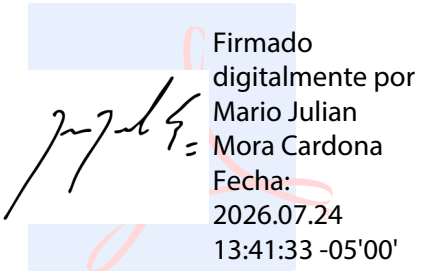
Santiago de Cali, 24 de Julio de 2026

Ingeniero:
Jorge Enrique Alvarez Patiño
Director Posgrados de Ingeniería
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ingeniería de Software, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado **Prototipo de identificación y clasificación de anomalías provenientes en Logfiles de la red CORE-MPLS de EMCALI para el evento de Potencia Óptica de la interfaz de comunicación**, el cual será realizado por el los estudiantes Daniel Cano Salgado con código 1872525 y William Zapata Castaño con código 8944479 pertenecientes al énfasis en Software, bajo la dirección del profesor Mario Julián Mora Cardona.

El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,



Firma	Firma	Firma
Daniel Cano Salgado	William Zapata Castaño	Mario Julián Mora Cardona
C.C. 1144097410 de Cali	C.C. 94311033 de Palmira	C.C. 94400220 de Cali

EMCALI
EICE - ESP

Santiago de Cali, 13 de Julio de 2026.

Señores

Pontificia Universidad Javeriana Cali.

Valle del Cauca - Cali.

Cordial Saludo.

Por medio de la presente, en mi calidad de Profesional Operativo III del Área de Conectividad e Internet, Unidad de Gestión de Plataformas de Servicios y Contenidos, Subgerencia Operativa, GERENCIA UNIDAD ESTRATÉGICA DE TIC de EMCALI E.I.C.E E.S.P., autorizo el uso y tratamiento de los datos de Logfiles de la red CORE-MPLS proporcionados para el desarrollo del proyecto de grado titulado "Prototipo de identificación y clasificación de anomalías provenientes en Logfiles de la red CORE-MPLS de EMCALI para el evento de Potencia Óptica de la interfaz de comunicación", realizado por los estudiantes de Magíster en Ingeniería de Software Daniel Cano Salgado (cod: 1872525) y William Zapata Castaño (cod: 8944479) de la Pontificia Universidad Javeriana Cali.

Los datos suministrados corresponden a registros de eventos (logs) generados por equipos de la red CORE-MPLS, los cuales han sido proporcionados exclusivamente con fines académicos e investigativos. Se garantiza que el tratamiento de dicha información se realizará conforme a las políticas de seguridad de la información y protección de datos personales vigentes, y que los datos serán utilizados únicamente para los propósitos descritos en el proyecto de grado mencionado.

Atentamente,

Ing. Jorge Andrés García Cano CC: 94330173
Profesional Operativo III – EMCALI E.I.C.E E.S.P.
Correo: jogarcia@emcali.com.co **Teléfono:** 602 899 8104

Prototipo de identificación y clasificación de anomalías provenientes en Logfiles de la red CORE-MPLS de EMCALI para el evento de Potencia Óptica de la interfaz de comunicación

Daniel Cano Salgado
William Zapata Castaño

Proyecto de grado entregado para obtener el título de
Magister en Ingeniería de Software

Dirigida por
MSc. Mario Julián Mora

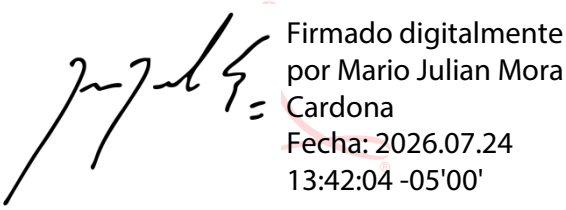
Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Maestría en Ingeniería de Software
Santiago de Cali
24 de Julio de 2026

Santiago de Cali, 24 de Julio de 2026.

Señores
Pontificia Universidad Javeriana Cali.
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software.
Cali.

Cordial Saludo.

Por medio de la presente hago constar que en mi calidad de director de trabajo de grado he revisado el proyecto titulado "Prototipo de identificación y clasificación de anomalías provenientes en Logfiles de la red CORE-MPLS de EMCALI para el evento de Potencia Óptica de la interfaz de comunicación" realizado por los estudiantes de Magister en Ingeniería de Software Daniel Cano Salgado (cod: 1872525) y William Zapata Castaño (cod: 8944479), el cual se encuentra terminado y considero que cumple con los requisitos para ser sustentado.
Atentamente,



MSc. Mario Julián Mora

Santiago de Cali, 24 de Julio de 2026.

Señores
Pontificia Universidad Javeriana Cali
Ph.D. Luisa Rincón
Directora Maestría en Ingeniería de Software
Cali.

Cordial Saludo.

Nos permitimos presentar a su consideración el proyecto de grado titulado "Prototipo de identificación y clasificación de anomalías provenientes en Logfiles de la red CORE-MPLS de EMCALI para el evento de Potencia Óptica de la interfaz de comunicación" con el fin de cumplir con los requisitos exigidos por la Universidad y para que sea sometido a revisión del jurado y cumpla su aprobación, para conseguir posteriormente el título de Magister en Ingeniería de Software.

Atentamente,

Daniel Cano Salgado

Daniel Cano Salgado
Código: 1872525

William Zapata Castaño
Código: 8944479

Ficha Resumen
Trabajo de Grado Maestría en Ingeniería de Software

TÍTULO: Prototipo de identificación y clasificación de anomalías provenientes en Logfiles de la red CORE-MPLS de EMCALI para el evento de Potencia Óptica de la interfaz de comunicación

1. Énfasis: Ingeniería de Software
2. Área de trabajo: Telecomunicaciones
3. Tipo de proyecto: Aplicado
4. Estudiantes: Daniel Cano Salgado, William Zapata Castaño
5. Correo electrónico: daniel29@javerianacali.edu.co, wzcjave3449@javerianacali.edu.co
6. Director: Mario Julián Mora Cardona
7. Vinculación del director: Planta
8. Correo electrónico del director: mariomora@javerianacali.edu.co
9. Empresa que lo avala: Empresas Municipales de Cali – EMCALI E.I.C.E. E.S.P.
10. Palabras clave: AIOps, Syslog, Series Temporales, LSTM, Predicción de Fallas, Ingeniería de Software
11. ODS que aplica al proyecto (Agenda 2030): ODS 9 – Industria, Innovación e Infraestructura
12. Fecha de inicio: 15 de enero de 2026
13. Resumen: El presente proyecto desarrolló un prototipo AIOps para la predicción anticipada de eventos de degradación de potencia óptica en las interfaces de la red CORE-MPLS de EMCALI E.I.C.E. E.S.P., mediante el análisis de registros de sistema (syslogs) utilizando técnicas de aprendizaje automático y aprendizaje profundo. La problemática central radica en que la gestión tradicional del Centro de Operaciones de Red (NOC) opera de forma reactiva, detectando fallas en las interfaces ópticas solo después de producirse, con el consecuente impacto en los Acuerdos de Nivel de Servicio (SLA) y en la continuidad de los servicios de telecomunicaciones. El objetivo general fue construir un prototipo funcional capaz de clasificar anomalías en archivos de log de la red CORE-MPLS de EMCALI para apoyar la toma de decisiones proactiva del equipo de soporte técnico. La metodología siguió el marco de referencia ASUM-DM (Analytics Solutions Unified Method for Data Mining), estructurado en ocho fases que abarcaron desde la comprensión del negocio y la ingeniería de características hasta la construcción de un prototipo de reporte operativo. A partir de aproximadamente 192,000 líneas de syslogs crudos se construyeron 67,187 ventanas temporales de 60 segundos con 14 características de ingeniería, incluyendo estadísticos móviles de eventos, telemetría

óptica (potencia Rx/Tx en dBm) y derivadas temporales de anomalías. Se evaluaron cinco modelos (ARIMA, Regresión Logística, XGBoost, Isolation Forest y LSTM) bajo un esquema estricto de separación temporal que garantizó la ausencia de filtración de información futura (data leakage). Los resultados demostraron la superioridad del modelo LSTM para capturar la dinámica temporal secuencial de los eventos de red, alcanzando un Recall del 90 % sobre el conjunto de prueba, detectando 135 de 150 eventos positivos. En la interfaz con mayor densidad de telemetría óptica (XGigabitEthernet3/0/0), el modelo obtuvo un F1-Score del 61.54 % (Precisión: 50 %, Recall: 80 %), proporcionando una ventana de anticipación aproximada de una hora para la ejecución de acciones preventivas. El prototipo fue entregado como un paquete Python modular con interfaz de línea de comandos (CLI) que genera reportes operativos en formato HTML, demostrando la viabilidad técnica del enfoque propuesto y aportando evidencia preliminar de su viabilidad operativa sobre el escenario evaluado.

Agradecimientos

A Dios, por ser el motor en cada etapa de nuestras vidas, por la sabiduría, la paciencia y la fortaleza brindadas para culminar con éxito este peldaño profesional.

A la Pontificia Universidad Javeriana Cali y al cuerpo docente de la Maestría en Ingeniería de Software, por abrirnos sus puertas, compartir su valioso conocimiento y fomentar en nosotros el rigor científico necesario para el desarrollo de esta investigación.

A nuestro director de tesis, por su guía incondicional, su paciencia y sus acertadas orientaciones. Su experiencia no solo enderezó el rumbo técnico de este proyecto en los momentos de incertidumbre, sino que nos enseñó el verdadero valor de la persistencia en la ingeniería.

A nuestras familias, por ser nuestro soporte incondicional. A nuestros padres, parejas y seres queridos, quienes entendieron cada hora de ausencia, cada desvelo frente al computador y cada taza de café dedicada a estructurar este trabajo. Este logro fruto de la constancia es tan suyo como nuestro.

Finalmente, a nuestros compañeros y amigos, con quienes compartimos debates, código y largas jornadas de aprendizaje. Su apoyo y camaradería hicieron este camino de maestría una experiencia mucho más enriquecedora.

Resumen

El presente proyecto aborda el diseño y desarrollo de un prototipo de software basado en Inteligencia Artificial para las Operaciones de TI (AIOps), orientado a la predicción anticipada de estados de indisponibilidad (*DOWN*) en interfaces de red ópticas pertenecientes a la infraestructura de telecomunicaciones de la red CORE-MPLS de EMCALI E.I.C.E. E.S.P. El enfoque convencional en la gestión de redes de datos suele ser reactivo o limitarse a umbrales estáticos que notifican fallas una vez que el servicio se ha degradado. Para mitigar esta problemática, se implementó un *pipeline* de procesamiento de datos automatizado y estructurado en ocho fases secuenciales siguiendo el marco ASUM-DM (*Analytics Solutions Unified Method for Data Mining*), el cual ingesta registros de eventos masivos (*syslogs* del orden de ∼192,000 líneas), extrae eventos de potencia óptica y, cuando están disponibles, métricas de telemetría RX/TX mediante expresiones regulares (Regex), y remueve la redundancia topológica mediante la exclusión estricta de subinterfaces lógicas, construyendo 67,187 ventanas temporales de 60 segundos con 14 características de ingeniería. El núcleo del sistema evalúa y compara cinco enfoques: métodos estadísticos (ARIMA), aprendizaje automático clásico (Regresión Logística, XGBoost, Isolation Forest) y aprendizaje profundo mediante una red neuronal recurrente LSTM (Long Short-Term Memory) construida sobre PyTorch. Para garantizar la viabilidad y aplicabilidad operativa real en el Centro de Operaciones de Red (NOC), el diseño del *dataset* impone una restricción de aislamiento temporal ($Gap \geq 60$ segundos) que previene la filtración de datos del futuro (*data leakage*), apuntando a un horizonte de predicción (*Lookahead*) de una hora. Los resultados, validados de forma honesta mediante la calibración de umbrales exclusivamente sobre el conjunto de entrenamiento, demuestran la superioridad del modelo LSTM para capturar la dinámica temporal secuencial: a nivel global alcanzó un *Recall* del 90 %, detectando 135 de 150 eventos positivos en el conjunto de prueba. Sobre la interfaz crítica `XGigabitEthernet3/0/0` del host `COLO_S07706_0101`, obtuvo una precisión del 50,0 %, un *Recall* del 80,0 % y un $F1$-Score de 61,54 %. Esto demuestra que la solución es capaz de anticipar 4 de cada 5 eventos de degradación de potencia óptica, proveyendo una ventana de acción útil para la ejecución de políticas de contingencia operativa. El artefacto final fue empaquetado en una interfaz de línea de comandos (CLI) ligera que genera reportes en formato HTML, verificando su compatibilidad técnica con entornos de producción estándar (ejecución sobre CPU, empaquetado autónomo).

Palabras Clave: AIOps, Syslog, Series Temporales, LSTM, Predicción de Fallas, Ingeniería de Software.

Abstract

This research addresses the design and development of a software prototype based on Artificial Intelligence for IT Operations (AIOps), aimed at the early prediction of unavailability states (*DOWN*) in optical network interfaces within the CORE-MPLS telecommunications infrastructure of EMCALI E.I.C.E. E.S.P. Conventional approaches in network management are predominantly reactive or limited to static thresholds that trigger alarms only after service degradation has occurred. To mitigate this issue, an automated data processing pipeline was implemented, structured into eight sequential phases following the ASUM-DM framework (*Analytics Solutions Unified Method for Data Mining*). This pipeline ingests massive event logs (∼192,000 syslog lines), extracts optical power events and, when available, RX/TX telemetry metrics using regular expressions, and removes topological redundancy through the strict exclusion of logical subinterfaces, constructing 67,187 temporal windows of 60 seconds each with 14 engineered features. The core inference system evaluates and compares five approaches: statistical methods (ARIMA), classical machine learning (Logistic Regression, XGBoost, Isolation Forest), and deep learning using a Long Short-Term Memory (LSTM) recurrent neural network built on PyTorch. To guarantee operational viability within a Network Operations Center (NOC), the dataset design enforces a temporal isolation constraint ($Gap \geq 60$ seconds) to prevent data leakage, targeting a one-hour prediction horizon (*Lookahead*). The results, robustly validated by calibrating decision thresholds exclusively on the training set, demonstrate the superiority of the LSTM model in capturing sequential temporal dynamics: globally it achieved a recall of 90 %, detecting 135 out of 150 positive events in the test set. On the critical interface `XGigabitEthernet3/0/0` of host `COLO_S07706_0101`, it achieved a precision of 50,0 %, a recall of 80,0 %, and an $F1$-score of 61,54 %. This proves that the solution can anticipate 4 out of 5 optical power degradation events, providing a valuable time window for executing operational contingency policies. The final artifact was packaged into a lightweight command-line interface (CLI) that generates HTML reports, verifying its technical compatibility with standard production environments (CPU execution, self-contained packaging).

Keywords: AIOps, Syslog, Time Series, LSTM, Failure Prediction, Software Engineering.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En el marco de la gestión de un ecosistema de telecomunicaciones, la revisión de los archivos de registro de eventos (*logfiles*) ocupa un lugar prioritario para verificar constantemente el estado de los equipos de red, de tal manera que se puedan encontrar indicios de anomalías que impliquen una intervención preventiva o correctiva. Estas decisiones son tomadas por el equipo humano de soporte técnico y en muchas ocasiones se ven superados por el volumen de datos provenientes en los *logfiles*. Los equipos de conmutación y enrutamiento de redes modernas generan cientos de miles de líneas de registro por período de operación, convirtiendo el análisis manual en una tarea inviable tanto por la magnitud de los datos como por la complejidad de las relaciones temporales entre los eventos registrados.

En este contexto, el paradigma de Inteligencia Artificial para las Operaciones de TI (AIOps) (Gartner, 2017) ha emergido como un enfoque prometedor para la automatización del análisis operativo en infraestructuras de telecomunicaciones. AIOps combina técnicas de aprendizaje automático, procesamiento de grandes volúmenes de datos y análisis de series temporales para detectar anomalías, predecir fallas y apoyar la toma de decisiones operativas de manera proactiva, superando las limitaciones de los enfoques reactivos basados en umbrales estáticos.

El presente trabajo de grado abordó esta problemática mediante un prototipo AIOps orientado a la predicción anticipada de estados de indisponibilidad (*DOWN*) en interfaces ópticas de la red de EMCALI E.I.C.E. E.S.P. El proyecto se desarrolló durante el período enero–junio de 2026 en el contexto de la Maestría en Ingeniería de Software de la Pontificia Universidad Javeriana Cali, siguiendo la metodología ASUM-DM (*Analytics Solutions Unified Method for Data Mining*). Se implementó un *pipeline* automatizado de ocho fases que transformó registros de eventos masivos (*syslogs*) en series temporales estructuradas, sobre las cuales se evaluaron cinco modelos de clasificación (ARIMA, Regresión Logística, XGBoost, Isolation Forest y LSTM), seleccionando una red neuronal recurrente LSTM (*Long Short-Term Memory*) (Hochreiter and Schmidhuber, 1997) como modelo final con base en su desempeño superior en la detección anticipada de fallas.

