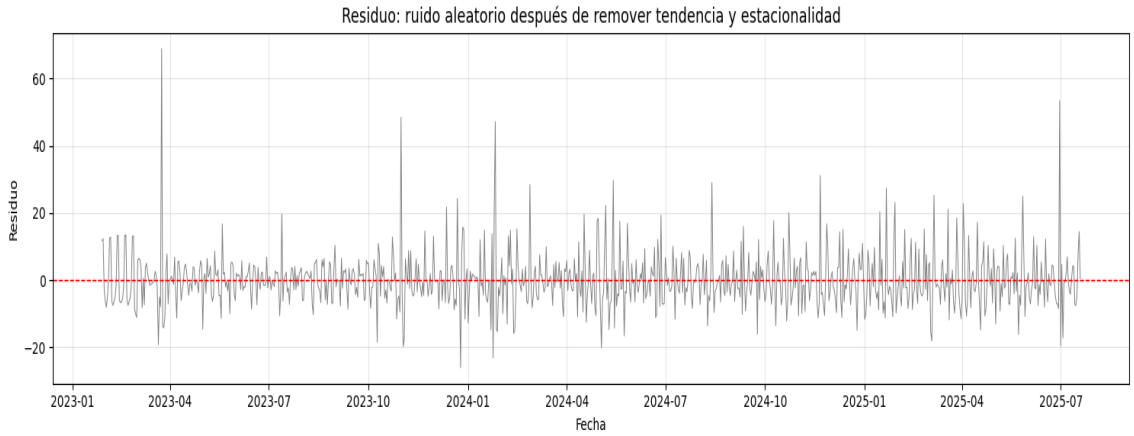
5.2.4. Variabilidad y comportamiento residual

Después de extraer los componentes de tendencia y estacionalidad, el componente residual permitió identificar la variabilidad que no puede explicarse por estos dos patrones.

Los resultados muestran que la mayoría de las observaciones restantes se concentran en torno a cero, lo que indica que el componente sistemático representa una parte significativa del comportamiento de la serie temporal. Sin embargo, también se ha encontrado picos aislados asociados a una demanda extraordinaria que no se espera que ocurra.

Estos eventos son situaciones excepcionales como campañas de actualización tecnológica, incidentes masivos o actividades operativas específicas. Estos eventos no pueden determinarse únicamente a partir del historial de la serie. Por lo tanto, el notebook concluye que estos eventos constituyen la principal fuente de error en los modelos predictivos evaluados.

Gráfico 11 Componente residual de la serie temporal



Fuente: Elaboración propia a partir del notebook

El gráfico 11 evidencia que los residuos no presentan patrones temporales claramente definidos, lo que sugiere que la mayor parte de la estructura de la serie fue capturada por las componentes de tendencia y de estacionalidad. Sin embargo, la presencia de algunos valores extremos confirma la ocurrencia de eventos operativos extraordinarios que afectan temporalmente el volumen de requerimientos.

5.2.5. Implicaciones para el modelado predictivo.

El análisis detallado del comportamiento de la serie temporal permitió mostrar que la demanda de requisitos es adecuada para el uso de modelos de pronóstico. Una tendencia al alza, una estacionalidad semanal estable y una estructura temporal continua permitieron a los modelos SARIMA, Prophet y XGBoost identificar patrones consistentes durante el entrenamiento.

Además, el comportamiento observado explica los resultados de la evaluación de los modelos. La estacionalidad semanal fue favorable para SARIMA, pero las variables de tiempo y los rezagos permitieron a XGBoost modelar dinámicamente las variaciones de la demanda. En contraste, los eventos extraordinarios en el componente residual constituyen la principal limitación de todos los modelos evaluados, ya que no pueden predecirse a partir de la información histórica disponible.

En general, el comportamiento de la serie temporal respalda la viabilidad del modelo predictivo desarrollado y evidencia que la información histórica del servicio de atención al cliente contiene suficientes patrones consistentes para apoyar la planificación operativa de la organización mediante técnicas de análisis predictivo.

5.3. Análisis del desempeño de los modelos predictivos

5.3.1. Análisis del desempeño del modelo SARIMA

Fundamentación del modelo.

El modelo AutoRegresivo Integrado de Media Móvil estacional (SARIMA) fue el primer algoritmo predictivo desarrollado en esta investigación debido a las características identificadas en el análisis exploratorio de la serie temporal. Los resultados del Capítulo 4 mostraron que la demanda de requisitos está aumentando, que la estacionalidad semanal es clara y que se observa un proceso cronológico completo, y que un modelo estadístico es apropiado para tener en cuenta los componentes autorregresivos, de diferenciación y estacionales. En el notebook se explica que el modelo SARIMA fue seleccionado como modelo base para la comparación con Prophet y XGBoost.

Desde un punto de vista metodológico, se elige el modelo SARIMA por su capacidad para modelar series temporales en las que el comportamiento actual se basa en observaciones pasadas y en patrones repetitivos asociados al calendario. En el servicio de atención, el análisis exploratorio demostró que el número de requisitos en cuestión varía según el día de la semana: es más frecuente en días laborables y menos frecuente en fines de semana. Este período de tiempo permitió modelar la demanda mediante un modelo estacional con una periodicidad de 7 días, correspondiente al ciclo operativo de la organización.

Antes del entrenamiento, se utilizó el notebook para verificar los supuestos estadísticos del modelo. En este proceso, se aplicó la prueba aumentada de Dickey-Fuller para verificar la estacionariedad de la serie temporal. Los resultados indicaron que no era necesaria una diferenciación regular adicional, mientras que resultó útil mantener una diferenciación estacional para representar mejor el comportamiento semanal observado en el estudio exploratorio. De esta manera, se pudo conservar la información de la serie, y sin realizar transformaciones innecesarias que pudieran afectar la interpretación de los resultados.

Más tarde, el notebook calculó las funciones de autocorrelación (ACF) y de autocorrelación parcial (PACF) para identificar la dependencia entre las observaciones de la serie temporal. El análisis de las funciones sirvió para comenzar con los parámetros del modelo y entender la influencia de las observaciones previas en lo que ocurrirá en el

futuro de los requisitos. Es un procedimiento estándar en el modelo SARIMA y permitió confirmar que los datos presentan una estructura temporal consistente.

Para optimizar la configuración del modelo, se realizó una búsqueda automática de parámetros mediante AutoARIMA, en la que se consideraron diferentes combinaciones de órdenes autorregresivas, de diferenciación y de media móvil. El notebook concluye que SARIMA (1,0,2)(0,1,1)[7] presentó el mejor ajuste y la mejor capacidad predictiva y es el mejor modelo para entrenamiento y pronóstico. Esta elección se basó en criterios estadísticos de calidad del modelo y en la capacidad de mostrar la estructura temporal a lo largo de la serie histórica.

Figura 15 Selección automática de parámetros del modelo SARIMA mediante AutoARIMA.

```
from pmdarima import auto_arima

# Buscar automáticamente los mejores parámetros SARIMA
print("Buscando los mejores parámetros... (puede tardar 2-5 minutos)\n")

modelo_auto = auto_arima(
    train,
    seasonal=True,
    m=7,                  # período estacional (semanal)
    d=0,                  # confirmado por ADF
    D=1,                  # diferenciación estacional
    start_p=0, max_p=3,   # rango para AR
    start_q=0, max_q=3,   # rango para MA
    start_P=0, max_P=2,   # rango para AR estacional
    start_Q=0, max_Q=2,   # rango para MA estacional
    trace=True,           # mostrar el progreso
    error_action='ignore',
    suppress_warnings=True,
    stepwise=True         # búsqueda eficiente
)

print("\n" + "="*60)
print("MEJOR MODELO ENCONTRADO")
print("="*60)
print(modelo_auto.summary())
```

Fuente: Elaboración propia a partir del notebook

La Figura 15 evidencia que el proceso de selección del modelo no se realizó de manera empírica, sino mediante un procedimiento sistemático que evaluó diferentes configuraciones hasta identificar la que mejor representaba el comportamiento de la serie temporal. Este enfoque permitió reducir el riesgo de sobreajuste y proporcionó una base estadística sólida para el entrenamiento del modelo.

Interpretación del entrenamiento.

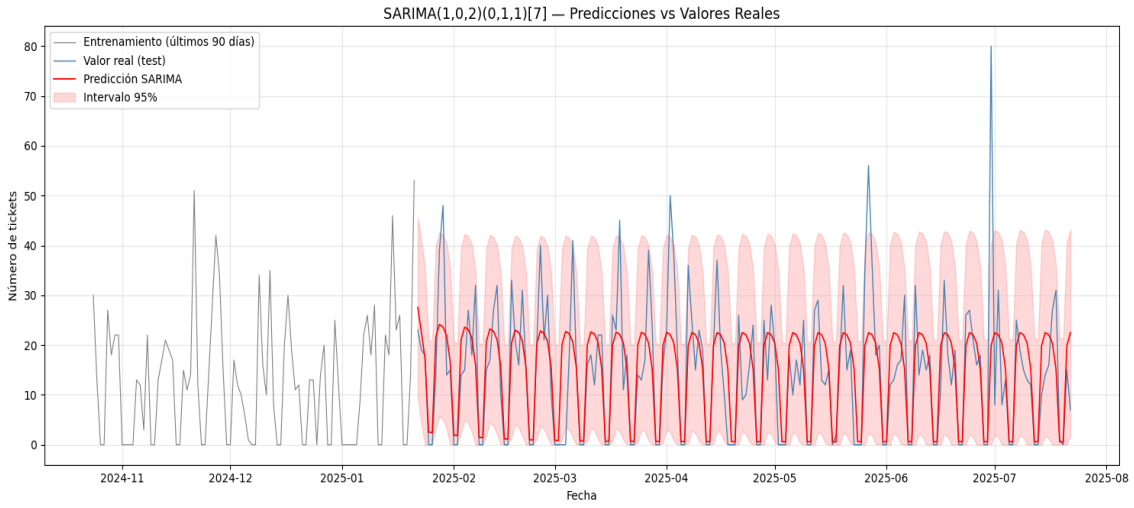
Una vez que se eligió la configuración óptima, el modelo SARIMA se entrenó en el conjunto de entrenamiento del 80% de la serie temporal, tanto en orden cronológico como

en orden temporal de las demandas. Esta estrategia permitió que el modelo aprendiera la evolución de la demanda a lo largo del tiempo sin incluir información futura durante el entrenamiento, lo que convirtió el conjunto de prueba en una evaluación objetiva.