A continuación se presenta la estructura del documento. En el presente capítulo se expone la definición del problema, los objetivos, el alcance, la justificación, la metodología adoptada y un resumen de los resultados obtenidos. En el Capítulo 2 se desarrolla el marco de referencia, incluyendo las bases teóricas y el estado del arte. El Capítulo 3 detalla el desarrollo del proyecto: diseño, arquitectura e implementación del *pipeline* y el prototipo. El Capítulo 4 expone el diseño de la evaluación, los resultados experimentales y su análisis. El Capítulo 5 presenta las conclusiones del proyecto. El Capítulo 6 documenta los riesgos éticos identificados y las medidas de mitigación adoptadas. Finalmente, los Capítulos 7 y 8 recogen las lecciones aprendidas y los trabajos futuros, respectivamente.

1.1. Definición del problema

1.1.1. Planteamiento del problema

La empresa EMCALI enfrentaba desafíos significativos en la gestión de su red de servicios, particularmente en lo referente a la detección y resolución oportuna de problemas que afectaban la calidad y disponibilidad de los servicios de Internet, Televisión y Telefonía ofrecidos a clientes corporativos y residenciales. La carencia de herramientas adecuadas para el monitoreo proactivo y la resolución eficiente de eventos en la Red MultiServicios (RMS, ver Figura 1.1), generaba retrasos sustanciales en la atención de incidentes, lo cual impactaba directamente la satisfacción del cliente y la percepción de la marca en el mercado.

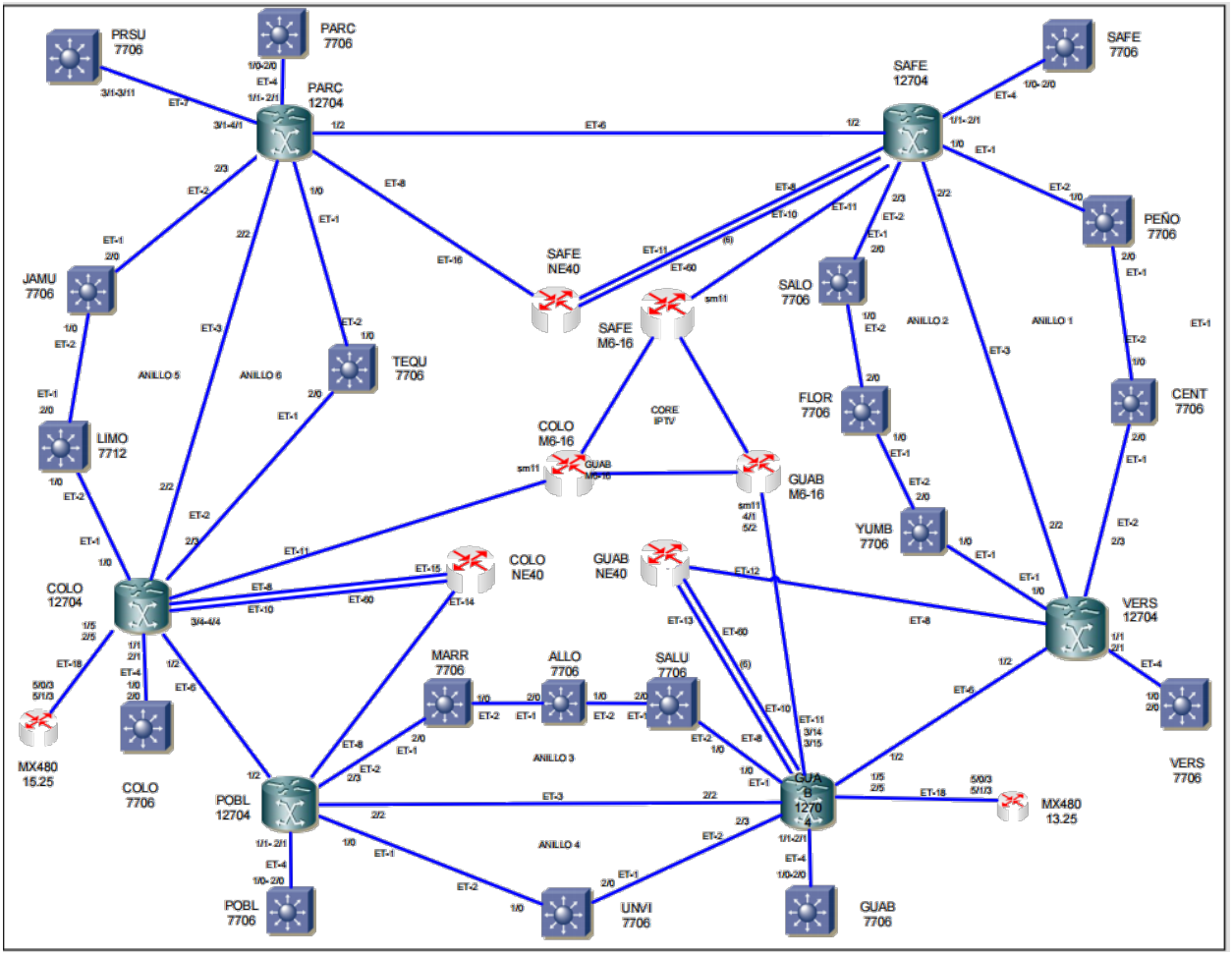


Figura 1.1: Esquema general de la Red MultiServicios (RMS) de EMCALI.

Esta situación se traducía en una disminución en la calidad percibida del servicio, erosionando la confianza del cliente y debilitando la reputación de la empresa. El personal técnico, encargado

de abordar los problemas de la red, se veía limitado por la falta de una infraestructura tecnológica que le permitiera identificar y responder de manera ágil a los eventos y daños presentados en el CORE-MPLS de la RMS. Los tiempos de respuesta prolongados y la dificultad para anticipar y prevenir problemas deterioraban la calidad de la atención ofrecida, incrementando la insatisfacción y generando una mayor demanda de soporte. En consecuencia, la empresa no solo enfrentaba un impacto negativo en la fidelidad de sus clientes, sino también una disminución en su eficiencia operativa, al requerir recursos adicionales para la resolución de problemas que podrían haberse evitado o mitigado con un enfoque más proactivo.

En este contexto, la evolución continua de la tecnología y el incremento sostenido en la generación de datos por parte de los sistemas de telecomunicaciones planteaban un desafío crítico: la necesidad de tomar decisiones informadas y eficientes en tiempo real. Los *logfiles*, que registran todas las acciones y transacciones dentro de los sistemas de telecomunicaciones, representan una fuente invaluable de información para la gestión estratégica de la red. Sin embargo, la analítica de grandes volúmenes de datos y la extracción de patrones significativos requerían habilidades y recursos especializados que no estaban disponibles en la infraestructura de EMCALI.

El desarrollo de un prototipo para la identificación y clasificación de anomalías provenientes de los *logfiles* de la red CORE-MPLS, específicamente en relación con los eventos de potencia óptica de la interfaz de comunicación, se planteó como una solución estratégica. El proyecto buscó dotar al equipo humano de soporte con una herramienta que le permitiera agilizar la toma de decisiones, anticipar fallas y realizar intervenciones con mayor precisión y oportunidad. De esta manera, se contribuyó a mejorar la eficiencia operativa, optimizar la calidad del servicio y fortalecer la rentabilidad del ecosistema de telecomunicaciones de la empresa.

1.1.2. Identificación del problema

La falta de herramientas de análisis automatizado en la Red Multiservicios (RMS) de la empresa EMCALI causaba un incremento en los tiempos de respuesta para la atención y reparación de servicios de telecomunicaciones, afectando la eficiencia operativa y la experiencia del cliente.

1.1.3. Contextualización

¿Cómo afecta la detección anticipada de eventos a la eficiencia operativa, la calidad del servicio y la experiencia del cliente?

1.1.4. Delimitación del problema

Dentro del universo de eventos registrados en los *logfiles* del CORE-MPLS, se identificaron los siguientes tipos de eventos susceptibles de evaluación:

- Afectación de servicios: Voz, Internet, TV.
- Rupturas de fibra óptica.
- Fluctuaciones o ausencia de potencia eléctrica.

- Retardo por deslizamiento de canales de Voz.
- *Loops* físicos en la red.
- Aumento en la temperatura de los equipos.
- Cambios en el estado de las interfaces ópticas UP/DOWN.
- Saturación de enlaces físicos.

De esta lista, el presente proyecto se delimitó específicamente al análisis de los **cambios en el estado de las interfaces ópticas UP/DOWN**, a partir de los **eventos de potencia óptica** extraídos de los registros *syslog* de los equipos de la red; la telemetría de potencia óptica (`rx_power_dbm`) se incorporó como atributo complementario. Esta delimitación respondió a la disponibilidad de datos estructurados en los *logfiles* y a la criticidad que representan las caídas de interfaces ópticas para la continuidad de los servicios de telecomunicaciones prestados por EMCALI.

1.1.5. Sistematización

Las preguntas de investigación que orientaron el desarrollo del proyecto fueron las siguientes:

- ¿Cómo se pueden identificar y extraer automáticamente los eventos relevantes de potencia óptica a partir de archivos *logfile* masivos generados por los equipos de la red CORE-MPLS?
- ¿Qué modelos de aprendizaje automático resultan más adecuados para la predicción anticipada de fallas ($DOWN$) en interfaces de red ópticas, considerando el desbalance extremo de clases presente en los datos operativos?
- ¿Cómo afecta la detección anticipada de eventos a la eficiencia operativa, la calidad del servicio y la experiencia del cliente?

1.2. Objetivos del proyecto

1.2.1. Objetivo General

Desarrollar un prototipo de clasificación de anomalías provenientes en LOGFILES de la red CORE-MPLS de EMCALI E.I.C.E. E.S.P. que sirva de insumo para la toma de decisiones del equipo de soporte técnico para el evento de Potencia Óptica.

1.2.2. Objetivos Específicos

1. Analizar los archivos de registro LOGFILES de los equipos de la red CORE-MPLS de EMCALI, identificando los eventos asociados a la potencia óptica de las interfaces de comunicación.
2. Definir los criterios que permiten identificar eventos anómalos relacionados con la potencia óptica dentro de los LOGFILES.
3. Seleccionar modelos de clasificación de anomalías adecuados para el análisis de los eventos de potencia óptica registrados en los LOGFILES.
4. Evaluar el desempeño de los modelos mediante métricas de precisión, recall y F1-score y seleccionar aquel que presente los mejores resultados para la clasificación de anomalías de potencia óptica.
5. Desarrollar un prototipo de aplicación que implemente el modelo de clasificación seleccionado para la identificación de anomalías en los LOGFILES.

1.3. Delimitaciones y alcances

El alcance del presente proyecto comprende el diseño y desarrollo de un prototipo de aprendizaje automático supervisado orientado a la identificación y clasificación de anomalías a partir del análisis de LOGFILES generados por la red **CORE-MPLS** de la empresa **EMCALI E.I.C.E. E.S.P.** El proyecto se circunscribe únicamente a esta organización y a su infraestructura de red, sin considerar escenarios pertenecientes a otras empresas o redes de telecomunicaciones.

El prototipo se enfoca de manera específica en el análisis de los ***eventos de potencia óptica*** de las interfaces de la red CORE-MPLS, los cuales serán empleados como insumo principal de entrada para los procesos de preprocesamiento de datos, validación y evaluación de los modelos que serán utilizados. En consecuencia, no se abordará el análisis de otras variables operativas o de desempeño de la red, tales como tráfico, latencia, errores de transmisión, disponibilidad de enlaces u otras métricas técnicas, aun cuando estas se encuentren presentes en los LOGFILES.

Adicionalmente, se asume que los eventos de potencia óptica son representativos y suficientes para identificar patrones relevantes de anomalías dentro del contexto del estudio, y que se contará con los recursos computacionales necesarios para el desarrollo y evaluación del modelo durante el período establecido para el trabajo de grado.

Se contempla el diseño, implementación y validación de un prototipo en entorno controlado. **No incluye la integración directa con la infraestructura productiva de EMCALI ni la puesta en operación en ambientes de misión crítica.** El empalme con sistemas existentes se plantea únicamente como diseño conceptual para futuras fases de implementación institucional.

1.4. Justificación del trabajo de grado

El incremento sostenido en la generación de datos por parte de los sistemas de telecomunicaciones ha convertido la toma de decisiones informadas en un desafío crítico. Los *logfiles*, que registran todas las acciones y transacciones dentro de los sistemas de telecomunicaciones, constituyen una fuente invaluable de información para la toma de decisiones estratégicas. Sin embargo, la analítica de grandes cantidades de datos y la extracción de *insights* significativos requiere habilidades y recursos especializados que no se encontraban disponibles en la infraestructura operativa de EMCALI.

Con el desarrollo de este proyecto, se pretendió brindar una herramienta con la que el equipo humano de soporte no contaba previamente, con el objetivo de agilizar la toma de decisiones y realizar intervenciones con mayor anticipación, mejorando la eficiencia y la rentabilidad del ecosistema de telecomunicaciones.

Desde la perspectiva de la **eficiencia operativa**, el enfoque convencional en la gestión de redes de datos era predominantemente reactivo, limitándose a umbrales estáticos que notificaban fallas una vez que el servicio ya se había degradado. Un sistema de alertas tempranas con un horizonte de predicción de una hora, como el desarrollado en este proyecto, buscó proporcionar una ventana de acción útil para la ejecución de políticas de contingencia operativa (desvío de tráfico, balanceo de cargas de red o programación de intervenciones en campo antes de la pérdida total del servicio), habilitando una gestión proactiva frente al esquema reactivo tradicional, en el que las fallas se detectaban únicamente una vez producida la degradación del servicio.

Desde la perspectiva **técnica y académica**, el proyecto contribuyó al conocimiento aplicado en la intersección entre la ingeniería de software y las telecomunicaciones, demostrando la viabilidad de implementar soluciones AIOps sobre registros de eventos reales de equipos de red en producción. La comparación sistemática de cinco modelos (ARIMA, Regresión Logística, XGBoost, Isolation Forest y LSTM) bajo condiciones idénticas y la documentación rigurosa de las decisiones de modelado adoptadas constituyeron aportes relevantes para investigaciones futuras en el dominio. Adicionalmente, la implementación de un *pipeline* de ocho fases con correcciones metodológicas explícitas, como la eliminación de *look-ahead bias* en el cálculo de *features* y la calibración de umbrales exclusivamente sobre datos de entrenamiento para evitar *data leakage*, estableció un precedente de rigor experimental aplicable a proyectos similares.

Desde la perspectiva **social**, la mejora en la capacidad de anticipación de fallas redundó en un potencial beneficio directo para los usuarios finales de los servicios de Internet, Televisión y Telefonía de EMCALI, al contribuir a la reducción de los tiempos de indisponibilidad y a la mejora en la experiencia del cliente. En un contexto donde EMCALI presta servicios de telecomunicaciones a clientes tanto corporativos como residenciales, la continuidad del servicio tiene un impacto directo en la productividad empresarial y en la calidad de vida de la comunidad atendida.

Desde la perspectiva **económica**, la detección anticipada de fallas permitió proyectar una reducción en los costos asociados a intervenciones correctivas de emergencia, despacho no planificado de cuadrillas de campo y penalizaciones contractuales por incumplimiento de acuerdos de nivel de servicio (SLA). La transición de un modelo reactivo a uno predictivo representó una optimización en la asignación de recursos operativos del área de soporte técnico.

1.5. Metodología de la investigación

La metodología adoptada para el desarrollo del proyecto se basó en el marco **ASUM-DM** (*Analytics Solutions Unified Method for Data Mining*), el cual proporcionó una guía estructurada para la planificación, ejecución y documentación del proyecto de analítica y minería de datos. ASUM-DM se centra en fases claramente definidas (comprensión del negocio, comprensión de los datos, preparación, modelado, evaluación y despliegue) que permitieron mantener un control riguroso sobre cada etapa del ciclo de vida del proyecto.

La Figura 1.2 presenta la estructura de las fases ejecutadas dentro del marco metodológico adoptado. Aunque dichas fases se representan de manera organizada, su ejecución no fue estrictamente secuencial, sino iterativa e incremental.

En particular, varias de las actividades se desarrollaron mediante ciclos de retroalimentación continua. Por ejemplo, la construcción y evaluación de modelos de aprendizaje automático implicó múltiples iteraciones, en las cuales se ajustaron parámetros, se redefinieron características y se recalibraron umbrales de decisión con base en los resultados obtenidos.

De igual manera, el diseño experimental y la validación de los modelos requirieron revisiones sucesivas para evitar sesgos y mejorar la capacidad de generalización del sistema. Este enfoque iterativo permitió refinar progresivamente el desempeño del prototipo y garantizar la coherencia entre los objetivos específicos y los resultados alcanzados.

A continuación, se describe cómo se ejecutó cada fase y su articulación con los objetivos específicos del proyecto.

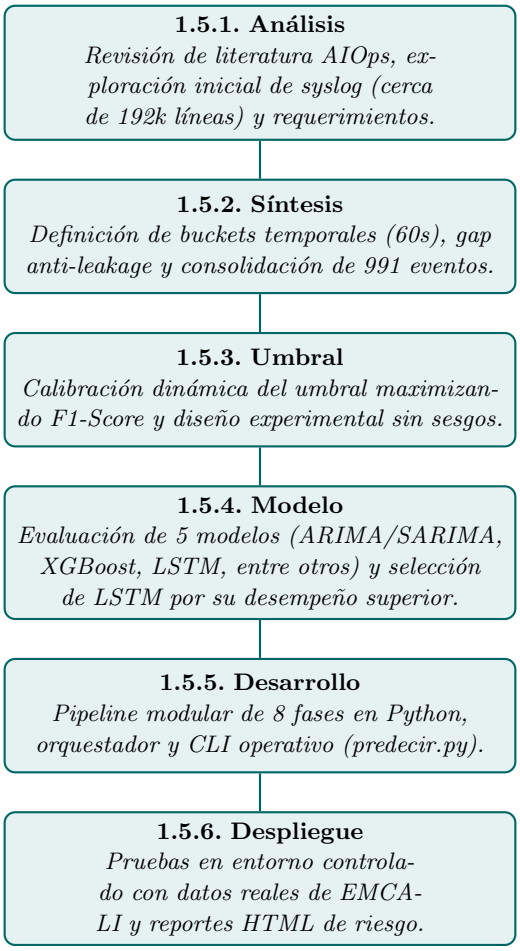


Figura 1.2: Estructura secuencial y flujo de actividades de la metodología de investigación basada en el marco ASUM-DM para la red CORE-MPLS de EMCALI.

1.5.1. Análisis

Esta fase se articuló con el Objetivo Específico, en adelante, **OE-1** (analizar los LOGFILES identificando eventos de potencia óptica). Se llevaron a cabo las siguientes actividades:

- Revisión de la literatura sobre el uso de inteligencia artificial en la identificación y clasificación de anomalías en LOGFILES de redes de telecomunicaciones. Se analizaron trabajos previos en AIOps (Gartner, 2017), análisis de logs con aprendizaje automático (Hussein and Répás, 2024; Zhang et al., 2019) y detección de anomalías en redes (Gómez et al., 2020; López-Avila et al., 2019).

- Análisis de la situación actual del equipo de soporte técnico de EMCALI y los desafíos que enfrentaba al procesar los *logfiles* de la red CORE-MPLS.

- Identificación de los objetivos y requerimientos del proyecto, según los objetivos generales y específicos del anteproyecto aprobado.

- Obtención y exploración inicial de los archivos de *syslog* generados por los equipos de la red CORE-MPLS, con un volumen aproximado de ∼192,000 líneas de registro.

1.5.2. Síntesis

Esta fase se articuló con los **OE1** y **OE2** (identificar eventos y definir criterios de anomalía). Se realizaron las siguientes actividades:

- Definición de los requerimientos del proyecto, incluyendo la naturaleza de los datos (registros *syslog* de los equipos de red, múltiples fuentes de parsing: `key=value`, texto libre y expresiones regulares (Regex)), los eventos objetivo (cambios de estado *DOWN* en interfaces ópticas) y los modelos candidatos a evaluar.

- Establecimiento de los parámetros para la evaluación de los modelos de clasificación: Precision, Recall, F1-Score y PR-AUC como métrica principal para datos desbalanceados.

- Diseño de la estructura temporal del dataset: *buckets* de 60 segundos como unidad temporal, con separación temporal ($Gap \geq 60$ segundos) entre los datos de observación y el horizonte de predicción de una hora para evitar filtración de información temporal (*data leakage*).

- Síntesis de los resultados del análisis exploratorio: caracterización del conjunto de interfaces monitoreadas, consolidación de los eventos de potencia óptica, y evidencia del fuerte desbalance de clases y de la distribución desigual de la telemetría óptica entre interfaces. El detalle cuantitativo se presenta en el Capítulo 3.

1.5.3. Umbral

Esta fase se articuló con el **OE2** (definir criterios para identificar eventos anómalos) y el **OE4** (evaluar desempeño con métricas). Se establecieron los siguientes criterios:

- Definición de las métricas de evaluación: Precision, Recall, F1-Score y PR-AUC. Se descartó el uso de Accuracy como métrica principal debido al desbalance extremo de clases. El código fuente del proyecto, incluyendo los experimentos de evaluación, se encuentra disponible en el repositorio público del proyecto (Cano Salgado and Zapata Castaño, 2026)[1].
- Definición de un umbral de clasificación dinámico, calibrado automáticamente sobre el conjunto de entrenamiento mediante la maximización del F1-Score en un subconjunto de validación (último 20 % del conjunto de entrenamiento). Este enfoque permitió que el umbral se adaptara a la distribución de probabilidades de cada modelo, en lugar de utilizar un valor fijo arbitrario. La calibración se realizó exclusivamente sobre datos de entrenamiento para evitar sesgo por *data leakage*.
- Definición del diseño experimental con separación temporal ($Gap \geq 60$ segundos) entre los datos de observación y el horizonte de predicción, asegurando que no se utilizara información futura para predecir (*look-ahead bias*).
- Establecimiento del criterio de aceptación: el modelo seleccionado debía demostrar capacidad de separación de probabilidades entre clases positiva y negativa en al menos una interfaz con telemetría óptica densa.

1.5.4. Modelo

Esta fase se articuló con el **OE3** (seleccionar modelos adecuados) y el **OE4** (evaluar desempeño). Se ejecutaron las siguientes actividades:

- Selección de cinco modelos de clasificación que cubrieron las categorías principales del aprendizaje automático:
  1. **ARIMA/SARIMA** (series temporales): modelo de pronóstico con ***rolling forecast*** y puntuación por residuos.
  2. **Regresión Logística** (supervisado lineal): línea base explicable con `class_weight` para desbalance.
  3. **XGBoost** (supervisado no lineal): *gradient boosting* con `scale_pos_weight`.
  4. **Isolation Forest** (no supervisado): detección de anomalías sin etiquetas de entrenamiento.
  5. **LSTM** (aprendizaje profundo): red neuronal recurrente implementada en PyTorch con ***learning rate scheduler*** y arquitectura de 128 unidades.

[1] https://github.com/Kans29/Tesis_Prediccion_LogFiles

- Evaluación comparativa de los cinco modelos bajo condiciones idénticas de datos, split temporal y métricas. Las decisiones de modelado adoptadas fueron documentadas con su justificación técnica.
- Selección del modelo LSTM como modelo final con base en su desempeño superior en la interfaz óptica de referencia frente al resto de modelos evaluados. Las métricas detalladas de esta comparación se presentan en el capítulo de evaluación (Sección 4.2.1).

1.5.5. Desarrollo

Esta fase se articuló con el **OE5** (desarrollar el prototipo funcional). Se implementaron los siguientes componentes:

- Implementación del *pipeline* automatizado de ocho fases en Python, estructurado en módulos reutilizables: (1) Parser, (2) Builder, (3) Events, (4) Dataset, (5) Analysis, (6) Models_Stat, (7) Models_ML, (8) Predict.
- Desarrollo del notebook de orquestación que permite la ejecución y reproducibilidad completa del experimento.
- Implementación del sistema de persistencia de modelos entrenados y exportación automatizada de gráficas y métricas.
- Desarrollo del prototipo CLI (`predecir.py`) que encapsuló el *pipeline* completo para su uso operativo: recibe un archivo de *syslog*, ejecuta el procesamiento completo y genera un reporte HTML autocontenido con la clasificación del riesgo por interfaz.

1.5.6. Despliegue

Esta fase completó la verificación del **OE5** y validó el cumplimiento integral del objetivo general. Se realizaron las siguientes actividades:

- Pruebas del prototipo en entorno controlado, validando el funcionamiento *end-to-end* del sistema con archivos de *syslog* reales proporcionados por EMCALI.
- Generación de reportes automatizados en formato HTML con la clasificación del nivel de riesgo por interfaz (Alto, Moderado, Bajo), incluyendo resumen ejecutivo, detalle por interfaz, distribución de probabilidades, línea de tiempo de alertas y metadatos del modelo.
- Verificación experimental del entrenamiento con múltiples interfaces simultáneamente frente al entrenamiento con una sola interfaz aislada, cuyos resultados se analizan en el capítulo de evaluación.

Esta metodología permitió desarrollar un prototipo funcional para pronosticar la posible caída de los servicios que prestan los equipos de la red del CORE-MPLS de EMCALI E.I.C.E. E.S.P., utilizando archivos de texto tipo LOGFILE provenientes de los equipos de la Red Multiservicios (RMS) de la empresa.

1.6. Resultados obtenidos

El proyecto alcanzó los siguientes resultados principales:

- **Pipeline de procesamiento:** Se implementó un *pipeline* automatizado que transformó el *syslog* crudo de la red CORE-MPLS en un conjunto de series temporales apto para modelado. Su arquitectura modular, los mecanismos de extracción y el volumen procesado se detallan en el Capítulo 3.
- **Evaluación comparativa de modelos:** Se evaluaron cinco modelos de clasificación (LSTM, ARIMA, Regresión Logística, XGBoost e Isolation Forest) bajo condiciones idénticas, seleccionándose la red neuronal recurrente LSTM (*Long Short-Term Memory*) implementada en PyTorch como modelo final por su desempeño superior.
- **Desempeño del modelo seleccionado:** Sobre la interfaz óptica crítica `COLO_S07706_0101 / XGigabitEthernet3/0/0`, el modelo LSTM alcanzó un desempeño predictivo operativamente relevante con un horizonte de predicción de una hora. Las métricas cuantitativas (Precisión, Recall, F1-Score con intervalo de confianza al 95 %) y el análisis desagregado por interfaz se presentan en el capítulo de Evaluación (Sección 4.2.1).
- **Prototipo funcional:** Se desarrolló el prototipo CLI `predecir.py`, una herramienta de línea de comandos que procesa un archivo de *syslog*, ejecuta el *pipeline* completo de predicción y genera un reporte HTML autocontenido con la clasificación del nivel de riesgo por interfaz (Alto, Moderado, Bajo).
- **Hallazgo sobre la variable de potencia óptica:** Se identificó que las interfaces con telemetría óptica densa fueron las únicas donde el modelo logró una separación efectiva de probabilidades y un rendimiento operativamente útil, lo que sugirió que la potencia óptica actúa como indicador de la calidad de la telemetría más que como predictor directo de fallas. Este hallazgo se analiza en detalle en el capítulo de evaluación.

Capítulo 2

Marco de referencia

Este capítulo presenta los fundamentos teóricos y los antecedentes que sustentan el desarrollo del prototipo de AIOps para la predicción de fallas en interfaces de red ópticas. Se describen los conceptos clave de detección de anomalías en telecomunicaciones, los modelos de series temporales y aprendizaje profundo empleados en el proyecto, y las métricas de evaluación adecuadas para escenarios de desbalance extremo de clases. Finalmente, se revisan los antecedentes académicos e industriales más relevantes.

2.1. Marco Teórico

2.1.1. AIOps: Inteligencia Artificial para las Operaciones de TI

El término **AIOps** (*Artificial Intelligence for IT Operations*) fue introducido por la firma analista Gartner (Gartner, 2017) para describir plataformas que combinan *big data* y aprendizaje automático con el objetivo de automatizar y mejorar las operaciones de TI. En el contexto de las telecomunicaciones, AIOps permite evolucionar desde un modelo de operación reactivo, donde las fallas se detectan después de ocurrir, hacia un modelo proactivo, donde los patrones de degradación se identifican con anticipación suficiente para ejecutar acciones correctivas.

2.1.2. Detección de anomalías en redes de telecomunicaciones

La detección de anomalías es una técnica de minería de datos orientada a identificar observaciones que se desvían significativamente del comportamiento esperado. En las redes de telecomunicaciones, los sistemas convencionales de monitoreo permiten detectar fallas consumadas en equipos o enlaces, pero rara vez disponen de capacidad para anticipar la degradación progresiva del servicio. Gómez et al. (2020) proponen un enfoque para la detección de anomalías en series de tiempo multivariadas aplicado a redes de telecomunicaciones, aprovechando técnicas de aprendizaje profundo. De manera complementaria, López-Avila et al. (2019) demuestran que la detección de anomalías basada en aprendizaje profundo mediante patrones establecidos permite analizar y evaluar cambios en el comportamiento de la red, superando las limitaciones de los enfoques estadísticos tradicionales basados en umbrales fijos.

2.1.3. Calidad de servicio (QoS) y su relación con la predicción de fallas

La Calidad de Servicio (*Quality of Service*, QoS) agrupa el conjunto de mecanismos que permiten garantizar niveles mínimos de rendimiento en las redes de telecomunicaciones, incluyendo la priorización de tráfico, el control de ancho de banda y la gestión de la congestión. Belzarena

and Aspirot (2010) analizan las arquitecturas de QoS (IntServ y DiffServ) en redes multiservicio, señalando que DiffServ ofrece una solución escalable para la clasificación y priorización del tráfico mediante ingeniería de tráfico.

En este contexto, la predicción anticipada de fallas en interfaces ópticas tiene un impacto directo sobre la QoS: al identificar una degradación inminente con una hora de anticipación, los operadores del NOC pueden ejecutar acciones de contingencia como el desvío de tráfico MPLS hacia rutas alternativas, el balanceo de carga entre interfaces activas o la programación de intervenciones preventivas en campo, evitando así la interrupción no planificada del servicio y el incumplimiento de los Acuerdos de Nivel de Servicio (SLA).

2.1.4. Inteligencia artificial aplicada a la detección de eventos en redes

La aplicación de técnicas de inteligencia artificial al análisis de registros de eventos (*log files*) en redes de telecomunicaciones permite identificar patrones, anomalías y tendencias que los sistemas de monitoreo convencionales basados en reglas estáticas no pueden capturar. Hussein and Répás (2024) demuestran que los métodos de aprendizaje automático supervisado permiten detectar anomalías en archivos de registro con alta efectividad cuando se dispone de datos etiquetados. Zhang et al. (2019) abordan el desafío de la inestabilidad en los datos de log, proponiendo un enfoque robusto de detección de anomalías que mantiene su efectividad ante cambios en la estructura y distribución de los registros. La combinación del análisis de logs con estas técnicas de IA ofrece la posibilidad de mejorar la eficiencia operativa, la confiabilidad y la capacidad predictiva de la gestión de redes.

2.1.5. Modelos de series temporales y aprendizaje automático

Una **serie temporal** es una secuencia de observaciones registradas en intervalos regulares de tiempo. En el dominio de las telecomunicaciones, las métricas de red (conteos de eventos, potencia óptica, tasas de error) constituyen series temporales cuya evolución contiene información predictiva sobre fallas futuras. Unos modelos relevantes para su análisis incluyen:

- **ARIMA/SARIMA.** Los modelos *AutoRegressive Integrated Moving Average* (ARIMA), formalizados por Box et al. (2015), son modelos estadísticos univariados que capturan la estructura de autocorrelación de una serie temporal mediante tres componentes: autorregresión (AR), diferenciación (I) y media móvil (MA). Su extensión estacional (SARIMA) incorpora componentes periódicos adicionales para modelar patrones cíclicos.
- **LSTM (Long Short-Term Memory).** Las redes **LSTM** son una arquitectura de red neuronal recurrente (RNN) propuesta por Hochreiter and Schmidhuber (1997), diseñada para resolver el problema del desvanecimiento del gradiente (*vanishing gradient*) que limita a las RNN convencionales. La arquitectura LSTM introduce tres compuertas (entrada, olvido y salida) que regulan el flujo de información a través de una celda de memoria interna, permitiendo al modelo aprender dependencias temporales de largo plazo. Esta capacidad resulta especialmente relevante para la predicción de fallas en redes, donde los patrones precursores de una caída pueden manifestarse minutos u horas antes del evento.