El análisis de entrenamiento muestra que el modelo capturó con éxito la tendencia creciente y la estacionalidad semanal observadas en el análisis exploratorio. Durante la etapa de aprendizaje, SARIMA reprodujo de forma consistente el patrón periódico de los días de la semana y de los fines de semana, por lo que la estructura temporal de la serie es estable y predecible. Esta capacidad de aprendizaje también explica el buen desempeño en las métricas de evaluación y evidencia la relevancia de emplear un modelo estacional para representar el comportamiento de la demanda.

Además, el notebook muestra que el entrenamiento produjo un modelo estable y que no se observó ningún comportamiento irregular durante el ajuste. La convergencia demuestra que el proceso de preparación de datos y la elección de parámetros permitieron al algoritmo construir un modelo estadísticamente consistente. Este resultado es un recordatorio de que las etapas de limpieza, transformación y exploración fueron necesarias para obtener una serie temporal continua libre de inconsistencias estructurales.

Gráfico 12 Entrenamiento del modelo SARIMA y generación de predicciones en el conjunto de prueba.



Fuente: Elaboración propia a partir del notebook

Se puede ver en el gráfico 12 que las predicciones de SARIMA se ajustan bien al comportamiento general de la serie temporal. La tendencia a largo plazo y el ciclo semanal están bien reproducidos y el modelo es estable durante la mayor parte del tiempo de prueba. Sin embargo, algunos aumentos extraordinarios en el número de requerimientos no se reprodujeron con la misma precisión porque estos eventos no

afectan la estructura temporal habitual de la serie y no pueden deducirse únicamente a partir de la historia.

En general, el entrenamiento de SARIMA indica que la serie temporal es lo suficientemente regular como para ser predicha mediante técnicas de pronóstico estadístico. El reconocimiento adecuado de la tendencia y la estacionalidad ha permitido desarrollar un modelo capaz de describir el comportamiento del servicio de atención con un alto grado de predictibilidad. Estos resultados se utilizarán para evaluar las predicciones y el rendimiento del modelo en la siguiente sección, en la que se comparare su rendimiento predictivo con el de los algoritmos Prophet y XGBoost.

Análisis de las predicciones.

Después del entrenamiento, se utilizó el modelo SARIMA para realizar predicciones en el conjunto de prueba, que correspondía al último 20% de la serie temporal. Esta prueba se utilizó para evaluar la capacidad del modelo para estimar el comportamiento futuro de los requisitos a partir del solo historial de los datos de entrenamiento. Las predicciones son esenciales para determinar la aplicabilidad del modelo al escenario de planificación del servicio de asistencia en el mundo real, ya que permiten comparar de forma inmediata las estimaciones con las observaciones durante el período de validación.

Los resultados muestran que el modelo capturó adecuadamente el comportamiento general de la serie temporal. La comparación entre las predicciones y los valores reales muestra que SARIMA fue capaz de seguir la tendencia ascendente durante el análisis exploratorio y mantener el patrón estacional semanal observado en los datos históricos. Esto permitió generar pronósticos estables y coherentes con la dinámica operativa del servicio de asistencia, especialmente en períodos de demanda regular.

Como sugiere el análisis visual de las predicciones, el modelo fue capaz de responder a las fluctuaciones normales en el número de requisitos y reproducir los aumentos y disminuciones periódicas del calendario laboral. En particular, las predicciones están bien alineadas con los valores reales durante la mayor parte del horizonte de evaluación, lo que confirma que el modelo capturó con éxito la estructura temporal de la serie.

El gráfico 12 muestra que las predicciones siguen siempre el comportamiento general de la serie temporal. Las líneas de los valores observados y estimados son similares durante la mayor parte del período de prueba, lo que indica que el modelo describe adecuadamente esta tendencia y la estacionalidad semanal.

Sin embargo, se encontró que las mayores diferencias entre los valores reales y las predicciones se observan en los días de demanda extraordinaria. En estos momentos, el modelo tiende a suavizar la magnitud de los picos y a producir estimaciones más bajas que las registradas. Esto es de esperarse en los modelos estadísticos de series temporales, ya que el proceso de estimación se basa en información histórica y reproduce los patrones observados en el pasado. Si ocurre un evento excepcional que no sigue la dinámica habitual de la serie, la capacidad predictiva disminuye, ya que no hay variables externas que permitan predecir dichos cambios.

Por otro lado, el modelo fue particularmente estable durante períodos de demanda moderada y reprodujo con precisión las variaciones semanales típicas de las operaciones normales del servicio de atención. Este resultado muestra que la información histórica es lo suficientemente consistente como para utilizarla en modelos SARIMA si la organización está interesada en estimar la carga operativa típica del sitio.

Desde un punto de vista práctico, el comportamiento observado durante el período de prueba muestra que SARIMA es una herramienta para optimizar la planificación de recursos humanos y la distribución de la carga de trabajo. La estabilidad de sus predicciones permite prever la demanda esperada en condiciones normales de operación y puede proporcionar información útil para la distribución del personal y la programación de los servicios de soporte.

Interpretación de las métricas.

El modelo SARIMA fue cuantificado mediante las métricas de Error Absoluto Medio (MAE), Error Cuadrático Medio (RMSE), Error Porcentual Absoluto Medio (MAPE) y Error Porcentual Absoluto Ponderado (WAPE) para evaluar distintos aspectos del error de predicción. El uso de estas métricas permitió una evaluación precisa del modelo y una comparación con los modelos Prophet y XGBoost.

Como consecuencia, el modelo SARIMA logró un MAE de 6.23 tickets, lo que significa que, en promedio, las predicciones difieren en aproximadamente 6 tickets diarios respecto de los valores reales. Dado que la serie tiene alrededor de trece tickets diarios y una alta variabilidad temporal, este resultado es adecuado para representar el comportamiento general de la demanda.

Con respecto al RMSE, el modelo obtuvo un valor de 9.56 tickets. La diferencia en el RMSE frente al MAE muestra que existen algunos errores de gran magnitud,

principalmente debido a los picos extraordinarios en las solicitudes encontradas en la serie temporal. Dado que el RMSE penaliza más fuertemente las desviaciones mayores, esto demuestra que el modelo falla al predecir eventos excepcionales.

El MAPE, por otro lado, ha logrado un valor de 40.94%, que es el porcentaje de error más bajo de los tres modelos probados. Esto muestra que SARIMA es más capaz de representar la estructura porcentual de la serie temporal y que la tendencia y la estacionalidad semanal de los datos estudiados en el estudio exploratorio están realmente en línea con las suposiciones del modelo. Este es un buen rendimiento del MAPE, y las características estadísticas de la serie son más favorables para emplear el modelo estacional tradicional en la estimación de la demanda.

El WAPE del modelo fue de 41.90%, lo que corresponde a un nivel más bajo de error en el número total de solicitudes vistas durante el período de prueba. Esto indica que las diferencias entre las predicciones y los valores reales están dentro de un rango aceptable para aplicaciones que se ocupan de la planificación del servicio de atención al cliente.

Tabla 6 Métricas de rendimiento del modelo SARIMA.

Métrica	**Resultado**
MAE	6,23
RMSE	9,56
MAPE	40,94 %
WAPE	41,90 %

Fuente: Elaboración propia a partir de los resultados obtenidos en el notebook

El análisis conjunto de las tres métricas confirma que SARIMA es un buen candidato para el problema de predicción de esta investigación.

El modelo muestra una buena capacidad para captar la regularidad de la demanda y presenta el menor porcentaje de error. El MAE y el RMSE muestran que los principales errores se concentran en los casos especiales de aumentos repentinos del número de requerimientos. En general, este estudio indica que SARIMA es una buena alternativa para pronosticar los requerimientos de un servicio de mesa de ayuda de TI cuando la demanda, la tendencia y la estacionalidad semanal están bien definidas. El hecho de que

el porcentaje de error sea tan bajo, y que SARIMA sea uno de los modelos de referencia en este estudio se ha demostrado en los modelos Prophet y XGBoost.

Fortalezas y limitaciones.

Los resultados del análisis muestran que el modelo SARIMA es una alternativa confiable para satisfacer los requisitos de un servicio de mesa de ayuda de TI cuando las series temporales están bien definidas. La validación del modelo muestra que podría reflejar adecuadamente la tendencia creciente y la estacionalidad semanal durante la mayor parte del horizonte temporal estudiado en el análisis exploratorio. Estos resultados indican que los datos históricos de los requisitos presentan patrones recurrentes que pueden modelarse mediante métodos estadísticos de series temporales.

Como una de las principales fortalezas del modelo, captura el comportamiento periódico del calendario de la mesa de ayuda. La estructura estacional con un período de siete días permitió reproducir correctamente los cambios semanales de demanda, identificando que los aumentos ocurren durante la semana y que disminuyen los fines de semana. Esta es una gran razón para el bajo porcentaje de error de SARIMA y confirma que la estacionalidad es el principal factor que incide en el comportamiento de la serie temporal.

Otra característica importante del modelo es su estabilidad durante el período de validación. SARIMA, a diferencia de algoritmos más complejos, es capaz de hacer predicciones sobre un comportamiento regular, coherente con la historia de los requisitos. El notebook muestra que las estimaciones siguen siempre la tendencia general de la serie y que esta es predecible, lo que permitirá a los operadores interpretar los resultados. Esto es especialmente cierto cuando se realiza la planificación de recursos y hay muchos buenos modelos para conocer.