Además de los modelos de series temporales, existen algoritmos de aprendizaje automático que ofrecen enfoques complementarios para la clasificación y detección de anomalías:

- **Regresión Logística:** Modelo lineal supervisado que estima la probabilidad de pertenencia a una clase mediante la función sigmoide (Pedregosa et al., 2011). Constituye una línea base explicable para evaluar si la complejidad de modelos más sofisticados se justifica.
- **XGBoost** (*eXtreme Gradient Boosting*): Algoritmo de ensamble basado en árboles de decisión con *gradient boosting*, propuesto por Chen and Guestrin (2016). Ofrece manejo nativo del desbalance de clases mediante el parámetro `scale_pos_weight`.
- **Isolation Forest:** Algoritmo de detección de anomalías no supervisado propuesto por Liu et al. (2008), que identifica observaciones anómalas aislándolas mediante particiones aleatorias del espacio de características. A diferencia de los modelos supervisados, no requiere etiquetas de entrenamiento, lo que permite su aplicación en escenarios donde la información de fallas históricas es limitada.

2.1.6. Métricas de evaluación para clasificación binaria

En problemas de clasificación binaria, la **exactitud** (*accuracy*) mide la proporción de predicciones correctas sobre el total de observaciones: $Acc = \frac{TP+TN}{TP+TN+FP+FN}$. Sin embargo, como señalan He and Garcia (2009), en escenarios donde la clase positiva representa una fracción mínima del total (desbalance de clases), esta métrica resulta engañosa, dado que un clasificador trivial que siempre prediga la clase mayoritaria puede alcanzar valores artificialmente altos. Por esta razón, He and Garcia (2009) recomiendan las siguientes métricas orientadas a la clase minoritaria:

- **Precisión** (*Precision*): proporción de predicciones positivas que son correctas. $P = \frac{TP}{TP+FP}$.
- **Recall** (Sensibilidad): proporción de positivos reales que el modelo detectó. $R = \frac{TP}{TP+FN}$.
- **F1-Score**: media armónica de Precisión y Recall. $F_1 = 2 \cdot \frac{P \cdot R}{P+R}$.
- **PR-AUC** (*Precision-Recall Area Under Curve*): área bajo la curva de Precisión vs. Recall, preferida sobre ROC-AUC en escenarios de desbalance extremo porque no se ve inflada por la abundancia de verdaderos negativos.

2.1.7. Herramientas y plataformas utilizadas

El desarrollo del prototipo se apoyó en un ecosistema de herramientas de código abierto del lenguaje Python:

- **PyTorch** (Paszke et al., 2019): framework de aprendizaje profundo utilizado para la implementación y entrenamiento de la red LSTM.

- **Scikit-learn** (Pedregosa et al., 2011): librería de aprendizaje automático utilizada para los modelos de línea base (Regresión Logística, Isolation Forest), el escalado de características (`StandardScaler`) y las métricas de evaluación.
- **XGBoost** (Chen and Guestrin, 2016): librería nativa de *gradient boosting* utilizada para el modelo de ensamble basado en árboles.
- **statsmodels** (Seabold and Perktold, 2010): librería de modelado estadístico utilizada para la implementación del modelo ARIMA/SARIMA.
- **Pandas** y **NumPy**: librerías fundamentales para la manipulación vectorizada de datos y la construcción del *pipeline* de ingeniería de características.

2.2. Estado del Arte

2.2.1. Antecedentes académicos

La detección de anomalías en archivos de registro (*log files*) constituye un área activa de investigación en la intersección entre la ciencia de datos y la ingeniería de redes. Uno de los principales desafíos radica en la cantidad masiva de datos no estructurados generados por las redes modernas, donde las soluciones basadas en inteligencia artificial han demostrado resultados prometedores para identificar patrones anómalos y predecir comportamiento futuro.

Entre los trabajos más relevantes para el presente proyecto se encuentran:

- Hussein and Répás (2024) presentan un enfoque de aprendizaje automático para detectar anomalías en archivos de registro, evaluando múltiples técnicas supervisadas y demostrando su efectividad frente a los métodos basados en reglas.
- Ghuli (2018) analizan enfoques estadísticos robustos para detectar anomalías en archivos de registro, demostrando que representan una alternativa de menor complejidad computacional en comparación con modelos basados en aprendizaje automático para escenarios con distribuciones conocidas.
- Zhang et al. (2019) proponen un enfoque robusto para la detección de anomalías basada en logs que aborda el problema de la inestabilidad en los datos de registro, donde la estructura y distribución de los logs cambian con el tiempo, un desafío frecuente en entornos de producción.
- He et al. (2016) desarrollaron **Loglizer**, un *toolkit* de análisis de logs que integra diversas técnicas de *parsing* y detección automática de anomalías mediante aprendizaje automático, estableciendo un referente metodológico para la construcción de *pipelines* de procesamiento de logs.
- Du et al. (2017) proponen **DeepLog**, un modelo basado en redes LSTM que trata las secuencias de entradas de log como lenguaje natural para detectar anomalías y diagnosticar fallas en

sistemas distribuidos. A diferencia del presente proyecto, DeepLog opera sobre logs de aplicación (HDFS) en modo de detección en tiempo real, sin horizonte de predicción anticipada.

El presente proyecto se diferencia de estos antecedentes en tres aspectos fundamentales: (1) opera sobre *syslogs* de equipos de conmutación óptica de una red CORE-MPLS en producción, no sobre logs de aplicaciones, sistemas operativos o servidores web; (2) formula el problema como predicción temporal con horizonte de una hora, en lugar de detección post-hoc de anomalías ya consumadas; y (3) implementa un diseño anti-*leakage* con separación temporal estricta ($Gap \geq 60$ segundos) que garantiza la validez de las métricas reportadas, un aspecto no abordado explícitamente en los trabajos revisados.

El Cuadro 2.1 sintetiza las características técnicas de los antecedentes académicos revisados frente a la propuesta desarrollada en este proyecto.

Nota: el F1 reportado en la fila de este trabajo corresponde al desempeño sobre la interfaz óptica foco (`XGigabitEthernet3/0/0`), resultado titular del proyecto; el F1 global agregado sobre las 17 interfaces fue sensiblemente menor (Sección 4.2.3). Los valores de los demás trabajos corresponden a su F1 global reportado.

2.2.2. Antecedentes industriales

En la industria de telecomunicaciones, diversas plataformas comerciales abordan el análisis de archivos de registro desde diferentes perspectivas:

- **Splunk** (Splunk Inc., 2021): plataforma líder para el análisis y monitoreo de archivos de registro, con capacidades de búsqueda en tiempo real y visualización.
- **ELK Stack** (Elastic NV, 2021): suite de código abierto compuesta por Elasticsearch, Logstash y Kibana para la ingesta, indexación y visualización de logs.
- **LogRhythm** (LogRhythm Inc., 2021): solución comercial de gestión de información y eventos de seguridad (SIEM) con capacidades de análisis de comportamiento.
- **IBM QRadar** (IBM Corporation, 2021): plataforma de seguridad que utiliza algoritmos de aprendizaje automático para detectar patrones y anomalías en grandes volúmenes de datos.
- **Graylog** (Graylog, Inc., 2026): plataforma de gestión centralizada de logs y ciberseguridad (SIEM) orientada a la recolección y análisis de registros de servidores, aplicaciones y dispositivos de red.

A diferencia de estas soluciones comerciales de propósito general, el prototipo desarrollado en este proyecto se diseñó específicamente para los registros de eventos de la red CORE-MPLS de EMCALI E.I.C.E. E.S.P., permitiendo una personalización más efectiva para el tipo de datos y la topología específica de la infraestructura. Mientras que las plataformas comerciales se enfocan principalmente en la detección reactiva y la correlación de eventos, la solución propuesta se orienta hacia la predicción anticipada de fallas mediante modelos de aprendizaje profundo.

Cuadro 2.1: Comparativa de antecedentes académicos frente a la propuesta del presente proyecto.

Trabajo	**Dominio y tipo de datos**	**Modelos empleados**	**Métricas**	**Mejor F1**	**Tipo de análisis**
Hussein and Répás (2024)	Múltiples (revisión de literatura)	Revisión: Isolation Forest, SVM, XGBoost, LSTM, CNN, Autoencoders	Precision, Recall, F1	— (revisión)	Detección
Zhang et al. (2019)	Logs de sistemas distribuidos (HDFS, Microsoft)	Bi-LSTM con mecanismo de atención, FastText, TF-IDF	Precision, Recall, F1	0.997	Detección
He et al. (2016)	Logs de sistemas (HDFS, supercomputador BGL)	Regresión Logística, Árboles de Decisión, SVM, PCA, Minería de Invariantes	Precision, Recall, F1	0.998	Detección
Ghuli (2018)	Logs de servidor web Apache	Rango intercuartílico, Desviación absoluta de la mediana, Perceptrón multicapa	Error de predicción	—	Detección y predicción
Gómez et al. (2020)	Métricas de red de operador móvil (series multivariadas)	DC-VAE (convoluciones dilatadas + autoencoder variacional)	F1, Precision, Recall	—	Detección en tiempo real
Du et al. (2017)	Logs de sistemas distribuidos (HDFS)	LSTM sobre secuencias de eventos de log	Precision, Recall, F1	0.960	Detección en tiempo real
Este trabajo	**Syslogs de red CORE-MPLS óptica (EMCALI)**	**LSTM, ARIMA, Regresión Logística, XGBoost, Isolation Forest**	**Precision, Recall, F1, PR-AUC**	**0.615**	**Predicción (1 hora)**

2.3. Resumen del capítulo

Este capítulo estableció las bases conceptuales del proyecto. El paradigma **AIOps** enmarca la transición desde una operación reactiva hacia una proactiva sobre datos de red, y la **QoS** justifica el valor de anticipar caídas en interfaces ópticas. Sobre esta base se presentaron los cinco modelos considerados para la predicción y detección de eventos (ARIMA/SARIMA, LSTM, Regresión Logística, XGBoost e Isolation Forest), junto con las métricas apropiadas para escenarios de desbalance extremo de clases (Precisión, Recall, F1-Score y PR-AUC), donde la exactitud resulta engañosa. La revisión del estado del arte, tanto académica como industrial, evidencia que las soluciones existentes se concentran en la detección reactiva o en la anomalía genérica sobre logs, dejando abierto el nicho en el que se ubica este trabajo: la predicción anticipada de fallas específicas de interfaces ópticas a partir de *syslogs* de una red MPLS real.

Capítulo 3

Desarrollo del Proyecto

En este capítulo se expone el diseño, la especificación de requisitos y la implementación del prototipo orientado a la predicción anticipada de condiciones de indisponibilidad (*DOWN*) en interfaces de red ópticas, a partir de registros de eventos (*syslogs*). El núcleo de la propuesta es un *pipeline* automatizado de ocho fases (Figura 3.1) que transforma texto no estructurado en series temporales aptas para el modelado estadístico y de aprendizaje profundo. El capítulo se organiza en tres secciones: el diseño de la solución (descripción, requisitos y arquitectura), la implementación técnica de cada fase del *pipeline* y el resumen del capítulo.

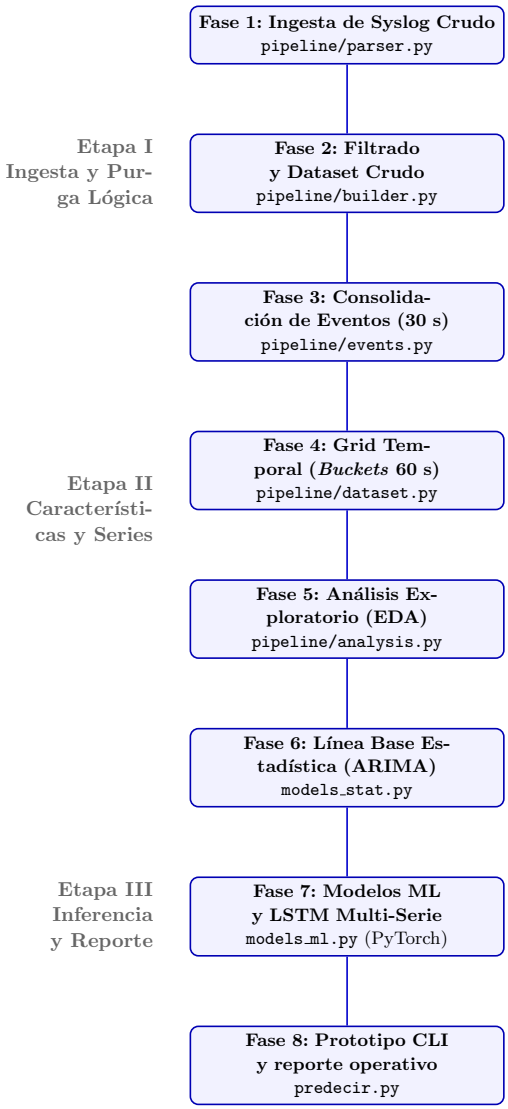


Figura 3.1: Flujo metodológico secuencial en 8 fases implementado en la arquitectura del software de AIOps para la predicción de fallas en interfaces de la red CORE-MPLS.

3.1. Diseño de la solución

3.1.1. Descripción de la propuesta

La propuesta de software consiste en el desarrollo de un componente modular y automatizado de Inteligencia Artificial para las Operaciones de TI (AIOps). Su propósito principal es mitigar la reactividad en el Centro de Operaciones de Red (NOC) mediante la generación de alertas tempranas con un horizonte de predicción (*Lookahead*) de una (1) hora. Este horizonte corresponde a la ventana mínima operativa que requiere el NOC para ejecutar acciones de contingencia (desvío de tráfico MPLS, balanceo de carga entre interfaces activas o despacho de cuadrillas de campo), validada con el equipo responsable del área. El sistema está optimizado para su despliegue sobre los equipos de conmutación y acceso de la red CORE-MPLS, operando específicamente sobre interfaces ópticas con telemetría densa, tales como la interfaz validada `COLO_S07706_0101 - XGigabitEthernet3/0/0`.

A diferencia de los enfoques tradicionales basados en umbrales estáticos que disparan alarmas cuando la falla ya ha ocurrido, esta solución analiza las firmas temporales y las degradaciones graduales en los *syslogs* (tales como alarmas de potencia óptica y conteo acumulado de advertencias). Los usuarios objetivo del sistema son los ingenieros de soporte de infraestructura y operadores del NOC, quienes disponen de una ventana de tiempo útil para desviar tráfico, balancear cargas de red o programar intervenciones en campo antes de la pérdida total del servicio.

La Figura 3.2 sitúa al equipo de soporte dentro del flujo operativo de la RMS de EMCALI y evidencia el punto donde el prototipo aporta valor. Los equipos de Core y Acceso (`COLO_S077` y `COLO_S127`) generan los registros que se consolidan como *Syslog Data* y se monitorean mediante Zabbix; el grupo de soporte accede a esta información de forma remota a través de una VPN. En el esquema actual, dicho acceso es fundamentalmente reactivo. El prototipo propuesto se integra sobre este mismo flujo de *syslogs* para entregar al equipo de soporte alertas anticipadas, transformando su rol de una respuesta correctiva a una intervención preventiva antes de que el servicio a los usuarios finales se vea afectado.

3.1.2. Requisitos

Para garantizar la viabilidad técnica y operativa de la solución dentro de una infraestructura de telecomunicaciones en producción, se definen los siguientes requisitos funcionales y no funcionales.

3.1.2.1. Requisitos Funcionales

- **RF-1 (Ingesta y Parseo Extensible):** El sistema debe ingerir archivos de registro masivos (*syslogs* crudos del orden de ∼192k líneas) y extraer pares clave=valor anidados en texto libre mediante expresiones regulares (Regex) precompiladas. En la Figura 3.4 se muestra un ejemplo de alarma capturada en los logfiles.
- **RF-2 (Filtrado de Topología Lógica):** El sistema debe ser capaz de identificar y descartar subinterfaces lógicas para evitar la redundancia de eventos e inflación artificial de métricas del puerto padre físico.