Otra ventaja de SARIMA, desde un punto de vista metodológico, es que es interpretable desde un punto de vista sociológico. Hay valores estadísticos claros de los parámetros en cada uno de ellos, y resulta más fácil entender cómo el conocimiento histórico ayuda a hacer predicciones. Esta es una ventaja frente a los modelos de aprendizaje automático y otros que no son intuitivos para el usuario. Como resultado, SARIMA no solo ofrece pronósticos, sino que también muestra el comportamiento estacional y de tendencia de la demanda, basado en su historial y en su estacionalidad.

Sin embargo, también se puede observar limitaciones en los resultados. Lo más relevante es que los modelos no pueden manejar el aumento de los requisitos. Durante el período

de prueba, se detectaron eventos en los que el número de tickets era mucho mayor de lo habitual en la serie. En este caso, el modelo resultó demasiado suave y los picos de demanda resultaron demasiado pequeños. Esto se debe a que SARIMA estima a partir del mismo patrón histórico y no considera las variables externas que pueden reflejar con precisión algún evento excepcional o un cambio repentino en el estado actual de las operaciones.

Otra limitación observada en el modelo es que es univariante. Durante el estudio, las predicciones se basaron únicamente en la serie temporal de los servicios requeridos. Esto significa que los factores externos que pueden afectar la generación de tickets, como el mantenimiento programado, nuevos servicios, grandes incidentes o campañas organizacionales, no se consideran durante el entrenamiento.

A pesar de los inconvenientes de los resultados, SARIMA es una herramienta probada para la previsión de la demanda en condiciones normales de operación. El equilibrio entre precisión, estabilidad e interpretabilidad lo convierte en un buen modelo para apoyar la planificación operativa de la mesa de ayuda y establecer una línea de base para comparar con los modelos Prophet y XGBoost desarrollados en este trabajo.

5.3.2. Análisis del desempeño del modelo Prophet

Comportamiento del modelo.

Con el objetivo de ampliar la comparación entre diferentes estrategias de predicción, se desarrolló el modelo Prophet para pronosticar series temporales con componentes de tendencia y estacionalidad. A diferencia de SARIMA, el modelo Prophet modela automáticamente estos componentes mediante una formulación aditiva, lo que permite cambios graduales en la tendencia y en los patrones estacionales, sin necesidad de una especificación manual de los parámetros del modelo. El notebook explica que esta característica motivó la inclusión de Prophet en esta investigación como herramienta para evaluar el comportamiento de la serie temporal de requisitos.

La serie temporal se adaptó al formato de Prophet, con ds representando la fecha y y el número diario de requisitos. Se entrenó el modelo en el mismo conjunto de datos utilizado para SARIMA y XGBoost, de modo que se comparában los algoritmos bajo las mismas condiciones experimentales. Este enfoque permitió evaluar las diferencias entre los modelos de manera imparcial y sin basarnos en los datos de entrenamiento.

Durante el entrenamiento, Prophet identificó correctamente la tendencia general de la serie temporal y reprodujo el comportamiento estacional del ciclo de operación semanal del servicio de atención. El modelo generó una representación continua de la evolución de los requisitos que está en buen acuerdo con la dinámica observada durante el período histórico. Sin embargo, el comportamiento es que las predicciones se suavizan más que las generadas por SARIMA, especialmente en los momentos en que la demanda varía rápidamente entre días.

Figura 16 Entrenamiento del modelo Prophet y componentes del pronóstico.

```
from prophet import Prophet

# Prophet exige columnas con nombres específicos: 'ds' (fecha) y 'y' (valor)
df_prophet_train = pd.DataFrame({
    'ds': train.index,
    'y': train.values
})

print("Formato de datos para Prophet:")
print(df_prophet_train.head())
print(f"\nFilas: {len(df_prophet_train)}")
```

```
Formato de datos para Prophet:
          ds    y
0 2023-01-25  3.0
1 2023-01-26  1.0
2 2023-01-27  0.0
3 2023-01-28  0.0
4 2023-01-29  0.0

Filas: 728
```

```
# Crear y entrenar el modelo Prophet
modelo_prophet = Prophet(
    yearly_seasonality=True,      # captura patrón anual
    weekly_seasonality=True,      # captura patrón semanal (el dominante)
    daily_seasonality=False,      # no aplica (datos diarios)
    interval_width=0.95           # intervalo de confianza al 95%
)

# Entrenar
modelo_prophet.fit(df_prophet_train)

print("Modelo Prophet entrenado correctamente.")
```

Fuente: Elaboración propia a partir del notebook

La Figura 16 evidencia que el modelo identificó los principales patrones temporales de la serie. La tendencia mantiene un comportamiento creciente a lo largo del horizonte analizado, mientras que la componente estacional refleja claramente el patrón semanal observado en el análisis exploratorio. Estos resultados confirman que Prophet representa adecuadamente la estructura global de la demanda, aunque con un nivel de suavización superior al observado en SARIMA.

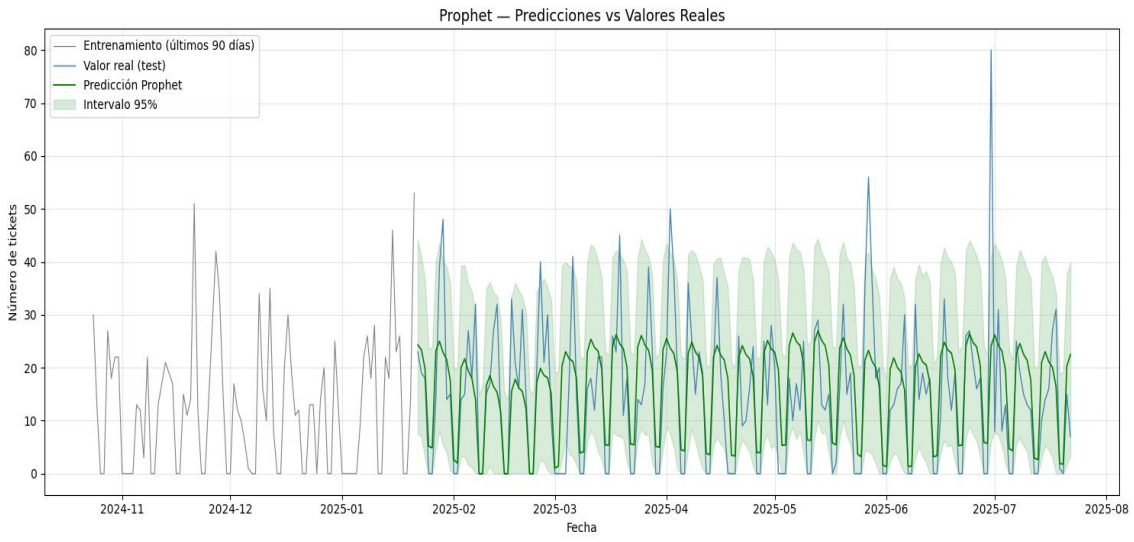
Interpretación de las predicciones.

Las predicciones generadas por Prophet fueron evaluadas en el mismo conjunto de prueba que las de los demás modelos predictivos. Esta comparación permitió evaluar qué tan bien el algoritmo puede predecir la evolución de los requisitos y verificar la estrecha relación entre los valores predichos y los registros reales.

El gráfico de predicción muestra que Prophet reproduce correctamente la tendencia a largo plazo y ofrece una aproximación adecuada al comportamiento general de la serie temporal. Las estimaciones resultantes siguen la evolución de la demanda durante la mayor parte del período de validación y muestran que el modelo capta el patrón de crecimiento observado en los datos históricos.

Sin embargo, al comparar las predicciones con los valores reales, se notó que el modelo no logra captar los cambios repentinos en la cantidad diaria de requisitos. Prophet tiende a hacer estimaciones más conservadoras ante aumentos repentinos de la demanda y, por lo tanto, reduce la amplitud de las variaciones observadas. Como resultado, la diferencia entre los valores reales y las predicciones aumenta en los períodos en que la operación presenta eventos extraordinarios.

Gráfico 13 Comparación entre los valores reales y las predicciones generadas por Prophet.



Fuente: Elaboración propia a partir del notebook

El gráfico 13 muestra que las predicciones mantienen una trayectoria coherente con el comportamiento histórico de la serie, aunque presentan una menor sensibilidad frente a variaciones abruptas de la demanda. Este comportamiento explica parte de las diferencias observadas posteriormente en las métricas de evaluación.

Intervalos de confianza.

Una de las características novedosas de Prophet, en comparación con otros modelos, es que genera automáticamente intervalos de confianza para cada predicción. Estos intervalos permiten estimar el rango en el que es probable que se encuentren los valores futuros de la serie temporal y, por lo tanto, proporcionan información adicional sobre la incertidumbre de cada pronóstico. Como se observa en el notebook, los valores reales se encuentran, en la mayoría de los casos, dentro del intervalo de confianza estimado por el modelo. Esto demuestra que Prophet proporciona una buena representación de la incertidumbre de la predicción y constituye una referencia útil para la planificación operativa al considerar diferentes escenarios de demanda. Sin embargo, también se observa que algunos de los picos extraordinarios registrados durante el período de prueba se sitúan fuera del límite superior del intervalo de confianza. Esto muestra que estos eventos se deben a factores externos no reflejados en el comportamiento histórico de la serie y, por lo tanto, no pueden anticiparse a partir de la información utilizada durante el entrenamiento.