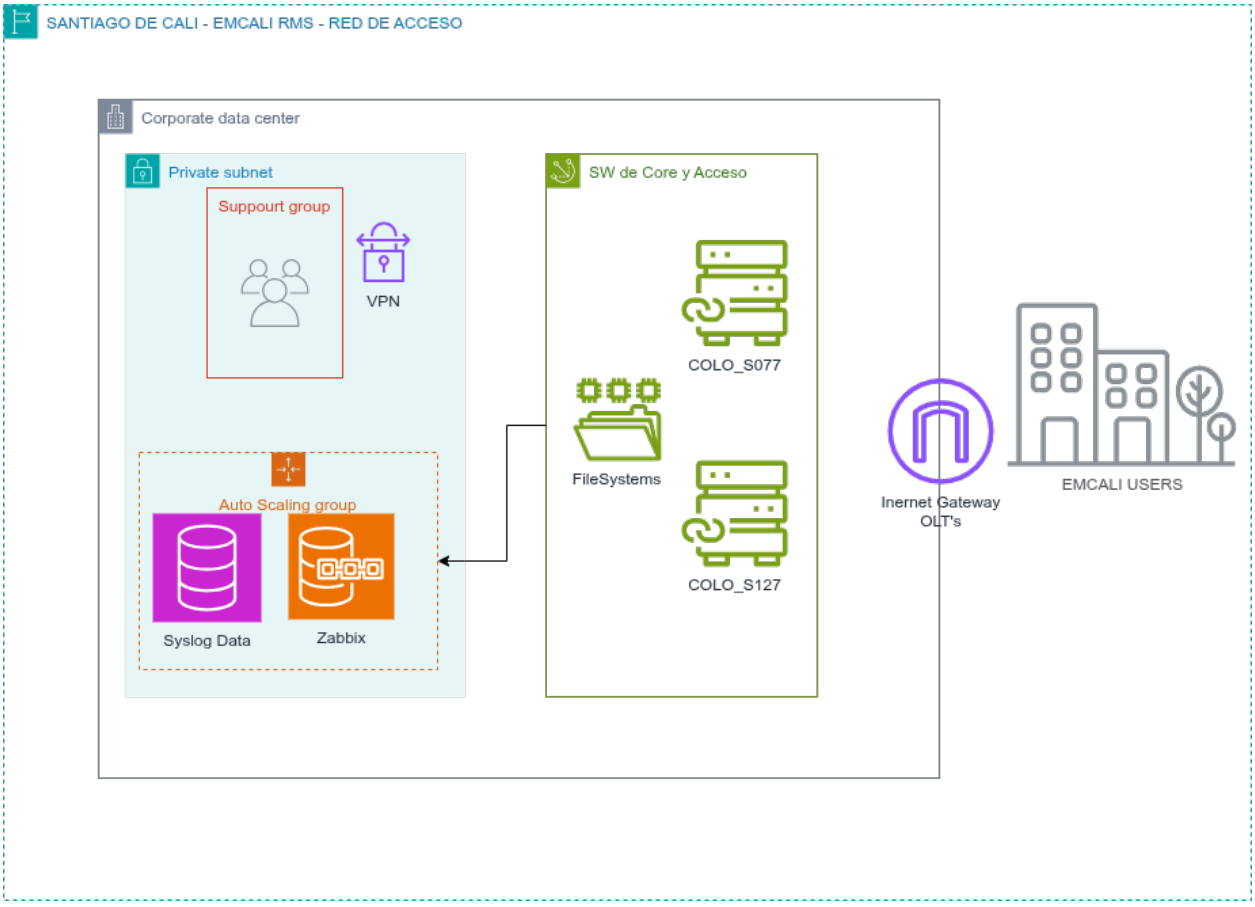


Figura 3.2: Arquitectura de acceso de la RMS de EMCALI y rol del equipo de soporte en el flujo operativo de *syslogs*.

- **RF-3 (Estructuración de Grid Temporal):** El sistema debe agrupar eventos discretos en ventanas uniformes (*Buckets*) de 60 segundos y computar variables rezagadas (*lags*) y estadísticas móviles (*rolling stats*).
- **RF-4 (Calibración de Alertas):** El sistema debe calcular la probabilidad de falla futura respetando una ventana de aislamiento ($Gap \geq 60$ s) para prevenir la filtración de datos (*data leakage*).
- **RF-5 (Inferencia Multi-Modelo):** El software debe proveer interfaces comunes para evaluar y comparar enfoques estadísticos clásicos como ARIMA/SARIMA (Box et al., 2015), y de aprendizaje automático como Regresión Logística (Pedregosa et al., 2011), Isolation Forest (Liu et al., 2008), XGBoost (Chen and Guestrin, 2016) y la red neuronal recurrente de memoria a largo plazo (LSTM), propuesta originalmente por (Hochreiter and Schmidhuber, 1997). El modelo fue desarrollado utilizando el framework de código abierto PyTorch (Paszke et al., 2019).

3.1.2.2. Requisitos No Funcionales

- **RNF-1 (Eficiencia en Memoria):** El procesamiento masivo de datos debe realizarse mediante operaciones vectorizadas (evitando bucles secuenciales iterativos línea por línea) utilizando las librerías *Pandas* y *Numpy*, limitando el análisis en caliente a ventanas de tiempo parametrizables (v.g., últimos 30 días).
- **RNF-2 (Persistencia y Reusabilidad):** Los modelos entrenados y los escaladores de datos deben ser serializables a disco (*checkpoints* en formato `.pkl` y `.pt`) para permitir la reanudación del *pipeline* o ejecuciones en caliente en la CLI sin re-entrenamiento.
- **RNF-3 (Independencia de Arquitectura de Inferencia):** El modelo final seleccionado (LSTM basado en *PyTorch*) debe ejecutarse en entornos de producción estándar sobre CPU, mitigando dependencias críticas de hardware especializado (GPU) o bloqueos de librerías dinámicas del sistema operativo host.

3.1.3. Arquitectura del pipeline

La arquitectura del sistema se organiza bajo el patrón de diseño de **Pipeline de Datos Orientado a Tuberías y Filtros** (*Pipes and Filters*). Este enfoque garantiza que la transformación desde el log crudo hasta la inferencia final ocurra en fases lógicas desacopladas y secuenciales.

El flujo macro del sistema se distribuye en tres grandes bloques funcionales que encapsulan las 8 fases operativas del proyecto:

1. **Subsistema de Ingesta y Purga (Fases 1 a 3):** Extrae las variables estructuradas desde el syslog crudo (`parser.py`), remueve el ruido topológico (`builder.py`) y consolida los logs en eventos lógicos mediante agrupamiento temporal de 30 segundos (`events.py`).

2. **Subsistema de Ingeniería de Características y Análisis (Fases 4 y 5):** Genera el *grid* de series de tiempo indexado por el par *(host, interfaz)* y realiza el análisis exploratorio de estacionariedad y autocorrelación (`dataset.py` y `analysis.py`).

3. **Subsistema de Modelado y Predicción (Fases 6 a 8):** Ejecuta los motores de inferencia estadística univariada (`models_stat.py`) y de aprendizaje profundo multi-serie (`models_ml.py`), y empaqueta la solución en un prototipo CLI ejecutable (`predecir.py`).

3.1.3.1. Diseño anti-leakage del etiquetado

El diseño del dataset de series de tiempo impone una estricta separación temporal para evitar el sesgo predictivo. El etiquetado de la variable objetivo ($Label_i$) para un *bucket* i con tiempo de finalización T_{end} se rige bajo la siguiente expresión:

$$Label_i = \begin{cases} 1 & \text{si ocurre al menos un evento } DOWN \in (T_{end} + \Delta t_{gap}, T_{end} + \Delta t_{gap} + \Delta t_{lookahead}] \\ 0 & \text{en caso contrario} \end{cases} \tag{3.1}$$

Donde $\Delta t_{gap} = 60$ segundos y $\Delta t_{lookahead} = 3600$ segundos (1 hora). Al asegurar que $\Delta t_{gap} \geq$ tamaño del bucket, se elimina cualquier posibilidad de que una alerta generada sea síncrona con la falla o sufra de contaminación del futuro.

3.2. Implementación

El software se implementó como un paquete modular en el lenguaje *Python*, estructurado en una librería reutilizable denominada `pipeline/` y un orquestador de producción implementado en un *Jupyter Notebook* ejecutable por etapas. A continuación se detalla la implementación de cada una de las ocho fases del *pipeline*.

3.2.1. Fase 1 — Ingesta del syslog crudo (`builder.py`, `parser.py`)

La ingesta del *syslog* crudo se implementa en el módulo `builder.py` (función `build_raw`), que recorre el archivo línea por línea (ver Figura 3.3 para la estructura del protocolo y Figura 3.4 para un fragmento real) y combina dos familias de mecanismos de extracción. Sobre el **encabezado** de cada línea aplica un conjunto de expresiones regulares (*Regex*) precompiladas que aíslan la marca temporal (*syslog* y dispositivo), el equipo (host), el trío módulo/severidad/evento, la interfaz y los valores de potencia óptica, además de los indicadores de estado (caída, recuperación y advertencia). Sobre el **contenido entre paréntesis** delega en las funciones auxiliares del módulo `parser.py`: un escáner de pila (`extraer_bloques_parentesis`) que, sin usar *Regex*, aísla los bloques, incluso anidados, y una serie de sub-parsers que separan los pares `clave=valor` y descomponen los campos compuestos `Information` y `ReasonDescr`. El Cuadro 3.1 detalla todos los mecanismos de extracción junto con el patrón que aplica cada uno.

Cuadro 3.1: Mecanismos de extracción y expresiones regulares (*Regex*) aplicados sobre cada línea del *syslog*: en `builder.py` (encabezado de la línea) y en `parser.py` (contenido entre paréntesis).

Campo o mecanismo	**Patrón (*Regex*) o técnica**
(a) Encabezado de la línea — `builder.py`	
`host`	`(COLO_\S+)`
`datetime_syslog`	`([A-Za-z]{3}\s+\d{1,2}\s+` `\d{2}:\d{2}:\d{2})`
`datetime_device`	`(\d{4}-\d{1,2}-\d{1,2}` `\s+\d{2}:\d{2}:\d{2})`
`module` / `severity` / `event`	`([A-Z0-9]+)/(\d+)/([A-Z0-9_]+)` (grupos 1, 2 y 3, respectivamente)

`interfaz` (respaldo por *Regex*)	`(GigabitEthernet\d+/\d+/\d+|`
	`XGigabitEthernet\d+/\d+/\d+|`
	`100GE\d+/\d+/\d+|40GE\d+)`

`rx_power_dbm`	`(?:RX power\(dBM\):\s*(-?\d+\.\d+))` `|(?:Now power is (-?\d+\.\d+))`
`tx_power_dbm`	`TX power\(dBM\):\s*(-?\d+\.\d+)`
`is_down`	`(turned into DOWN state|` `has entered the DOWN state)`

`is_recovered`	`(Optical module recovered from power abnormal|turned into`
	`UP state|`
	`has entered the UP state)`

`is_warning`	`(warning threshold|lower warning threshold|upper warning`
	`threshold|`
	`Optical module power is abnormal)`

`alert_msg`	`ReasonDescr="([^"]*?warning threshold[^"]*?\.)` (con respaldo a un patrón general del mensaje)
`interfaz` (Filtro 3: subinterfaz)	`\.\d+$` (descarta subinterfaces lógicas)
(b) Contenido entre paréntesis — `parser.py`	
`extraer_bloques_parentesis`	Escáner de pila carácter a carácter (**sin** *Regex*); aísla los bloques `(...)`, incluso anidados.
Búsqueda por clave (`_INTERFAZ_KEYS`)	Sin *Regex*: toma la interfaz del primer campo disponible entre `Interface`, `InterfaceName`, `IfName`, `PhysicalName`, etc.
`parsear_key_values`	`,\s*(?=\w+=)`
`parsear_information`	`,\s*(?=[^,]+:)` y `(\d+M):\s*(\w+)`

`parsear_reason`	`Now power is ([\-\d\.]+)`
	`Set lower threshold is ([\-\d\.]+)`
	`Default lower threshold is ([\-\d\.]+)`
	`interface description is as follows:\s*(.+)`

`extraer_interfaz_desde_texto`	`Interface\s+([A-Za-z0-9\-\.\/]+)` `on the interface\s+([A-Za-z0-9\-\.\/]+)`

Figura 3.3: Niveles de severidad del protocolo Syslog (RFC 5424) (Gerhards, 2009).

Nota. Adaptado de *The Syslog Protocol* (RFC 5424, p. 11), por R. Gerhards, 2009, IETF.

A modo de ejemplo, la Figura 3.4 presenta un fragmento real del *syslog* procesado que ilustra, en orden cronológico, los tipos de evento de interés para el proyecto: desde la alarma de potencia óptica por debajo del umbral de advertencia hasta la caída del enlace y su *trap* SNMP equivalente. El detalle de cada uno de los cinco registros representativos se describe en el pie de la figura.

```
Apr  5 12:40:59 COLO_S07706_0101 ENTITYTRAP/3/OPTMAYINVALID:OID 1.3.6.1.4.1.2011.5.25.219.2.4.5 The optical
    ↪ power exceeds the upper warning threshold or falls below the lower warning threshold. (Index
    ↪ =68272910, EntityPhysicalIndex=68272910, PhysicalName="XGigabitEthernet4/0/12", EntityTrapFaultID
    ↪ =136223, EntityTrapReasonDescr="The receive power is below the lower warning threshold. The interface
    ↪  description is as follows: COLO_8905_0101_LAB_SAFE.")
Apr  6 12:43:26 COLO_S07706_0101 IFNET/1/IF_LINKUP:OID 1.3.6.1.6.3.1.1.5.4 Interface 46 turned into UP state
    ↪ .(AdminStatus=1,OperStatus=1,InterfaceName=XGigabitEthernet3/0/0)
Apr  6 08:19:27 10.29.205.13/10.29.205.13 2026-4-6 13:19:27 COLO_S07706_0101 %%01IFPDT/4/IF_STATE(l)
    ↪ [50110280]:Interface XGigabitEthernet3/0/0 has turned into DOWN state.
Apr  6 08:19:27 10.29.205.13/10.29.205.13 2026-4-6 13:19:27 COLO_S07706_0101 %%01IFADP/4/PORTDOWNINFO(l)
    ↪ [50110281]:Slot=3;Interface XGigabitEthernet3/0/0 has turned into DOWN state. (Information=Physical
    ↪ state(PMD): down, Physical state(PCS): down, RX loss of signal: yes, Current lane0 RX power(dBM):
    ↪ -30.02, Current lane0 TX power(dBM): -2.59, Local fault: yes, Remote fault: no, PMA fault: NA, PCS
    ↪ fault: NA, High BER: NA, PCS block lock: NA, Transmit local fault: NA, Receive local fault: NA, PMD
    ↪ receive signal OK: NA, XGXS PLL has lost lock: NA, XGXS PLL lock detected: NA, RX lock: no)
Apr  6 13:19:27 COLO_S07706_0101 IFNET/1/IF_LINKDOWN:OID 1.3.6.1.6.3.1.1.5.3 Interface 46 turned into DOWN
    ↪ state.(AdminStatus=1,OperStatus=2,InterfaceName=XGigabitEthernet3/0/0)
```

Figura 3.4: Fragmento real del archivo de *syslog* de la red CORE-MPLS de EMCALI. Se muestran cinco registros representativos: alarma de potencia óptica bajo el umbral de advertencia (`OPTMAYINVALID`), la interfaz foco en estado *UP* (`IF_LINKUP`) y su posterior caída a estado *DOWN* (`IF_STATE`), el detalle físico del puerto con la potencia óptica medida en el momento de la caída (`PORTDOWNINFO`, `RX power(dBM): -30.02` y `TX power(dBM): -2.59`) y el *trap* SNMP de enlace caído (`IF_LINKDOWN`).

El Cuadro 3.2 resume las características del archivo crudo de entrada y del dataset resultante tras el procesamiento completo del *pipeline*.

Cuadro 3.2: Características del archivo de *syslog* crudo y del dataset procesado.

Característica	**Crudo**	**Procesado**
Líneas / *Buckets*	∼192,359	67,187
Equipos (hosts)	2	2
Interfaces	184	17
Eventos consolidados	—	991
Positivos ($label = 1$)	—	2,311 (3.44 %)

El archivo de *syslog* analizado abarca diez fechas calendario, del 5 al 14 de abril de 2026, correspondientes a la ventana de captura habilitada para el proyecto. Este período recoge tanto la operación nominal de la red como episodios de degradación y caída de interfaces ópticas, lo que aporta la variabilidad necesaria para el aprendizaje supervisado del fenómeno objetivo.

3.2.2. Fases 2 y 3 — Filtrado y consolidación de eventos (`builder.py`, `events.py`)

El Cuadro 3.3 formaliza, a modo de diccionario de datos, las 14 columnas que `builder.py` produce por cada registro de *syslog* (`_COLUMNAS`, volcadas en `dataset_raw.csv`) y que constituyen la materia prima para las fases posteriores de filtrado, consolidación e ingeniería de características.

Cuadro 3.3: Diccionario de datos de las 14 columnas generadas por `builder.py` para cada registro del *syslog* crudo (`dataset_raw.csv`).

Campo	Descripción	Tipo	Ejemplo
`datetime_syslog`	Marca temporal del encabezado *syslog* (sin año; este se infiere posteriormente).	Fecha/hora	`Apr 6 12:43:26`
`datetime_device`	Marca temporal generada por el dispositivo (incluye año).	Fecha/hora	`2026-4-6 13:19:27`
`host`	Identificador del equipo emisor.	Texto	`COLO_S07706_0101`
`module`	Módulo funcional Huawei que emite el mensaje.	Texto	`IFPDT`, `ENTITYTRAP`
`severity`	Nivel de severidad Huawei del mensaje (0–7).	Entero	`4`
`event`	Tipo de evento (*event_type*) que clasifica la incidencia.	Categórica	`IF_STATE`, `OPTMAYINVALID`
`interfaz`	Interfaz física asociada al registro.	Texto	`XGigabitEthernet3/0/0`
`rx_power_dbm`	Potencia óptica de recepción medida.	Flotante (dBm)	`-30.02`
`tx_power_dbm`	Potencia óptica de transmisión medida.	Flotante (dBm)	`-2.59`
`is_down`	Indicador de caída de la interfaz (mensaje de estado *DOWN*).	Binaria	`True`
`is_recovered`	Indicador de recuperación (estado *UP* o recuperación óptica).	Binaria	`True`
`is_warning`	Indicador de alarma de umbral de advertencia óptica.	Binaria	`True`
`alert_msg`	Texto de la alarma relevante (de `ReasonDescr` o del mensaje general).	Texto	`The receive power is below the lower warning threshold.`
`raw_log`	Línea original completa del *syslog*, conservada para trazabilidad.	Texto	(línea cruda)

El módulo `builder.py` realiza tres filtros restrictivos sobre el volumen crudo de datos utilizando máscaras booleanas vectorizadas de Pandas:

- **Filtro 1:** Exclusión analítica de registros sin interfaz identificable (25.836 registros, 13.4 %

del volumen crudo). Antes de descartarlos, el *pipeline* verificó que ninguno de ellos portaba telemetría óptica útil (potencia RX/TX, alarmas o estados de falla), garantizando que no se perdiera señal relevante junto con el ruido.

- **Filtro 2:** Purga de logs sin variables útiles (sin presencia de potencia RX/TX, alarmas específicas o estados explícitos de falla) (164.728 registros, 85.6 %).
- **Filtro 3 (Filtro Crítico Topológico):** Eliminación de subinterfaces lógicas mediante la expresión regular `\.\d+$`, evitando la contaminación por duplicación de eventos en canales lógicos multiplexados (517 registros, 0.3 %).

La aplicación conjunta de estos tres filtros redujo el volumen de 192.359 registros crudos a 1.278 con relevancia óptica (un 99.3 % de descarte) y, junto con la posterior consolidación de eventos, el espacio de análisis pasó de las 184 interfaces presentes en el *syslog* crudo a las 17 interfaces físicas reales del dataset final de modelado.

El módulo `events.py` recibe el dataset filtrado y consolida los registros en eventos lógicos: asigna un mismo `event_id` a los registros *syslog* sucesivos de un mismo par *(host, interfaz)* mientras el intervalo respecto al registro anterior no supere los 30 segundos (umbral de *gap*); un nuevo evento comienza cuando ese intervalo se excede. Se trata, por tanto, de una agrupación por proximidad temporal (tipo sesión) y no de un bloque de duración fija: una ráfaga encadenada de líneas asociadas a una misma incidencia se contabiliza como un solo evento, aunque en conjunto abarque más de 30 segundos, lo que reduce la redundancia inherente al *syslog*.

3.2.3. Fase 4 — Ingeniería de características (`dataset.py`)

Para dotar a los modelos de la capacidad de detectar patrones de escalada gradual de fallas, el módulo `dataset.py` genera un *grid* de series de tiempo con *buckets* de 60 segundos e inyecta un subconjunto optimizado de 14 características (*FEATURES_SUBSET*), derivadas de las variables estructurales del *syslog* extraídas por el *parser* (Figura 3.4). La elección de 60 s como tamaño de *bucket* responde a la resolución temporal del *syslog* (marcas de tiempo con precisión de segundo) y a la restricción anti-*leakage* $\Delta t_{gap} \geq$ tamaño del *bucket* descrita en la Sección 3.1.3.1, que fija en un minuto la separación mínima entre la observación y el horizonte de predicción. La unidad de análisis es el **evento consolidado** definido en la Fase 3. El Cuadro 3.4 presenta el diccionario de datos de las 14 características, indicando su categoría, tipo de dato y regla de derivación a partir de dichos eventos.

Cuadro 3.4: Diccionario de datos de las 14 características de entrenamiento (*FEATURES_SUBSET*) generadas por `dataset.py` sobre cada *bucket* de 60 s.

Variable	**Tipo**	**Categoría**	**Descripción y derivación**
`event_count`	Entero	Conteo directo	Número de eventos consolidados cuyo instante de inicio cae en el *bucket*. Cada evento se cuenta una sola vez, en el *bucket* donde arranca.
`warning_count`	Entero	Conteo directo	Número de registros *syslog* de severidad *warning* contenidos en los eventos cuyo *onset* cae en el *bucket* (no cuenta eventos, sino líneas). Suma de `is_warning_count`.
`down_count`	Entero	Conteo directo	Número de registros *syslog* de caída (*DOWN*) de interfaz contenidos en los eventos cuyo *onset* cae en el *bucket*. Suma de `is_down_count`.
`event_count_log1p`	Flotante	Transformación	Compresión logarítmica $\ln(1 +$ `event_count`$)$ para atenuar el impacto de *outliers*.
`event_count_accel`	Flotante	Transformación	Segunda derivada de `event_count_roll_mean` (`diff().diff()`); capta la aceleración de la tendencia reciente de eventos.
`rx_power_dbm`	Flotante (dBm)	Telemetría óptica	Potencia óptica de recepción media del *bucket*, con relleno hacia adelante (*forward fill*). Densidad global $\sim$58 % tras el relleno.
`tx_power_dbm`	Flotante (dBm)	Telemetría óptica	Potencia óptica de transmisión media del *bucket*, con relleno hacia adelante.
`power_missing`	Binaria (0/1)	Indicador	Vale 1 si `rx_power_dbm` era nulo antes del *forward fill*; señala la ausencia/antigüedad de la medición óptica (*staleness*).

Continúa en la página siguiente

(Cuadro 3.4 — continuación)

Variable	Tipo	Categoría	Descripción y derivación
`event_count_roll_mean`	Flotante	Estadístico móvil	Media de `event_count` en la ventana móvil (`min_periods=1`); tendencia reciente.
`event_count_roll_std`	Flotante	Estadístico móvil	Desviación estándar de `event_count` en la ventana móvil; dispersión reciente.
`warning_count_roll_mean`	Flotante	Estadístico móvil	Media de `warning_count` en la ventana móvil.
`warning_count_roll_std`	Flotante	Estadístico móvil	Desviación estándar de `warning_count` en la ventana móvil.
`anomaly_score`	Flotante	Derivada combinada	$\max\big(0,\ z(\texttt{event\_count}) - z(\texttt{rx\_power\_dbm})\big)$ con estandarización expansiva (sólo datos pasados, sin *look-ahead*).
`power_trend`	Flotante	Derivada combinada	Pendiente de una regresión lineal (`polyfit` grado 1) de `rx_power_dbm` sobre la ventana móvil; resume la tendencia de la potencia óptica en la ventana.

Las interfaces ópticas de la red corresponden a transceptores SFP/QSFP con Monitoreo de Diagnóstico Digital (*Digital Diagnostic Monitoring*, DDM) habilitado, mecanismo que expone la telemetría de potencia óptica empleada por las variables `rx_power_dbm` y `tx_power_dbm`. La Figura 3.5 muestra la información del transceptor de la interfaz foco `XGigabitEthernet3/0/0`.

La variable `rx_power_dbm` corresponde a un valor numérico continuo (punto flotante, con dos decimales de precisión) que expresa la potencia óptica de recepción del transceptor en decibelios-milivatio (dBm), una escala logarítmica referida a 1 mW ($P_{\mathrm{dBm}} = 10\,\log_{10}(P_{\mathrm{mW}}/1\,\mathrm{mW})$). Por tratarse de niveles de potencia inferiores a 1 mW, sus valores son negativos: en el conjunto analizado oscilaron entre $\sim$−5 dBm (recepción saludable) y −40 dBm (piso de medición del transceptor, reportado ante la pérdida total de señal, *RX loss of signal*), con una media en torno a −17,9 dBm. Valores progresivamente más negativos indican una señal óptica más débil; cuando la potencia cae por debajo del umbral de advertencia inferior ($\sim$−22 dBm en la red analizada), el *switch* emite la alarma `OPTMAYINVALID` (Figura 3.4), lo que la convierte en un indicador temprano de degradación del enlace, cabe anotar que los valores máximos y mínimos de operación los aplica el transceptor óptico.

No obstante, aunque esta variable se encuentra dentro del alcance del proyecto, su baja cobertura de telemetría (solo cerca del $\sim$0.04 % de los *buckets* contiene una lectura numérica instantánea y, aun tras el relleno hacia adelante, cerca del 42 % permanece nulo) impidió emplearla como señal primaria de modelado. Esta baja cobertura se explica porque, en su mayoría, los registros de

```
<COLO_S07706_0101>display transceiver interface XGigabitEthernet 3/0/0

XGigabitEthernet3/0/0 transceiver information:
---------------------------------------------------------------
Common information:
  Transceiver Type               :10GBASE_LR_SFP
  Connector Type                 :LC
  Wavelength(nm)                 :1310
  Transfer Distance(m)           :10000(9um)
  Digital Diagnostic Monitoring  :YES
  Vendor Name                    :INNOLIGHT
  Vendor Part Number             :TR-PX13L-N00
  Ordering Name                  :
---------------------------------------------------------------
Manufacture information:
  Manu. Serial Number            :INEA00040109
  Manufacturing Date             :2014-04-11
  Vendor Name                    :INNOLIGHT
---------------------------------------------------------------
Alarm information:
  RX loss of signal
  RX power low
---------------------------------------------------------------
Diagnostic information:
  Temperature(                       :31
  Voltage(V)                             :3.30
  Bias Current(mA)                       :24.30
  Bias High Threshold(mA)                :90.00
  Bias Low  Threshold(mA)                :2.00
  Current Rx Power(dBM)                  :-40.00
  Default Rx Power High Warning(dBM)     :0.00
  Default Rx Power Low  Warning(dBM)     :-19.00
  Default Rx Power High Threshold(dBM)   :1.00
  Default Rx Power Low  Threshold(dBM)   :-20.00
  Current Tx Power(dBM)                  :-2.57
  Default Tx Power High Warning(dBM)     :1.00
  Default Tx Power Low  Warning(dBM)     :-6.99
  Default Tx Power High Threshold(dBM)   :2.01
  Default Tx Power Low  Threshold(dBM)   :-7.96
  User Set Rx Power High Threshold(dBM) :1.00
  User Set Rx Power Low Threshold(dBM)  :-20.00
  User Set Tx Power High Threshold(dBM) :2.01
  User Set Tx Power Low Threshold(dBM)  :-7.96
  Transceiver phony alarm                :Yes
---------------------------------------------------------------
<COLO_S07706_0101>
```

Figura 3.5: Información del transceptor óptico SFP de la interfaz foco `XGigabitEthernet3/0/0` (módulo 10 Gb). El campo *Digital Diagnostic Monitoring* habilitado (`YES`) es el que permite obtener la telemetría de potencia óptica y sus valores Threshold.

syslog informan la condición de potencia óptica de forma **cualitativa**, señalando que la recepción se encuentra por debajo del umbral de advertencia (por ejemplo, la alarma `OPTMAYINVALID`), sin adjuntar un valor numérico; la lectura numérica de `rx_power_dbm` solo aparece en eventos específicos (como `PORTDOWNINFO`). Por esta razón, el enfoque se construyó sobre la señal basada en **eventos** de potencia óptica, en particular `event_count` y sus variables derivadas, reservando `rx_power_dbm` como un atributo complementario y como indicador de la calidad de la telemetría, validado sobre la interfaz foco donde su densidad alcanzó el 99,6 %.

3.2.3.1. Manejo de valores faltantes

Dado que el *syslog* es una fuente dispersa e irregular, el tratamiento de los valores faltantes se abordó de forma diferenciada según la naturaleza de cada variable y bajo una restricción transversal: no introducir información futura (*look-ahead*) en ningún paso. El Cuadro 3.5 resume la estrategia aplicada en cada etapa del *pipeline*.

Cuadro 3.5: Estrategia de manejo de valores faltantes por etapa del *pipeline* y grupo de variables.

Etapa	Variables	Estrategia	Justificación
Ingesta (`builder.py`)	Filas sin interfaz o sin ninguna señal útil	Eliminación (Filtros 1 y 2)	Descarta registros no informativos antes de construir la serie.
Ingeniería (`dataset.py`)	Conteos: `event_count`, `warning_count`, `down_count`	Relleno con cero (`fillna(0)`)	La ausencia de evento en el *bucket* es un cero real, no un dato perdido.
	Potencia óptica: `rx_power_dbm`, `tx_power_dbm`	Propagación hacia adelante (`ffill`); sin *backfill*	Reutiliza la última lectura válida sin introducir valores futuros.
	`power_missing`, `seconds_since_last_power`	Codificación de la ausencia como variable	Modela de forma explícita la falta y la antigüedad del dato óptico.
	`event_count_roll_std`, `anomaly_score`, `power_trend`	`fillna(0)` y desviación `fillna(1)`	Evita indefiniciones (un solo punto) y divisiones por cero.
Modelado (`models_ml.py`)	NaN residual al inicio de cada serie	Imputación a cero (`nan_to_num`) tras el escalado	Cero ≈ media tras `StandardScaler` (ajustado solo en *train*); imputación neutra y sin fuga.

Tres principios rigen este tratamiento. En primer lugar, la ausencia de un evento en un *bucket* de 60 s constituye información legítima y no un dato perdido, por lo que los conteos se completan con cero y no con imputaciones estadísticas. En segundo lugar, la telemetría de potencia óptica, cuya lectura numérica está ausente en el 99,96 % de los *buckets*, se propaga únicamente hacia adelante (`ffill`) y nunca hacia atrás, para no filtrar valores futuros; además, la propia condición de ausencia se preserva como señal mediante las variables `power_missing` (indicador binario) y

`seconds_since_last_power` (antigüedad de la última medición), de modo que el modelo distingue una medición reciente de una obsoleta. En tercer lugar, los valores que permanecen nulos al inicio de cada serie (un 41,9 % para `rx_power_dbm`, al no existir una lectura previa que propagar) se imputan con cero *después* de haber ajustado el `StandardScaler` exclusivamente sobre la partición de entrenamiento; puesto que el escalado centra cada variable en su media, el cero equivale al valor medio y constituye una imputación neutra que, al aplicarse tras la partición temporal, no contamina el conjunto de prueba.

3.2.4. Fase 5 — Análisis exploratorio (`analysis.py`)

El módulo `analysis.py` ejecuta el análisis exploratorio sobre el dataset de series de tiempo generado en la Fase 4, con el objetivo de caracterizar la distribución de las variables y validar la existencia de patrones temporales explotables que orienten la configuración de los modelos. La interfaz `XGigabitEthernet3/0/0` del equipo `COLO_S07706_0101` fue seleccionada como serie foco por poseer la mayor densidad de telemetría óptica (`rx_power_dbm`: 99,6 % no nulo) y la mayor concentración de eventos *DOWN* en el conjunto de entrenamiento.

3.2.4.1. Estadística descriptiva y distribución de frecuencias

Como paso previo al análisis temporal se caracterizó el dataset mediante la estadística descriptiva de las variables numéricas y el conteo de frecuencias de las variables categóricas del *syslog* crudo. El Cuadro 3.6 resume el mínimo, la media, la desviación estándar y el máximo de las 14 características. Las variables de conteo (`event_count`, `warning_count`, `down_count`) resultan altamente dispersas, con media cercana a cero y la mayoría de los *buckets* en valor nulo, lo que refleja el carácter esporádico de los eventos y anticipa el fuerte desbalance de clases del problema. La potencia óptica `rx_power_dbm` varía entre −40 dBm (piso de medición del transceptor, asociado a pérdida de señal y registrado durante las caídas de puerto) y −5,38 dBm, con una media de −17,9 dBm. Destaca además el máximo de `anomaly_score` (94,2 frente a una media de 0,56), indicio de una cola derecha pesada que se retoma en el análisis de valores atípicos.

En cuanto a las variables categóricas, la Figura 3.6 muestra que el 94,1 % de los registros corresponde a la severidad 4 (*Warning*) y que no aparecen los niveles 0 (*Emergency*) ni 7 (*Debug*). La Figura 3.7 evidencia que el campo `event` está dominado por alarmas de capa 2, `MACEXCDALARM` (54,1 %) y `MACHASHCONFLICTALARM` (30,4 %), repartidas además entre 117 tipos distintos. Este hallazgo es determinante: el campo `event` **no permite por sí solo aislar los incidentes de potencia óptica**, pues estos no poseen un código de evento único ni consistente. Por ello la relevancia óptica se determina a partir del **contenido del mensaje** mediante las expresiones regulares descritas en el Cuadro 3.1 (marcadores `is_warning`, `is_down`, `is_recovered` y presencia de valores RX/TX), lo que justifica la batería de filtros aplicada en las Fases 2 y 3.