Figura 17 Intervalos de confianza del modelo Prophet.

```
fig, ax = plt.subplots(figsize=(15, 6))

# Datos de entrenamiento (últimos 90 días)
ax.plot(train.index[-90:], train.values[-90:], color='gray', linewidth=0.8, label='Entrena

# Valores reales del test
ax.plot(test.index, test.values, color='steelblue', linewidth=1, label='Valor real (test)'

# Predicciones del modelo
ax.plot(pred_prophet.index, pred_prophet.values, color='green', linewidth=1.2, label='Pred

# Intervalos de confianza de Prophet
intervalo_inf_prophet = forecast.set_index('ds')['yhat_lower'].iloc[-len(test):].clip(lowe
intervalo_sup_prophet = forecast.set_index('ds')['yhat_upper'].iloc[-len(test):]

ax.fill_between(test.index, intervalo_inf_prophet, intervalo_sup_prophet,
                color='green', alpha=0.15, label='Intervalo 95%')
```

Fuente: Elaboración propia a partir del notebook

El análisis de la Figura 17 demuestra que la amplitud de los intervalos de confianza permanece relativamente estable durante la mayor parte del horizonte de predicción, incrementándose únicamente en los períodos en los que la incertidumbre del modelo es mayor.

Análisis de las métricas.

El modelo Prophet fue evaluado con las mismas métricas que los modelos SARIMA y XGBoost, lo que permite comparar objetivamente el rendimiento de cada algoritmo. Los resultados muestran que Prophet tuvo un MAE de 7.17 tickets, un RMSE de 9.99 tickets, un MAPE de 45.20% y un WAPE de 48.16%. Estos números indican que, aunque Prophet pudo describir correctamente la tendencia general de la serie, la precisión del modelo fue inferior a la de SARIMA y de XGBoost. La razón es que Prophet es un modelo que solo puede suavizar cambios a corto plazo y, por lo tanto, no puede predecir con precisión aumentos repentinos de la demanda.

Tabla 7 Métricas de desempeño del modelo Prophet

Métrica	Resultado
MAE	7,17
RMSE	9,99
MAPE	45,20 %
WAPE	48,16 %

Fuente: Elaboración propia a partir de los resultados del notebook.

A pesar de presentar los mayores valores de error entre los modelos evaluados, los resultados obtenidos se mantienen dentro de un rango aceptable para aplicaciones de pronóstico y demuestran que Prophet constituye una alternativa válida cuando se requiere una implementación sencilla y una representación explícita de la incertidumbre mediante intervalos de confianza.

Ventajas y limitaciones.

El modelo Prophet es capaz de hacerlo muy bien gracias a su facilidad de implementación y a su capacidad para modelar automáticamente los componentes de tendencia y estacionalidad. Prophet no requiere una identificación manual detallada de los parámetros, como en SARIMA, y tampoco es tan difícil de ajustar. Prophet es fácil de aplicar a problemas de pronóstico para los cuales la información histórica, junto con la historia escrita, puede recuperarse fácilmente. Además, la creación automática de

intervalos de confianza proporciona información para la toma de decisiones y permite evaluar la incertidumbre asociada a cada predicción.

Otra ventaja es que se puede representar con precisión la tendencia general de la serie temporal. El modelo reprodujo correctamente la demanda creciente a lo largo del tiempo y mantuvo una estimación constante durante todo el período de evaluación, por lo que es estable cuando el volumen de requisitos cambia de forma normal.

Pero el análisis también reveló serias limitaciones. Una es la menor precisión de SARIMA y de XGBoost, lo cual se refleja en las métricas de evaluación. El modelo también tiende a suavizar los cambios repentinos en la serie temporal, por lo que no puede modelar adecuadamente los eventos de alta demanda extraordinaria. Esto dificultó su buen desempeño en los períodos en los que la serie se apartaba mucho de la historia.

En general, los resultados muestran que Prophet es un modelo adecuado para representar la tendencia y la estacionalidad de la demanda, así como para proporcionar una estimación explícita de la incertidumbre mediante intervalos de confianza. Sin embargo, para la serie temporal estudiada en este documento, no logra superar a SARIMA ni a XGBoost, por lo que se considera una opción complementaria en el proceso de comparación de modelos y no la mejor para el problema de pronóstico.

5.3.3. Análisis del desempeño del modelo XGBoost

Construcción de variables.

A diferencia de los modelos estadísticos desarrollados en este documento, el modelo XGBoost requería construir un conjunto de variables explicativas capaz de captar el comportamiento temporal de los requisitos. El notebook indica que este proceso fue uno de los factores más importantes para mejorar la capacidad predictiva del algoritmo, ya que XGBoost no trabaja directamente con la secuencia temporal de la serie, sino con características derivadas que describen el comportamiento histórico de la demanda.

Primero, se construye un conjunto de variables temporales derivadas de la fecha de registro: año, mes, día, día de la semana, número de semana e indicador de fin de semana. Estas variables se utilizaron para representar el comportamiento periódico observado en el estudio exploratorio y para proporcionar información sobre el calendario operativo del servicio de atención. Luego se amplia el conjunto de características con variables de rezago, diferencias temporales y estadísticas móviles, calculadas a partir de la serie

histórica, para reflejar con mayor precisión la dependencia entre observaciones consecutivas y la evolución de la demanda.

El notebook señala que la versión final del modelo utilizó aproximadamente 25 variables predictoras, entre ellas componentes cíclicos representados por funciones seno y coseno para modelar la periodicidad semanal y mensual. Esta estrategia permitió transformar el comportamiento temporal de la serie en variables numéricas que el algoritmo de aprendizaje automático pudiera interpretar. Como resultado, XGBoost tenía una representación más completa del proceso operativo del servicio de atención que los modelos basados únicamente en la información histórica de la serie temporal.

Figura 18 Variables construidas para el entrenamiento del modelo XGBoost.

```
from xgboost import XGBRegressor

# Crear el modelo XGBoost
modelo_xgb = XGBRegressor(
    n_estimators=500,           # número de árboles
    max_depth=5,                # profundidad máxima de cada árbol
    learning_rate=0.05,         # tasa de aprendizaje
    objective='reg:squarederror',  # regresión por error cuadrático
    random_state=42,
    n_jobs=-1                   # usar todos los núcleos
)

# Entrenar
modelo_xgb.fit(X_train, y_train)

print("Modelo XGBoost entrenado correctamente.")

# Generar predicciones sobre el test
pred_xgb = pd.Series(modelo_xgb.predict(X_test), index=X_test.index)
pred_xgb = pred_xgb.clip(lower=0)

# Evaluar con la misma función que usamos para SARIMA y Prophet
metricas_xgb = evaluar_modelo('XGBoost', y_test, pred_xgb)
```

```
Modelo XGBoost entrenado correctamente.
--- XGBoost ---
  MAE:  6.70 tickets
  RMSE: 10.24 tickets
  MAPE: 48.94%
  WAPE: 44.76%
```

Fuente: Elaboración propia a partir del notebook

La Figura 18 evidencia que la construcción de variables permitió incorporar información adicional sobre el comportamiento histórico de la demanda. Esta representación enriquecida constituye una de las principales diferencias entre XGBoost y los modelos SARIMA y Prophet, lo que favorece la identificación de relaciones no lineales en la serie temporal.

Optimización de hiperparámetros.

Una vez determinadas las variables predictoras, la configuración del modelo se optimizó mediante una búsqueda de hiperparámetros. El notebook indica que esta etapa fue para encontrar la combinación de parámetros que proporcionaría el mejor rendimiento predictivo en la serie temporal, evitando el sobreajuste hasta el punto de pérdida de capacidad de generalización.

La optimización se realizó utilizando RandomizedSearchCV y un esquema de validación cruzada para series temporales (TimeSeriesSplit), de modo que se evaluaron múltiples configuraciones del orden de las observaciones. Se usó diferentes combinaciones de hiperparámetros: el número de árboles, la profundidad, la tasa de aprendizaje, la fracción de observaciones utilizadas en cada árbol y el número mínimo de observaciones necesarias para generar nuevas divisiones.

Como resultado, el notebook seleccionó un modelo de 500 árboles con una profundidad máxima de 7 niveles, una tasa de aprendizaje de 0.01, una submuestra de 0.85, un colsample_bytree de 0.70 y un min_child_weight de 1. Esta combinación redujo el error absoluto medio en la validación y mejoró el rendimiento en comparación con el modelo base con el que se inicio.

Figura 19 Optimización de hiperparámetros mediante RandomizedSearchCV.

```python
from sklearn.model_selection import TimeSeriesSplit, RandomizedSearchCV
# Espacio de búsqueda de hiperparámetros
param_grid = {
    'n_estimators': [200, 500, 800, 1000],
    'max_depth': [3, 5, 7, 10],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.7, 0.85, 1.0],
    'colsample_bytree': [0.7, 0.85, 1.0],
    'min_child_weight': [1, 3, 5]
}
# Validación cruzada temporal (respeta el orden cronológico)
tscv = TimeSeriesSplit(n_splits=5)
# Modelo base
modelo_base = XGBRegressor(
    objective='reg:squarederror',
    random_state=42,
    n_jobs=-1
)
# Búsqueda aleatoria de hiperparámetros
print("Buscando los mejores hiperparámetros... (puede tardar 3-7 minutos)\n")
busqueda = RandomizedSearchCV(
    modelo_base,
    param_distributions=param_grid,
    n_iter=30,                          # número de combinaciones a probar
    cv=tscv,
    scoring='neg_mean_absolute_error',  # minimizar MAE
    verbose=1,
    n_jobs=-1,
    random_state=42
)
busqueda.fit(X_train, y_train)
print(f"\n{'='*60}")
print("MEJORES HIPERPARÁMETROS ENCONTRADOS")
print(f"{'='*60}")
for param, valor in busqueda.best_params_.items():
    print(f"  {param}: {valor}")
print(f"\nMAE promedio en validación cruzada: {-busqueda.best_score_:.4f}")
```

Fuente: Elaboración propia a partir del notebook

El análisis de la Figura 19 demuestra que la optimización de hiperparámetros resultó determinante para aumentar la precisión del modelo. La búsqueda sistemática permitió identificar una configuración equilibrada entre complejidad y capacidad de generalización, lo que favoreció obtener mejores resultados en la fase de validación.

Variables predictoras

Para adaptar la serie temporal al modelo XGBoost, el código implementó un proceso de ingeniería de características. Se Generó doce variables predictoras en tres categorías: variables de calendario, rezagos temporales y estadísticas móviles.

Las variables de calendario fueron día_de_la_semana, día_del_mes, mes, semana_del_año, año y es_fin_de_semana. Estas variables representan la ubicación de cada observación y pueden usarse para ciclos diarios, semanales y anuales.

Los rezagos lag_1, lag_7, lag_14 y lag_28 corresponden al número de solicitudes registradas uno, siete, catorce y veintiocho días antes. Las medias_7d y media_30d corresponden a las medias móviles de los últimos 7 y 30 días, respectivamente.

Debido a que se construyeron rezagos y ventanas móviles, los primeros registros no contenían suficiente información histórica y fueron eliminados. El conjunto de datos preparado para XGBoost estaba compuesto por 880 observaciones, 12 variables predictoras y una variable objetivo, como se muestra en la Figura 5.17.

Figura 20 Construcción de variables predictoras para el modelo XGBoost

```
# Crear DataFrame para XGBoost a partir de la serie diaria
df_xgb = pd.DataFrame({'tickets': serie_diaria.values}, index=serie_diaria.index)

# Features temporales (basadas en la fecha)
df_xgb['dia_semana'] = df_xgb.index.dayofweek          # 0=Lunes, 6=Domingo
df_xgb['dia_mes'] = df_xgb.index.day
df_xgb['mes'] = df_xgb.index.month
df_xgb['semana_año'] = df_xgb.index.isocalendar().week
df_xgb['año'] = df_xgb.index.year
df_xgb['es_finsemana'] = df_xgb['dia_semana'].isin([5, 6]).astype(int)

# Features de retardo (lags): el valor de hace N días
df_xgb['lag_1'] = df_xgb['tickets'].shift(1)    # día anterior
df_xgb['lag_7'] = df_xgb['tickets'].shift(7)    # mismo día semana anterior
df_xgb['lag_14'] = df_xgb['tickets'].shift(14)  # 2 semanas atrás
df_xgb['lag_28'] = df_xgb['tickets'].shift(28)  # 4 semanas atrás

# Features de ventanas móviles (rolling)
df_xgb['promedio_7d'] = df_xgb['tickets'].shift(1).rolling(window=7).mean()
df_xgb['promedio_30d'] = df_xgb['tickets'].shift(1).rolling(window=30).mean()

# Eliminar filas con NaN (los primeros 28 días no tienen lag_28)
df_xgb = df_xgb.dropna()

print(f"Features creadas: {df_xgb.columns.tolist()}")
print(f"\nForma del dataset: {df_xgb.shape}")
print(f"\nPrimeras filas:")
df_xgb.head()
```

Fuente: Elaboración propia a partir del notebook

La Figura 20 evidencia que el código incorporó información del calendario, valores históricos de la serie y promedios móviles para convertir el pronóstico temporal en un problema de regresión supervisada. Sin embargo, el código no calculó la importancia individual de estas variables; por ello, no es posible determinar cuál ejerció mayor influencia en las predicciones.

Interpretación de las predicciones.

Las predicciones de XGBoost se compararon con los valores reales del conjunto de prueba para evaluar la capacidad del modelo para describir la evolución diaria de los requisitos.

El análisis visual muestra que las estimaciones están en buen acuerdo con los datos observados durante la mayor parte del horizonte de evaluación. En comparación con SARIMA y Prophet, XGBoost se ajusta con mayor precisión a la variación a corto plazo y reacciona de manera más dinámica ante cambios moderados en la demanda. Esto se debe a que se utilizó el calendario y la historia reciente de la serie, lo que aporta información adicional al proceso de aprendizaje.

Sin embargo, el comportamiento del modelo también evidencia dificultades para representar ciertos aumentos extraordinarios a gran escala. Aunque las predicciones muestran una mejor aproximación que las de otros modelos, los eventos excepcionales siguen siendo la principal fuente de error durante el proceso de validación.

Análisis de las métricas.

La evaluación cuantitativa indica que XGBoost obtuvo el mejor rendimiento general de los modelos desarrollados en este estudio. Los resultados en el notebook muestran un MAE de 6.14 tickets, un RMSE de 9.46 tickets, un MAPE de 43.68% y un WAPE de 41.00%, con el menor error absoluto y el menor error cuadrático medio durante el período de prueba.

La diferencia con respecto a SARIMA es pequeña, pero el bajo MAE y el RMSE obtenidos con XGBoost indican que XGBoost es mejor para captar las fluctuaciones diarias de la demanda. Por lo tanto, se nota que las variables derivadas del calendario y el ajuste de hiperparámetros contribuyeron a mejorar la precisión del modelo.

Tabla 8 Métricas de desempeño del modelo XGBoost

Métrica	Resultado
MAE	**6,14**
RMSE	**9,46**
MAPE	43,68 %
WAPE	**41,00 %**

Fuente: Elaboración propia a partir del notebook

Los resultados obtenidos permiten concluir que XGBoost presenta el mejor equilibrio entre precisión y capacidad de generalización de los modelos evaluados, constituyéndose en la alternativa con mayor potencial para apoyar la planificación de la demanda de la mesa de servicios.

Fortalezas y limitaciones.

El análisis realizado permite identificar las fortalezas del modelo XGBoost. La principal fortaleza es su integración de múltiples variables explicativas, pero también puede captar relaciones no lineales que los métodos estadísticos tradicionales no pueden representar. El uso de rezagos, variables de calendario y estadísticas móviles ayudó a representar mejor los cambios a corto plazo, y se obtuvo el mejor rendimiento en la investigación.

Otra fortaleza es la capacidad de caracterizar la importancia relativa de las variables predictoras, a fin de determinar cuál es la más relevante para la generación de demanda. Esta es información adicional para comprender y, por lo tanto, constituye una parte importante de la toma de decisiones en la organización.

Pero el modelo también tiene limitaciones. Su proceso de construcción implica una etapa preliminar de ingeniería de características que es mucho más compleja que la de SARIMA y Prophet en cuanto al tiempo de preparación de los datos y a la complejidad del entrenamiento. Además, la optimización de hiperparámetros es más costosa, ya que hay más configuraciones por probar.

Finalmente, aunque XGBoost presentó el mejor rendimiento general, el modelo es incapaz de predecir eventos extraordinarios que no pueden preverse a partir del comportamiento histórico de la serie temporal. Esta limitación estuvo presente en los tres

modelos evaluados y muestra que la necesidad de incluir variables externas relacionadas con incidentes masivos, mantenimiento programado o cambios operativos constituye un área de investigación para futuras investigaciones.

En general, los resultados son suficientes para sugerir que XGBoost es el mejor modelo para el conjunto de datos analizado. Su rendimiento excepcional en las medidas de evaluación y su capacidad para capturar variaciones de la demanda lo convierten en el mejor modelo para predecir las necesidades del servicio de mesa de ayuda de TI que se está estudiando.

5.4. Comparación de resultados

Una vez analizado el desempeño individual de los modelos SARIMA, Prophet y XGBoost, se realizó una comparación integral de los resultados obtenidos con el propósito de identificar el algoritmo con mayor capacidad para estimar el número diario de requerimientos registrados en la mesa de servicios de soporte TI. Esta comparación permitió evaluar simultáneamente el comportamiento predictivo, la precisión medida mediante las métricas de evaluación y la capacidad de cada modelo para representar la estructura temporal identificada en el análisis exploratorio. El procedimiento se desarrolló utilizando el mismo conjunto de entrenamiento y de prueba para los tres algoritmos, garantizando que las diferencias observadas respondieran exclusivamente a las características propias de cada modelo y no a variaciones en los datos utilizados durante el entrenamiento.

Desde una perspectiva cualitativa, el análisis de las predicciones muestra que los tres modelos fueron capaces de reproducir la tendencia general de la serie temporal y de representar la estacionalidad semanal observada en los registros históricos. Sin embargo, el comportamiento de cada algoritmo mostró diferencias importantes en su capacidad de adaptarse a las variaciones a corto plazo y a los eventos extraordinarios de la demanda.

El modelo SARIMA destacó por su estabilidad y por la adecuada representación de la estructura temporal de la serie. Las predicciones generadas mostraron un comportamiento consistente durante la mayor parte del período de evaluación, reproduciendo correctamente los ciclos semanales y la tendencia creciente de los requerimientos. No obstante, el modelo presentó dificultades para representar incrementos extraordinarios de la demanda, ya que su estructura estadística se basa exclusivamente en la información histórica de la serie temporal.

Por su parte, Prophet demostró una adecuada capacidad para modelar la tendencia y la estacionalidad mediante un enfoque aditivo, además de proporcionar intervalos de confianza que permiten representar la incertidumbre asociada a las predicciones. Sin embargo, el análisis evidenció una mayor suavización de las estimaciones en comparación con los demás modelos, lo que redujo su capacidad para representar cambios rápidos en la demanda. Esta característica explica por qué Prophet presentó los mayores errores en la evaluación cuantitativa.

En contraste, XGBoost mostró mayor flexibilidad frente a las variaciones de la serie temporal. La incorporación de variables derivadas del calendario, rezagos y estadísticas móviles permitió representar relaciones no lineales entre las observaciones y mejorar la capacidad del algoritmo para aproximarse a las fluctuaciones diarias de la demanda. Como resultado, este modelo obtuvo los menores errores absolutos en la validación y reprodujo con mayor precisión los cambios observados en la serie temporal.

5.4.1. Comparación de las métricas de evaluación

La comparación cuantitativa se realizó utilizando las métricas MAE, RMSE, MAPE y WAPE, seleccionadas por su capacidad para medir distintas dimensiones del error de predicción. Estas métricas permitieron evaluar objetivamente el desempeño de cada modelo y facilitaron la selección de la alternativa más precisa para el problema de investigación.