La Figura 3.8 cuantifica estos marcadores sobre el *syslog* crudo: en conjunto, apenas el 0,93 % de los registros (1 795 de 192 359) posee relevancia óptica, con predominio de las advertencias de umbral (`is_warning`, 0,44 %) sobre las transiciones a estado *DOWN* (`is_down`, 0,26 %) y las recuperaciones (`is_recovered`, 0,23 %). De especial importancia es que solo el 0,02 % de los registros

Cuadro 3.6: Estadística descriptiva de las 14 características sobre el dataset completo ($n = 67\,187$ *buckets* de 60 s). Las variables `rx_power_dbm` y `tx_power_dbm` se resumen sobre los *buckets* con lectura óptica disponible.

Variable	Mín.	Media	Desv.	Máx.
`event_count`	0	0.014	0.117	1
`warning_count`	0	0.013	0.111	1
`down_count`	0	0.002	0.097	9
`event_count_log1p`	0	0.010	0.081	0.693
`event_count_accel`	−0,333	0.000	0.024	0.667
`rx_power_dbm`	−40,00	−17,88	13.09	−5,38
`tx_power_dbm`	−6,38	−3,70	1.46	−2,58
`power_missing`	0	1.00	0.02	1
`event_count_roll_mean`	0	0.014	0.040	1.000
`event_count_roll_std`	0	0.044	0.111	0.707
`warning_count_roll_mean`	0	0.013	0.034	1.000
`warning_count_roll_std`	0	0.040	0.105	0.707
`anomaly_score`	0	0.561	1.542	94.218
`power_trend`	−5,218	−0,001	0.047	0.727

reporta un **valor numérico** de RX; la telemetría de potencia se informa mayoritariamente de forma cualitativa (cruces de umbral) y no como lectura instantánea, lo que motivó centrar la ingeniería de características en el conteo de eventos en lugar de en la magnitud de RX. Esta escasez estructural de la señal de interés se traslada al objetivo de aprendizaje: la Figura 3.9 muestra que solo el 3,44 % de los *buckets* (2 311 de 67 187) precede a un evento *DOWN* dentro del horizonte de predicción, configurando un problema de clasificación **fuertemente desbalanceado** que condiciona la elección de métricas y la ponderación de clases discutidas en el capítulo de Evaluación.

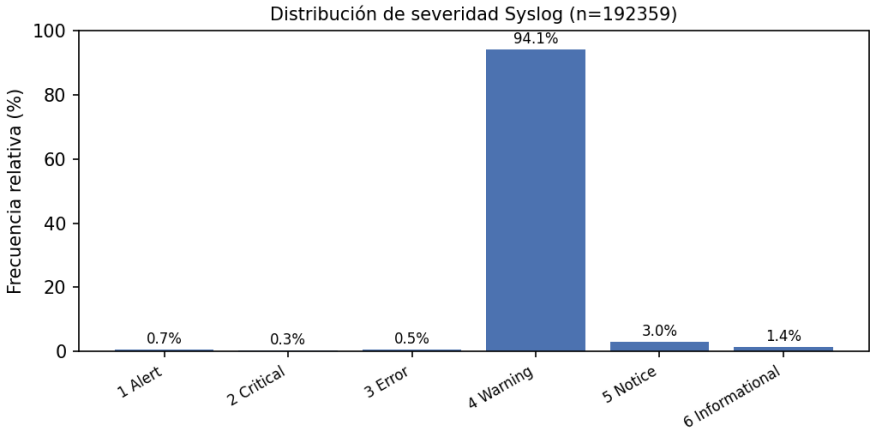


Figura 3.6: Distribución de frecuencia de los niveles de severidad Syslog en el *syslog* crudo ($n = 192\,359$ registros).

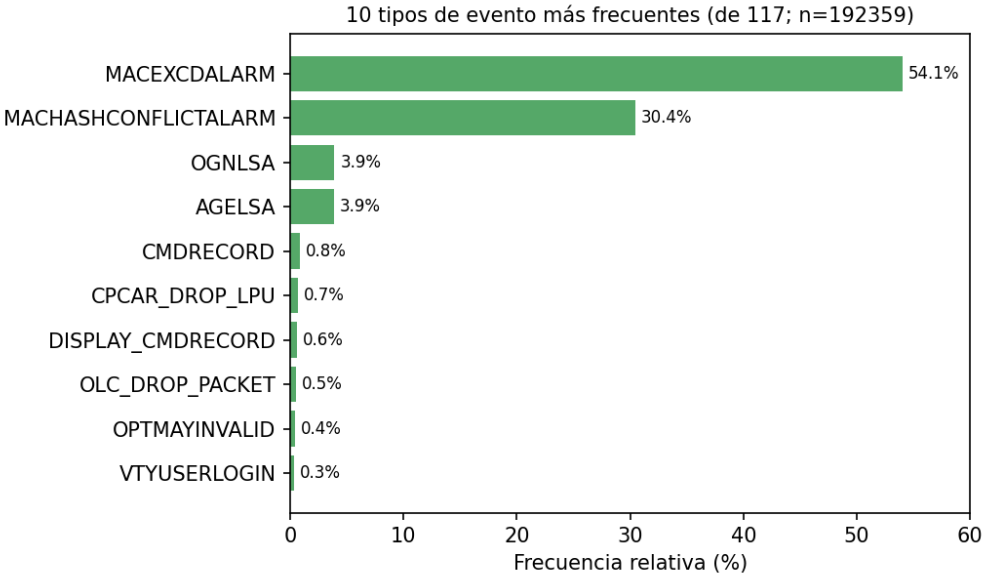


Figura 3.7: Diez tipos de evento (`event_type`) más frecuentes de los 117 presentes en el *syslog* crudo. El campo `event` está dominado por alarmas de capa 2 y no identifica por sí solo los incidentes ópticos, que carecen de un código único.

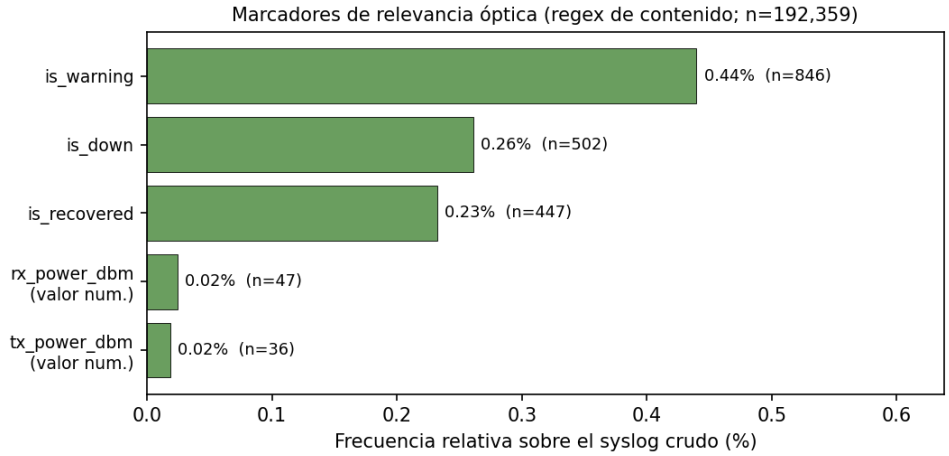


Figura 3.8: Marcadores de relevancia óptica extraídos por expresiones regulares sobre el contenido del mensaje (`is_warning`, `is_down`, `is_recovered` y presencia de valores numéricos RX/TX). En conjunto representan el 0,93 % del *syslog* crudo ($n = 192\,359$).

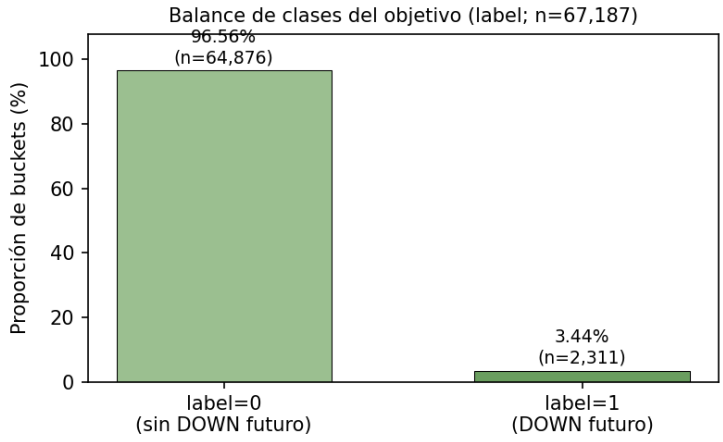


Figura 3.9: Balance de clases del objetivo de aprendizaje (`label`): solo el 3,44 % de los *buckets* de 60 s precede a un evento *DOWN* dentro del horizonte de predicción ($n = 67\,187$).

3.2.4.2. Análisis de valores atípicos (*outliers*)

El análisis de valores atípicos se realizó sobre las **14 características de entrenamiento**, y no sobre las variables crudas del *syslog*, dado que son estas features las que ingresan al modelo tras su estandarización. La Figura 3.10 presenta el diagrama de cajas de las 14 variables en unidades estandarizadas (*z-score*), escala en la que opera el `StandardScaler` del *pipeline*. Su interpretación, sin embargo, exige distinguir la naturaleza de cada grupo de variables, pues la regla clásica del rango intercuartílico (IQR) resulta engañosa en este contexto.

En primer lugar, las variables de **conteo** (`event_count`, `warning_count`, `down_count` y sus derivadas) presentan distribuciones con **fuerte inflación de ceros** (*zero-inflation*): más del 98 % de los *buckets* de 60 s registra un valor nulo. En particular, `event_count` es prácticamente binaria (0 en 66 249 *buckets* y 1 en 938; nunca superior a 1). En consecuencia, los puntos que la regla IQR marca como "atípicos" **no son anomalías ni errores de medición, sino los propios eventos poco frecuentes**: al ser el valor mediano y los cuartiles iguales a cero, cualquier registro no nulo se señala automáticamente como extremo. Se trata, por tanto, de una propiedad estructural de un fenómeno raro, que coincide con la clase positiva a predecir, y no de un problema de calidad de datos que debiera depurarse. Las únicas colas de mayor magnitud entre los conteos corresponden a `down_count` (hasta 9 caídas en un *bucket*), que son episodios de fallo reales y, de nuevo, la señal de interés.

En segundo lugar, la variable `anomaly_score` exhibe los valores estandarizados más altos (máximo de 94,2), pero estos deben leerse con cautela. Dicha característica se define como una **novedad relativa** $\max\big(0,\, z(\texttt{event\_count}) - z(\texttt{rx\_power\_dbm})\big)$ calculada con estandarización de **ventana expansiva** (sólo con el historial pasado de cada interfaz). Al inicio de cada serie, cuando la varianza acumulada es casi nula, la aparición del primer evento genera un *z-score* desproporcionado; en efecto, el máximo global se produce en un *bucket* con un *único* evento y potencia óptica *alta* (−5,65 dBm), lo que confirma que ese pico es un **artefacto de la ventana expansiva** y no una anomalía física severa. Por ello `anomaly_score` se interpreta como un indicador ordinal de novedad respecto al comportamiento propio de cada interfaz, no como una magnitud absoluta comparable entre series. Finalmente, `rx_power_dbm` y `tx_power_dbm` muestran cajas casi degeneradas por su telemetría cuasi-constante, de modo que sus supuestos "atípicos" bajo IQR son igualmente un artefacto de baja variabilidad.

Del análisis anterior se concluye que el conjunto **no contiene valores atípicos espurios que deban podarse** en el sentido tradicional: los aparentes outliers son, o bien la manifestación de un fenómeno raro (inflación de ceros y episodios de fallo), o bien un efecto colateral del propio esquema de estandarización expansiva. En coherencia con ello, no se aplicó ninguna *winsorización*, que eliminaría la clase positiva, y la influencia de las colas queda acotada por diseño: la transformación `event_count_log1p`$= \ln(1 +$ `event_count`$)$ comprime la escala de los conteos; el `clip(lower=0)` sobre `anomaly_score` evita valores negativos; el `StandardScaler` se ajusta únicamente sobre la partición de entrenamiento (evitando fuga de información); y los modelos basados en árboles empleados (*XGBoost*) son intrínsecamente insensibles a la magnitud de los valores extremos.

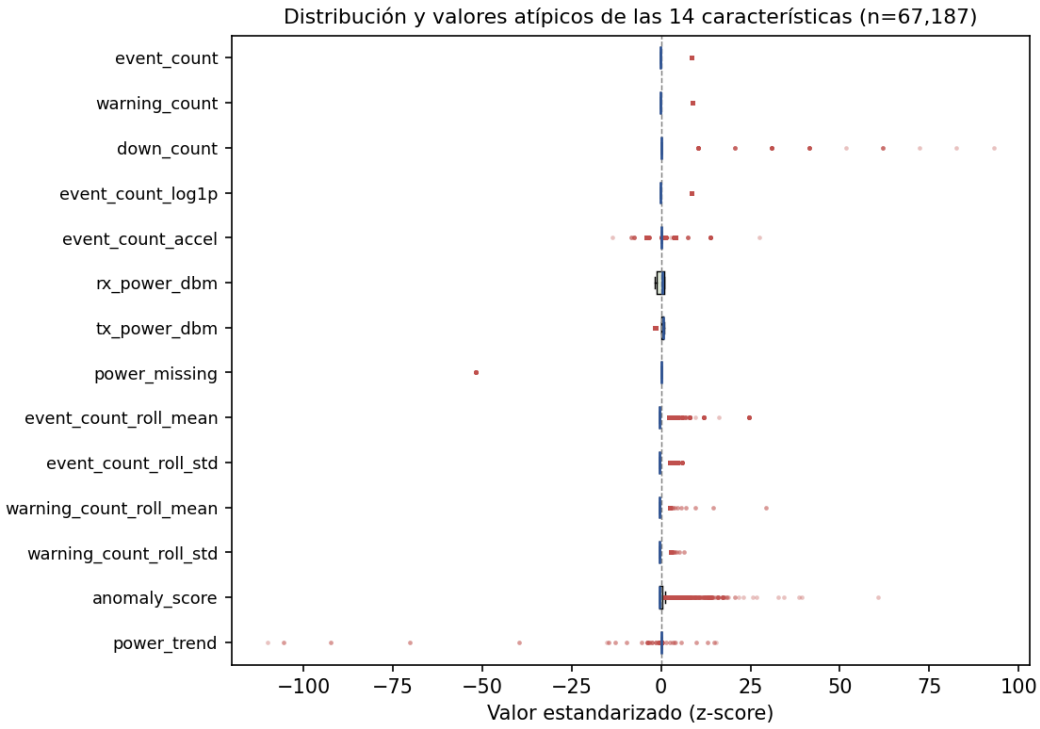


Figura 3.10: Diagrama de cajas de las 14 características de entrenamiento en escala estandarizada (*z-score*, $n = 67\,187$ *buckets*). Los puntos marcados por la regla IQR no corresponden a errores de medición: en las variables de conteo reflejan la *inflación de ceros* propia de un fenómeno raro (los eventos y caídas son la clase positiva), mientras que los valores altos de `anomaly_score` obedecen a la estandarización de ventana expansiva al inicio de cada serie.

3.2.4.3. Autocorrelación temporal

El análisis espectral y la descomposición estacional aplicados sobre la variable `event_count_roll_mean` confirmaron que la señal de eventos opera en escalas sub-horarias y no presenta estacionalidad significativa, es decir, los episodios de falla son aperiódicos (ver Figuras 3.12 y 3.13).

La Función de Autocorrelación (ACF) calculada sobre la misma variable (Figura 3.11) reveló correlación significativa en los primeros rezagos (0 a 3), confirmando que la actividad reciente de eventos es predictiva de la actividad futura inmediata. Este hallazgo validó dos decisiones de diseño: (1) el uso de ventanas deslizantes de 10 *buckets* (10 minutos) como entrada del modelo LSTM, y (2) la selección del orden autorregresivo del modelo ARIMA.