Tabla 9 Comparación de las métricas de desempeño de los modelos predictivos

Modelo	MAE	RMSE	MAPE	WAPE
SARIMA (1,0,2)(0,1,1)[7]	6,23	9,56	**40,94 %**	41,90 %
Prophet	7,17	9,99	45,20 %	48,16 %
XGBoost optimizado	**6,14**	**9,46**	43,68 %	**41,00 %**

Fuente: Elaboración propia a partir de los resultados obtenidos en el notebook

Los resultados de la Tabla 9 muestran que XGBoost optimizado obtuvo el mejor rendimiento general y los MAE, RMSE y WAPE más bajos. Estos resultados indican que el modelo presenta la mayor precisión en la estimación del número diario de requerimientos y reduce el error absoluto respecto de los otros algoritmos evaluados.

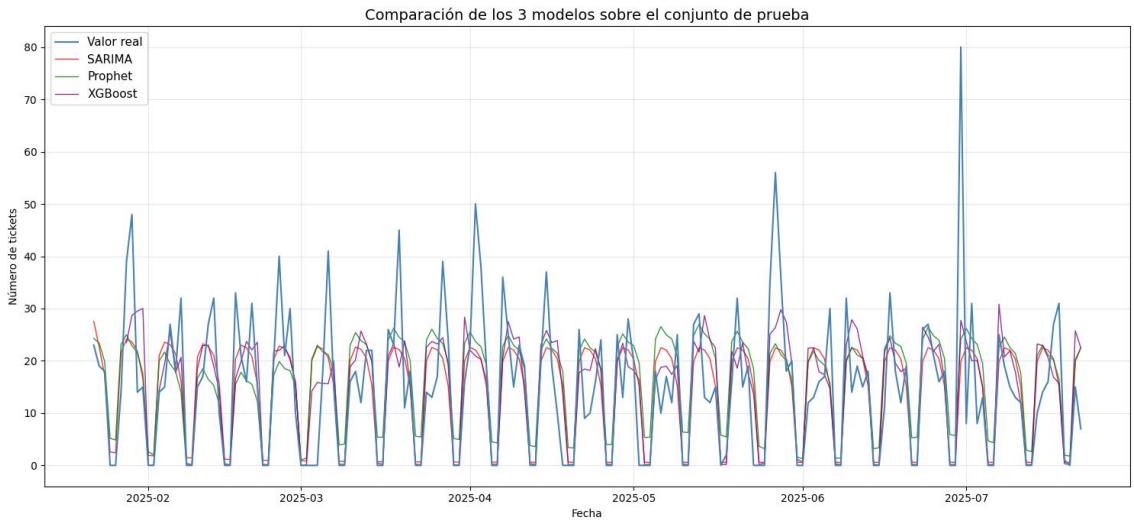
Por otro lado, SARIMA presenta el MAPE más bajo, lo que refleja un mejor ajuste al comportamiento de la serie temporal. Esto indica que la estructura estadística del modelo es adecuada para una serie con tendencia creciente y estacionalidad semanal claramente definida. Las diferencias respecto de XGBoost son pequeñas, pero el mejor rendimiento porcentual de SARIMA constituye un resultado importante desde un punto de vista metodológico.

En contraste, Prophet mostró los valores más altos en las cuatro métricas estudiadas. Aunque el modelo describió correctamente la tendencia y la estacionalidad, su enfoque menos conservador ante las fluctuaciones a corto plazo se tradujo en un mayor error en la estimación de los requerimientos diarios.

5.4.2. Comparación del comportamiento predictivo

Además del análisis cuantitativo, la comparación de los modelos se realizó mediante la evaluación visual de las predicciones generadas en el conjunto de prueba. Este análisis permitió identificar diferencias en la capacidad de cada algoritmo para reproducir el comportamiento observado en la serie temporal.

Gráfico 14 Comparación de las predicciones generadas por SARIMA, Prophet y XGBoost.



Fuente: Elaboración propia a partir del notebook

El gráfico 14 muestra que los tres modelos siguen la tendencia general de la demanda, pero difieren significativamente en cómo modelan las variaciones diarias. SARIMA muestra un comportamiento más estable y se ajusta bien a la estacionalidad semanal. Prophet muestra una trayectoria más suave y no se ve afectado por cambios a corto plazo.

Por otro lado, XGBoost reacciona mejor a las variaciones de la demanda y se aproxima a los valores observados durante el período de validación.

El análisis también muestra que ninguno de los tres modelos logró anticipar por completo los aumentos extraordinarios en algunos días del período de prueba. Estas son situaciones excepcionales que no forman parte del comportamiento normal de la serie y no pueden deducirse de la información histórica utilizada durante el entrenamiento. Este fue el caso de los tres algoritmos y representa una oportunidad para futuras investigaciones que incluyan variables externas como incidentes masivos, mantenimiento programado o cambios operativos en la organización.

5.4.3. Análisis complementario considerando días laborables

Además de la evaluación de los modelos predictivos, también se realizó un análisis complementario solo de los días laborables. El propósito en este experimento fue ver si excluir los fines de semana tenía un impacto en los modelos que previamente se había observado que tenían diferentes requisitos según el día de la semana.

Para este trabajo, se construyó una nueva serie temporal excluyendo los sábados y domingos. Luego, los datos se dividieron en 520 observaciones para entrenamiento y 130 para prueba. De esta manera, las observaciones se almacenan en el tiempo y los modelos se prueban, así como la serie temporal.

La estructura temporal también se cambió para que SARIMA incluyera el componente estacional. Dado que la nueva serie contenía solo los días laborables de cada semana, el modelo seleccionado para este experimento fue SARIMA (1,1,2)(1,0,1)[5]. Los resultados de la evaluación fueron MAE de 8.14, RMSE de 11.33, MAPE de 37.78% y WAPE de 39.12%.

La comparación con el experimento principal muestra que se requiere una interpretación conjunta de las métricas. En toda la serie, SARIMA tiene un MAPE de 40.94%, mientras que solo en los días laborables, este número se ha reducido a 37.78%. Desde un punto de vista porcentual, por lo tanto, hay una reducción en el error. Pero no se pudo ver tal mejora en las métricas absolutas ya que el MAE aumentó de 6.23 a 8.14 tickets.

También se realizó el mismo experimento con Prophet y XGBoost. De los resultados registrados en el notebook, usar solo días laborables resultó en valores de MAE de 9.60 para Prophet y 8.50 para XGBoost. El análisis en el notebook indica que la eliminación

de los fines de semana no condujo a una mejora general en las métricas absolutas de los modelos.

Tabla 10 Resultados del análisis complementario considerando días laborables

Modelo	MAE días laborables	Resultado relevante
SARIMA	8,14	MAPE = 37,78 %
Prophet	9,60	Incremento del error absoluto
XGBoost	8,50	Incremento del error absoluto

Fuente: Elaboración propia a partir de los resultados del notebook.

Los resultados muestran que excluir los fines de semana no proporcionó una mejora general en el rendimiento predictivo. Aunque SARIMA mostró una reducción en el MAPE, las métricas absolutas fueron menos favorables. Por esta razón, el análisis de los días laborables puede considerarse como una evaluación complementaria y no como una prueba para reemplazar la serie temporal completa utilizada en el experimento principal.

5.4.4. Análisis complementario de predicción por categoría con Prophet

Además de la predicción del volumen total de requisitos, el notebook también desarrolló un análisis complementario para evaluar el comportamiento de Prophet en diferentes tipos de tickets. Este experimento permitió ver si la estrategia utilizada para series globales puede aplicarse a series de menor volumen correspondientes a tipos específicos de requisitos.

Para desarrollar el análisis, se seleccionó individualmente las cinco categorías con al menos 200 registros, y las categorías restantes, cuyo número de observaciones estaba por debajo de este umbral, se agruparon bajo el nombre "Otros". Este grupo, según el notebook, consistía en 63 categorías con menos de 200 registros. Para cada serie, se utilizó un 80% de entrenamiento y un 20% de prueba.

Los resultados mostraron diferencias significativas en el rendimiento de Prophet dependiendo de la categoría analizada.

Tabla 11 Desempeño de Prophet por categoría

Categoría	Registros	MAE	RMSE	MAPE
/Computador/Otros Requerimientos	7.915	6,98	9,24	80,08 %
/Computador	1.821	1,28	1,87	72,07 %
/Computador/Configuración	1.005	1,28	1,73	92,36 %
/Computador/Instalación_desinstalación_actualización	838	0,80	1,38	68,88 %
/Vinculación – Promoción	242	0,93	1,26	**32,10 %**
Otras (63 categorías)	368	0,52	0,95	81,93 %

Fuente: Elaboración propia a partir de los resultados obtenidos en el notebook.

Las métricas absolutas disminuyen drásticamente en algunas categorías con menor volumen de requisitos. Pero este comportamiento no debe interpretarse automáticamente como una mayor precisión predictiva, ya que la escala de cada serie es diferente. Por ejemplo, el grupo "Otros" tuvo el MAE más bajo con 0.52, pero registró un MAPE de 81.93%. Esto subraya la necesidad de observar las métricas absolutas y relativas conjuntamente al comparar series con diferentes niveles de demanda.

Entre las categorías evaluadas, /Vinculación - Promoción tiene el MAPE más bajo de 242 registros, correspondiente al 32.10%. En contraste, /Computadora/Configuración tiene un MAPE de 92.36%, mientras que tiene un MAE de solo 1.28. Así que el resultado sugiere que el rendimiento de Prophet no solo depende del número de registros disponibles, sino también de la serie de registros para cada categoría.

Por lo tanto, el análisis por categoría es una buena extensión del experimento principal, pero sus resultados no justifican a Prophet como el modelo preferido para este nivel de desagregación.

5.4.5. Selección del modelo

La selección del modelo se realizó en base a los resultados en el conjunto de prueba y se consideraron las métricas MAE, RMSE, MAPE y WAPE. Estas evaluaciones revelaron

que los modelos SARIMA y XGBoost optimizado lograron los mejores resultados, y Prophet tuvo más errores en las métricas consideradas.

El modelo XGBoost optimizado logró un MAE de 6.14 tickets, RMSE de 9.46 tickets, MAPE de 43.68% y WAPE de 41.00%. De manera similar, el modelo SARIMA(1,0,2)(0,1,1)[7] alcanzó un MAE de 6.23 tickets, RMSE de 9.56 tickets, MAPE de 40.94% y WAPE de 41.90%. Prophet logró un MAE de 7.17 tickets, RMSE de 9.99 tickets, MAPE de 45.20% y WAPE de 48.16%. Según estos resultados, XGBoost tuvo los valores más bajos en tres de las cuatro métricas, mientras que SARIMA presentó el MAPE más bajo.

Sin embargo, la diferencia entre XGBoost y SARIMA fue pequeña. En el caso del MAE fue de 0.09 tickets, mientras que en el caso del RMSE fue de 0.10 tickets. Por lo tanto, no se observa una amplia superioridad de XGBoost sobre SARIMA, sino solo una ventaja cuantitativa marginal en MAE, RMSE y WAPE. Esto es consistente con lo que se ha observado en el notebook donde se nota que ambos modelos tienen un rendimiento muy cercano.

Para los propósitos de este estudio, se eligió el XGBoost optimizado como el modelo de referencia para el pronóstico final, ya que logró los valores más bajos de MAE, RMSE y WAPE en la evaluación final en el conjunto de prueba. Sin embargo, SARIMA sigue siendo un modelo competitivo que resulta en menos varianza que XGBoost y tiene el MAPE más bajo de los modelos evaluados.

Así, la selección de XGBoost es el resultado de los criterios cuantitativos establecidos para esta investigación y conjunto de datos, y no evidencia de la superioridad general del algoritmo sobre SARIMA.

5.5. Discusión de resultados y cumplimiento de objetivos

5.5.1. Discusión de los resultados obtenidos

El desarrollo de esta investigación mostró que las técnicas de análisis predictivo son una alternativa interesante para estimar el comportamiento futuro de los requerimientos registrados en un servicio de mesa de ayuda de TI. Desde la preparación y el análisis exploratorio de los datos, la modelización y la etapa de evaluación, se obtuvo que la información histórica de los tickets proporciona un patrón de tiempos consistente que puede aprovecharse mediante modelos de predicción. Por lo tanto, el estudio muestra que

la demanda de requerimientos no sigue un comportamiento aleatorio, sino que presenta una tendencia creciente y una estacionalidad semanal claramente identificable.

Uno de los principales hallazgos de la investigación es que la calidad de los datos incide de manera significativa en el rendimiento de los modelos predictivos. El proceso de preparación incluyó la corrección de inconsistencias en el archivo exportado del sistema de gestión de tickets, la normalización de los formatos de fecha y la creación de una serie temporal continua de 910 observaciones diarias, lo que garantizó que los tres modelos se entrenaran en la misma base de datos. Esto muestra que la calidad de los datos es clave para la predicción y confirma la relevancia de las primeras etapas propuestas por el método CRISP-DM.

Los resultados del análisis exploratorio también mostraron que el comportamiento de los requerimientos es, en general, semanal. La descomposición de la serie temporal mostró que la mayor parte de la variabilidad puede explicarse por la tendencia creciente y la estacionalidad del calendario laboral, mientras que los residuos corresponden principalmente a eventos extraordinarios vinculados a aumentos atípicos de la demanda. Este comportamiento justificó el uso de modelos especializados de series temporales y permitió interpretar los resultados obtenidos durante la fase de modelización.

Comparando los tres algoritmos desarrollados aquí y se observó que todos presentan diferencias en su capacidad predictiva. SARIMA, Prophet y XGBoost reproducen la tendencia de la serie temporal, pero cada uno tiene características específicas de la metodología. SARIMA muestra un comportamiento estable y es adecuado para la estacionalidad semanal; Prophet es realmente bueno en la tendencia y proporciona un buen intervalo de confianza para el análisis de incertidumbre; y XGBoost es más preciso porque utiliza el calendario, el desfase temporal y las estadísticas móviles durante el entrenamiento.

Para sintetizar los resultados, la Tabla 12 compara las principales métricas utilizadas para evaluar los modelos.

Tabla 12 Comparación de las métricas de desempeño de los modelos predictivos

Modelo	MAE	RMSE	MAPE	WAPE
SARIMA (1,0,2)(0,1,1)[7]	6,23	9,56	**40,94 %**	41,90 %
Prophet	7,17	9,99	45,20 %	48,16 %
XGBoost optimizado	**6,14**	**9,46**	43,68 %	**41,00 %**

Fuente: Elaboración propia a partir del notebook

Como se muestra en la Tabla 12, XGBoost obtuvo el mejor rendimiento general y registró los valores de MAE, RMSE y WAPE más bajos. SARIMA obtuvo el MAPE más bajo, lo que confirma su capacidad para representar la estructura porcentual de la serie temporal. Estas diferencias se redujeron, lo que indica que ambos modelos son buenas alternativas para los requisitos de pronóstico. Prophet tuvo un error ligeramente mayor, lo cual se debió al suavizado de las predicciones ante cambios repentinos en la demanda.

Como se puede ver en el gráfico 14, los tres modelos son capaces de reproducir la tendencia general de la serie temporal, pero XGBoost es mejor para predecir las variaciones diarias de la demanda. SARIMA es más estable según el horario semanal, mientras que Prophet es más suave. También está claro que ninguno de los modelos es capaz de predecir los aumentos extraordinarios durante el período de prueba, ya que no pueden aprenderse a partir de los datos de entrenamiento. Este es uno de los resultados más interesantes del estudio, ya que sugiere que debería considerar factores externos en trabajos futuros para predecir mejor los eventos excepcionales.

En general, los resultados muestran que el uso de técnicas de aprendizaje automático y modelos estadísticos de series temporales puede proporcionar estimaciones precisas de los requisitos futuros del servicio de asistencia. Además, la presencia de preparación de datos y series temporales para variables temporales son factores importantes para obtener modelos predictivos correctos en entornos de soporte tecnológico y permitir la toma de decisiones.

5.5.2. Cumplimiento del objetivo general

El objetivo general de la investigación fue desarrollar un modelo predictivo para estimar los requerimientos de un servicio de mesa de ayuda de TI, basado en el historial registrado por la organización. Los resultados obtenidos dejan claro que este objetivo se logró. En

el transcurso de la investigación, se desarrolló una serie temporal continua a partir del historial de datos de tickets, se empleó el método CRISP-DM para estructurar el análisis, se desarrollaron tres modelos predictivos (SARIMA, Prophet y XGBoost), y se evaluó su rendimiento según MAE, RMSE, MAPE y WAPE. Al comparar los resultados, XGBoost es el modelo con el mejor rendimiento general en el conjunto de datos analizado. La evidencia de los capítulos anteriores muestra que el modelo desarrollado permite estimar el comportamiento futuro de la demanda con un nivel adecuado de precisión para apoyar la planificación operativa de la mesa de ayuda y, así, cumplir con el propósito establecido al inicio de la investigación.

5.5.3. Cumplimiento de los objetivos específicos

Los objetivos específicos delineados al inicio de la investigación guiaron el desarrollo de las distintas fases de la metodología CRISP-DM, desde la comprensión y preparación de los datos hasta el modelado y la evaluación de los algoritmos predictivos. Los resultados mostraron que se alcanzó los tres objetivos específicos.

El primero de estos propósitos fue recopilar datos históricos sobre los requisitos del servicio de asistencia analizando el historial de tickets de la organización. Esta información se recopiló del servicio de asistencia para verificar que los datos estuvieran estructurados y que las variables necesarias para analizarlos estuvieran disponibles. Esta información se utilizó para desarrollar el estudio y constituyó la base histórica necesaria para construir la serie temporal empleada en el modelado.

El segundo objetivo de preprocesamiento de datos se logró mediante diversas actividades de preparación de datos. Estas incluyeron la limpieza de datos, la corrección de formatos de fecha, la eliminación de registros inválidos, la búsqueda de valores faltantes y el cambio de variables. Finalmente, se construyó una serie temporal continua y se generaron variables ajustadas al comportamiento histórico de los datos requeridos. Se aseguraron la calidad, la consistencia y la fiabilidad de los datos para el entrenamiento de modelos predictivos.

El tercer objetivo, a saber, desarrollar modelos predictivos basados en ciencia de datos y en modelos de aprendizaje automático, se logró mediante el modelado y la evaluación basados en CRISP-DM. Se desarrollaron tres modelos predictivos: SARIMA, Prophet y XGBoost con el mismo conjunto de datos. Los modelos fueron posteriormente evaluados mediante las métricas MAE, RMSE, MAPE y WAPE para una comparación objetiva de

su rendimiento predictivo. Los resultados mostraron que los modelos desarrollados son capaces de predecir el comportamiento futuro de los requisitos del servicio de asistencia y constituyen una herramienta para la planificación operativa y la toma de decisiones.

En general, los resultados mostraron que se cumplieron los tres objetivos principales y se logró el objetivo general de la investigación mediante una metodología estructurada basada en técnicas de ciencia de datos y de aprendizaje automático.

Tabla 13 Cumplimiento de los objetivos específicos

Objetivo específico	Evidencia obtenida	Resultado
Colectar datos históricos de requerimientos de la mesa de servicio	Recopilación del historial de tickets y validación de la estructura y la disponibilidad de los datos para el estudio.	Cumplido
Preprocesar los datos recopilados	Limpieza de datos, corrección de inconsistencias, transformación de variables temporales y construcción de la serie temporal utilizada en el modelado.	Cumplido
Desarrollar los modelos predictivos basados en metodologías de ciencia de datos y modelos de machine learning	Implementación y evaluación de los modelos SARIMA, Prophet y XGBoost mediante las métricas MAE, RMSE, MAPE y WAPE.	Cumplido

Fuente: Elaboración propia con base en los resultados de la investigación.

5.5.4. Aporte de la investigación

El objetivo principal de este estudio es demostrar que el análisis predictivo puede aplicarse para apoyar la planificación de un servicio de mesa de ayuda de TI basado en el volumen de solicitudes previsto.

En cuanto a la metodología, el estudio es una implementación completa de la metodología CRISP-DM al integrar las etapas de comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación y análisis de resultados en un problema real de gestión de servicios tecnológicos. Este enfoque proporciona una metodología reproducible que puede aplicarse en organizaciones con características similares para desarrollar modelos predictivos basados en registros históricos de tickets.

En el ámbito técnico, el estudio también muestra la utilidad de comparar modelos estadísticos tradicionales con algoritmos de aprendizaje automático. La implementación simultánea de SARIMA, Prophet y XGBoost permitió identificar las fortalezas y limitaciones de cada enfoque. También demostró que las variables basadas en el calendario, los retardos temporales y las estadísticas móviles pueden mejorar la capacidad predictiva de los métodos de aprendizaje automático. Este resultado será de interés para futuras investigaciones orientadas a pronosticar la demanda de servicios tecnológicos.

El estudio también ofrece una herramienta de apoyo para la toma de decisiones en la organización. La capacidad de anticipar el comportamiento futuro de la demanda también permite planificar los recursos a asignar, distribuir eficientemente la carga y minimizar la incertidumbre en las operaciones de la mesa de ayuda.

Figura 21 Pronóstico generado por el modelo XGBoost seleccionado.

```
# Generar pronóstico futuro de 30 días por cada categoría
fecha_inicio_pronostico = serie_diaria.index[-1] + pd.Timedelta(days=1)
print(f"Pronóstico desde: {fecha_inicio_pronostico.date()}")
# Diccionario para guardar pronósticos
pronosticos_30dias = {}
for cat in todas_categorias:
    # 1. Filtrar tickets de la categoría
    df_cat = df_categorias[df_categorias['Categoria_modelo'] == cat]
    # 2. Agrupar por día
    serie_cat = df_cat.groupby(df_cat['Fecha de registro'].dt.date).size()
    serie_cat.index = pd.to_datetime(serie_cat.index)
    serie_cat = serie_cat.asfreq('D').fillna(0)
    # 3. Preparar datos
    df_full_cat = pd.DataFrame({'ds': serie_cat.index, 'y': serie_cat.values})
    # 4. Reentrenar con TODA la data disponible (no solo train)
    modelo_full = Prophet(
        yearly_seasonality=True,
        weekly_seasonality=True,
        daily_seasonality=False,
        interval_width=0.95
    )
    modelo_full.fit(df_full_cat)
    # 5. Predecir los próximos 30 días
    futuro = modelo_full.make_future_dataframe(periods=30, freq='D')
    forecast = modelo_full.predict(futuro)
    # 6. Extraer solo el pronóstico futuro
    pred_futuro = forecast.set_index('ds')['yhat'].iloc[-30:].clip(lower=0).round().astype(int)
    pronosticos_30dias[cat] = pred_futuro

# Construir tabla final
tabla_pronostico = pd.DataFrame(pronosticos_30dias)
tabla_pronostico['TOTAL DIARIO'] = tabla_pronostico.sum(axis=1)

print("\n--- PRONÓSTICO DE LOS PRÓXIMOS 30 DÍAS POR CATEGORÍA ---\n")
print(tabla_pronostico.head(15))
```

Fuente: Elaboración propia a partir del notebook

Como se observa en la Figura 21, el modelo seleccionado reproduce adecuadamente el comportamiento esperado de la demanda y constituye una herramienta útil para apoyar la planificación de la capacidad operativa de la organización.

5.5.5. Limitaciones de la investigación

Aunque se obtuvieron resultados, el desarrollo de la investigación presentó algunas limitaciones que deben tenerse en cuenta al interpretar el alcance de los modelos predictivos que se ha adoptado.

La primera limitación es utilizar únicamente el historial de un único servicio de atención para el análisis. Como resultado, el comportamiento observado refleja las condiciones operativas de la organización en cuestión y podría ser distinto en instituciones con procesos de soporte distintos, niveles de demanda diferentes o características organizacionales distintas.

Otra limitación significativa es que no se contaba con variables externas que permitan dar cuenta de eventos extraordinarios. Los modelos se basaron principalmente en los datos históricos de la serie temporal y el calendario. Por lo tanto, los aumentos excepcionales derivados de eventos masivos, mantenimiento o cambios tecnológicos no pueden ser predichos por los algoritmos utilizados. Este comportamiento se observa al comparar las predicciones con los valores reales y constituye una de las principales causas de los errores en las evaluaciones.

Además, el horizonte temporal de entrenamiento se limitó a enero de 2023 a julio de 2025. Aunque esta cantidad de información fue suficiente para identificar patrones estacionales y crear buenos modelos predictivos, un historial más detallado mejoraría la capacidad de generalización y permitiría analizar ciclos operativos más largos.

Estas limitaciones no afectan los objetivos del estudio, pero sí permiten ampliar el alcance de trabajos futuros al incluir nuevas variables explicativas y conjuntos de datos provenientes de diferentes organizaciones.

5.5.6. Implicaciones para la organización

Los resultados de la investigación muestran que un modelo predictivo para estimar los requisitos puede servir como herramienta de apoyo para la gestión estratégica y operativa del servicio de mesa de ayuda de TI.

La capacidad de anticipar el volumen de requisitos permite una mejor planificación de los recursos humanos, lo que permite que la composición del personal se base en la demanda prevista y reduce la sobrecarga cuando la demanda es alta. La misma información del modelo también puede utilizarse para planificar el mantenimiento, los turnos y la asignación de tecnología, con el fin de optimizar y mejorar la eficiencia operativa.

Los resultados también muestran que comprender la tendencia y la estacionalidad de la demanda permite implementar estrategias preventivas para abordar períodos de alta carga de trabajo, lo que favorece los niveles de servicio y mejora la capacidad de respuesta a los requerimientos de los usuarios.

Finalmente, el estudio demuestra que el uso de la ciencia de datos en la gestión de servicios tecnológicos es una herramienta eficaz para mejorar la toma de decisiones basada en evidencia. Los modelos predictivos no reemplazan la experiencia del personal operativo, pero proporcionan datos cuantitativos relevantes para la planificación y la gestión eficiente de los recursos disponibles en el sistema.

En general, las implicaciones del modelo predictivo desarrollado son que tiene aplicaciones prácticas en la organización para mejorar la gestión de la mesa de ayuda y que constituye una base para la futura adopción de soluciones analíticas orientadas a la mejora continua de los procesos de soporte de TI.

6. Conclusiones y Recomendaciones

6.1. Conclusiones

- Prophet y SARIMA producen pronósticos visualmente similares con la diferencia clave de que Prophet refleja mejor la tendencia futura. Sin embargo, esta característica no se tradujo en un mejor desempeño cuantitativo en este caso particular. Prophet seguirá siendo evaluado para la predicción por categoría en la sección final del proyecto.
- Los tres modelos producen pronósticos válidos y coherentes con la estructura de la serie. XGBoost ofrece mayor amplitud en los picos, lo que puede ser preferible para planificación operativa de capacidad. SARIMA es el más predecible y estable. Prophet ofrece un equilibrio entre ambos. La elección del modelo final depende del criterio operativo: si se prioriza el error promedio (MAE), SARIMA y XGBoost empatan. Si se valoran la interpretabilidad y la cuantificación de la incertidumbre,

SARIMA tiene ventaja. Si se requiere flexibilidad para incorporar variables externas futuras, XGBoost es la mejor opción.

- Aunque el análisis laborable mejora marginalmente el MAPE (de 40.94% a 37.78% en SARIMA), las métricas absolutas (MAE y RMSE) empeoran porque los modelos pierden la ventaja predictiva de los días estructuralmente fáciles (fines de semana con cero tickets). Adicionalmente, Prophet muestra un deterioro significativo al perder su estacionalidad semanal natural.

6.2. Recomendaciones

- Se recomienda establecer un procedimiento periódico y automatizado para la recopilación de los datos generados por la mesa de servicios. Este proceso debe conservar la estructura de las variables, los formatos de fecha y las categorías de los requerimientos, con el propósito de contar con información actualizada y consistente para futuros entrenamientos de los modelos.
- Es conveniente documentar y estandarizar las actividades de preprocesamiento aplicadas al conjunto de datos, incluyendo la eliminación de registros duplicados, el tratamiento de valores nulos, la corrección de formatos, la depuración de registros inconsistentes y la construcción de la serie temporal. Esto permitirá reproducir el proceso y reducir posibles errores al incorporar nuevos datos.
- Se recomienda actualizar y volver a entrenar periódicamente los modelos SARIMA, Prophet y XGBoost, ya que el comportamiento de los requerimientos puede cambiar con el tiempo. La periodicidad del reentrenamiento deberá definirse de acuerdo con el volumen de nuevos registros y con los cambios observados en la demanda del servicio.
- Finalmente, se sugiere desarrollar un tablero de control que muestre los requerimientos históricos, los valores pronosticados, las métricas de evaluación y las diferencias entre la demanda real y la estimada. Este mecanismo facilitaría el seguimiento continuo del modelo y permitiría detectar oportunamente una disminución de su capacidad predictiva.

Bibliografía

AXELOS. (2019). ITIL Foundation: ITIL 4 Edition. TSO.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). Time Series Analysis: Forecasting and Control (5th ed.). Wiley.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step Data Mining Guide.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3rd ed.). OTexts.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International Journal of Forecasting, 22(4), 679–688.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning* (2nd ed.). Springer.

Mitchell, T. M. (1997). Machine Learning. McGraw-Hill.

Provost, F., & Fawcett, T. (2013). Data Science for Business. O'Reilly Media.

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37–45.