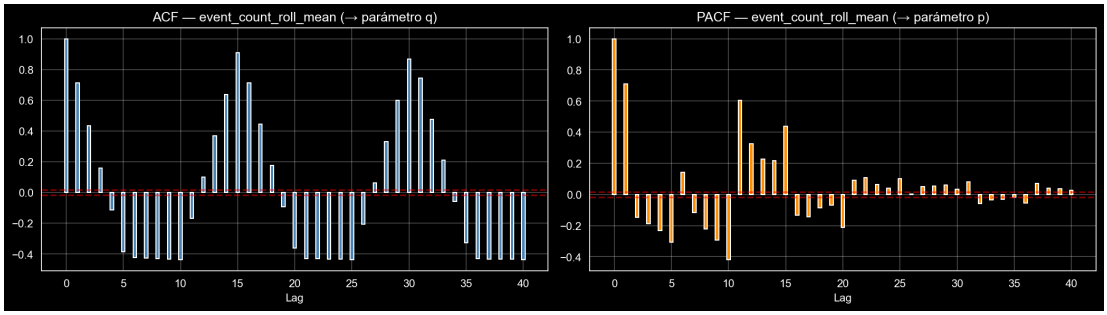


Figura 3.11: Función de Autocorrelación (ACF) de `event_count_roll_mean` en la interfaz foco. La correlación significativa en los primeros rezagos (fuera de las bandas de confianza) confirmó la existencia de memoria de corto plazo en la señal de eventos, validando la ventana de 10 *buckets* del LSTM.

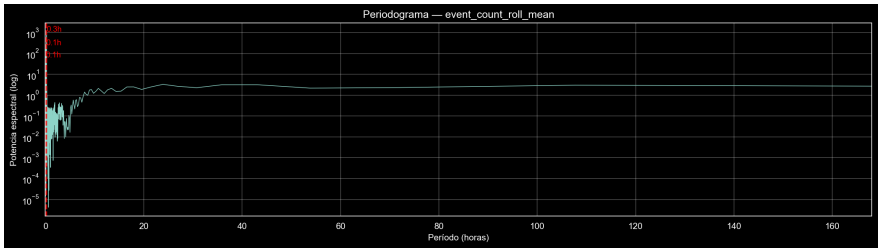


Figura 3.12: Periodograma (FFT) de la variable `event_count_roll_mean` en la interfaz foco `XGigabitEthernet3/0/0`. Los picos de densidad espectral se concentran en períodos de 0.3 h y 0.1 h (18 y 6 minutos), confirmando que la señal de eventos opera en escalas sub-horarias.

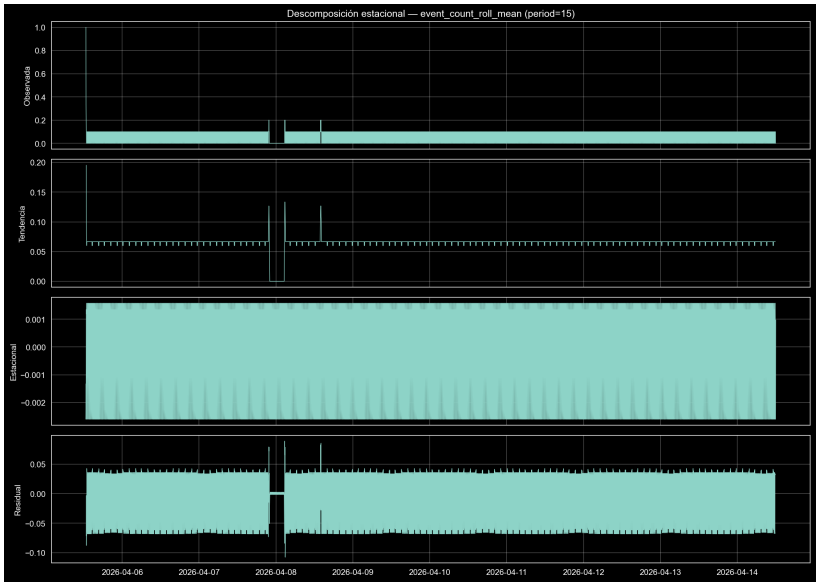


Figura 3.13: Descomposición estacional (STL) de `event_count_roll_mean` con período de 15 *buckets*. La componente estacional presenta amplitud despreciable (∼0.002) frente a la señal observada (∼1.0), confirmando que los eventos de falla son aperiódicos. La componente de tendencia captura los episodios de escalada previos a las caídas de interfaz.

3.2.4.4. Distribución temporal de eventos en la interfaz foco

La Figura 3.14 presenta el diagrama cronológico de actividad de la interfaz foco a lo largo de todo el período de observación, graficando tres variables: `event_count`, `warning_count` y `down_count`. Los eventos de tipo *DOWN* (panel inferior) se concentran en dos clústeres temporales, precedidos por ráfagas de actividad en `event_count` (panel superior), lo que constituyó la evidencia empírica de que existen patrones precursores de falla capturables por un modelo con memoria temporal. La variable `warning_count` resulta **nula en toda la serie** para esta interfaz específica, lo que es consistente con el análisis de importancia de *features* del capítulo de Evaluación. Esta observación no invalida su inclusión en el modelo multi-serie, dado que otras interfaces del dataset sí presentan conteos de advertencias no nulos.

3.2.5. Fase 6 — Modelado estadístico de línea base (`models_stat.py`)

El módulo `models_stat.py` implementa un flujo univariado sobre la interfaz con mayor tasa de fallas. Utiliza el modelo estadístico ARIMA/SARIMA (Box et al., 2015), implementado a través de la clase `SARIMAX` de la librería `statsmodels` (Seabold and Perktold, 2010), para proyectar la variable continua `event_count_roll_mean`. El orden (p, d, q) se selecciona mediante *grid search* por AIC; la componente estacional (P, D, Q, s) se activa únicamente cuando el periodograma detecta estacionalidad significativa, degenerando en un ARIMA puro en caso contrario. Para transformar el

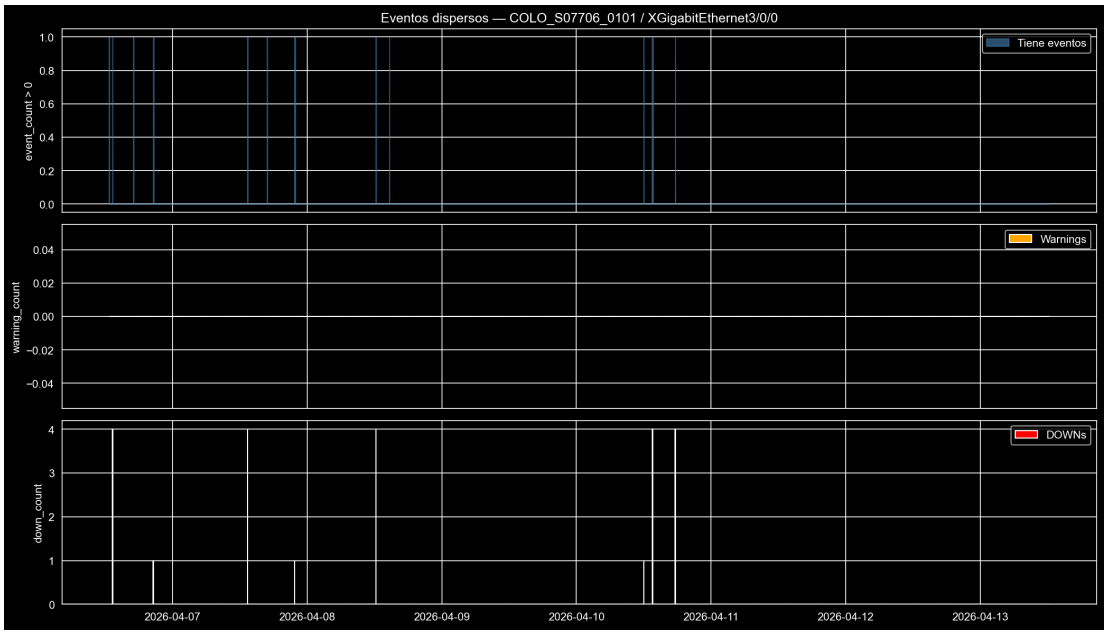


Figura 3.14: Diagrama cronológico de actividad en la interfaz foco `XGigabitEthernet3/0/0`. Panel superior: presencia de eventos (`event_count >0`); panel central: `warning_count` (nulo para esta interfaz); panel inferior: eventos *DOWN*. La concentración de actividad previa a los *DOWN* evidencia patrones precursores de falla.

pronóstico continuo en una clasificación binaria de falla, se implementó un algoritmo de búsqueda en rejilla (*Grid Search*) de 40 candidatos sobre el conjunto de entrenamiento para calibrar dinámicamente el umbral óptimo que maximiza el F1-Score.

3.2.6. Fase 7 — Modelos de aprendizaje automático y LSTM (`models_ml.py`)

El componente principal de predicción se implementó en `models_ml.py` utilizando la arquitectura de red neuronal recurrente **LSTM (Long Short-Term Memory)** (Hochreiter and Schmidhuber, 1997) sobre la librería **PyTorch** (Paszke et al., 2019). Además de la red LSTM, este módulo implementa los modelos de línea base supervisada (Regresión Logística y XGBoost (Chen and Guestrin, 2016) mediante su librería nativa) y no supervisada (Isolation Forest (Liu et al., 2008)), estos últimos implementados mediante *Scikit-learn* (Pedregosa et al., 2011), permitiendo la comparación directa bajo las mismas condiciones de evaluación.

La red LSTM procesa tensores tridimensionales con dimensiones `(samples, window_size, features)`, donde el tamaño de la ventana temporal se fijó en 10 *buckets* (equivalente a 10 minutos de historial de red). La arquitectura de la red se compone de:

1. Una capa recurrente LSTM con 128 unidades internas estructurada para entrenamiento multiserie simultáneo sobre todas las interfaces de la red.

2. Una capa de regularización *Dropout* con una tasa de descarte del 0,3 para mitigar el sobreajuste (*overfitting*) ante el escenario de desbalance extremo del dataset (3,44 % de casos positivos).

3. Capa lineal de salida con función de pérdida `BCEWithLogitsLoss`, ponderada por pesos inversos de clase (`class_weight`) en lugar de técnicas de sobremuestreo sintético (SMOTE) que destruirían la correlación temporal de las series.

3.2.7. Fase 8 — Prototipo CLI y reporte operativo (`predecir.py`)

Como artefacto de transferencia tecnológica, se desarrolló el script ejecutable `predecir.py`. Este script actúa como una interfaz de línea de comandos (CLI) que encapsula el *pipeline* completo en un único flujo de ejecución operativa. Al recibir un archivo de *syslog* como entrada, el prototipo ejecuta de forma secuencial siete pasos internos y genera automáticamente un reporte operativo en formato HTML autocontenido.

Dependencias y artefactos del modelo. Para ejecutar el prototipo, deben existir tres artefactos previamente generados durante el entrenamiento:

- `modelo_lstm.pt`: pesos serializados del modelo LSTM entrenado (formato PyTorch).

- `scaler.pkl`: objeto `StandardScaler` de scikit-learn, ajustado sobre los datos de entrenamiento, que normaliza las 14 *features* a media cero y desviación unitaria.

- `threshold.json`: archivo JSON con el umbral de decisión calibrado (0,4842), la lista de *features* y el tamaño de ventana, generado durante el entrenamiento mediante maximización del F1-Score sobre el subconjunto de validación.

Argumentos de ejecución. El prototipo acepta los siguientes argumentos desde línea de comandos:

- `logfile` (obligatorio): ruta al archivo de *syslog* en texto plano.
- `--output` / `-o`: nombre del reporte HTML de salida. Por defecto: `reporte_YYYYMMDD_HHMM.html`.
- `--dias`: número de días recientes a analizar (por defecto: 30). Permite procesar archivos de cualquier tamaño sin desbordar la memoria al descartar registros anteriores al período de análisis.
- `--year`: año a asignar a los *timestamps* del syslog (los registros en formato syslog estándar no incluyen año). Si no se especifica, el prototipo lo infiere automáticamente desde las marcas temporales de los dispositivos.

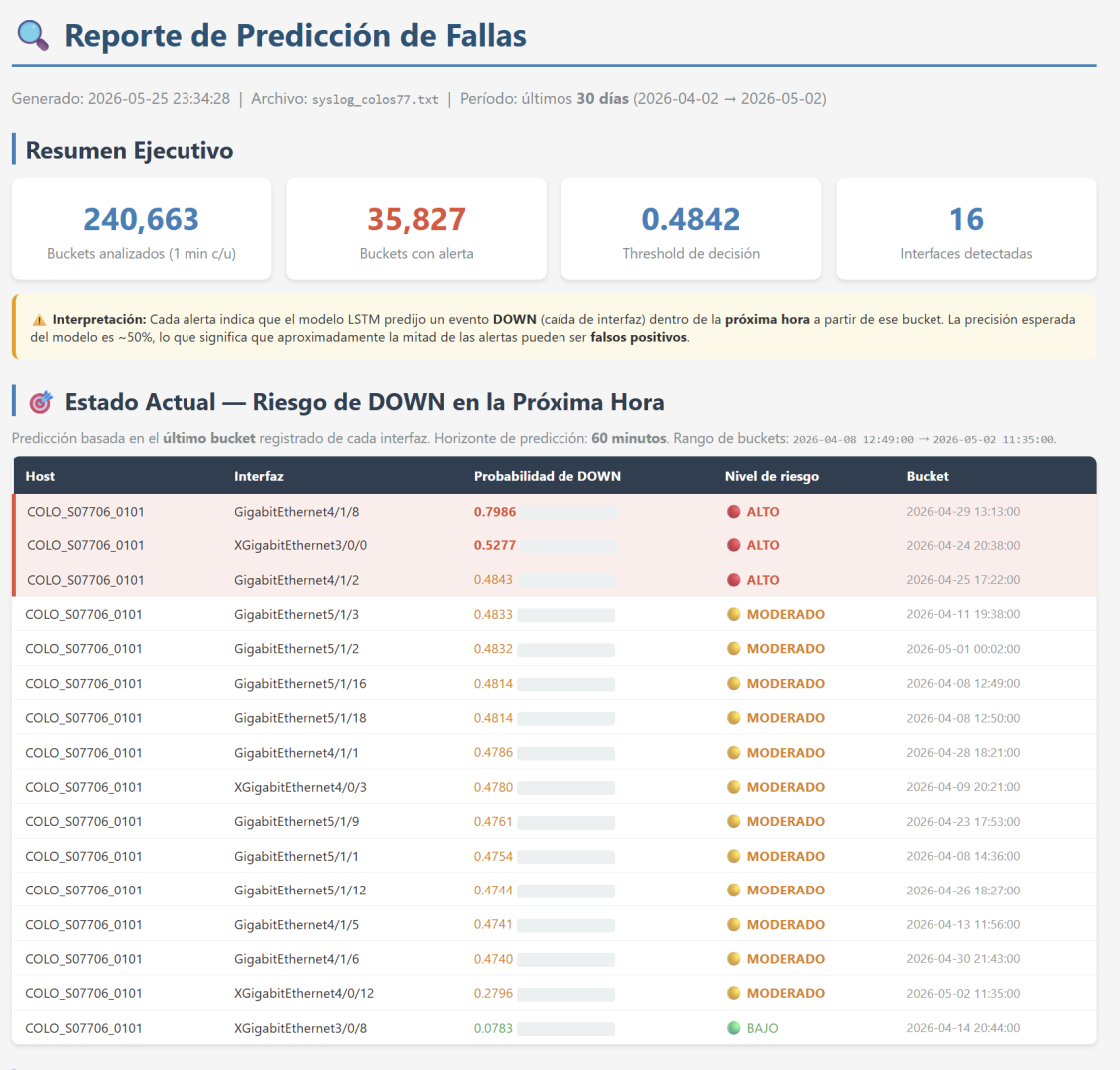
Flujo interno de 7 pasos. El prototipo ejecuta la siguiente secuencia de transformaciones:

1. **Carga de artefactos:** Verifica la existencia de los tres archivos del modelo e inicializa el umbral, el escalador y los pesos de la red LSTM.
2. **Parseo del syslog:** Lee el archivo línea por línea y construye un *DataFrame* con registros estructurados, extrayendo campos como *datetime*, host, interfaz, severidad y potencia óptica.
3. **Filtrado:** Elimina registros sin señal útil y purga subinterfaces lógicas mediante expresión regular, reduciendo el espacio de análisis a las interfaces físicas activas presentes en el dataset.
4. **Consolidación de eventos lógicos:** Fusiona registros sucesivos de un mismo par *(host, interfaz)* separados por menos de 30 segundos respecto al log anterior (umbral de *gap*), reduciendo la redundancia del syslog.
5. **Construcción de series de tiempo:** Genera *buckets* de 60 segundos con los 14 *features* de ingeniería (conteos, estadísticos móviles, telemetría óptica y derivadas combinadas).
6. **Normalización y predicción LSTM:** Aplica el `StandardScaler` sobre las ventanas deslizantes de 10 *buckets* y ejecuta la inferencia mediante el modelo LSTM. Por cada ventana genera una probabilidad entre 0 y 1 que indica la confianza de falla en la próxima hora.
7. **Generación del reporte HTML:** Produce un archivo HTML autocontenido con las siguientes secciones:
   - **Resumen ejecutivo:** total de *buckets* analizados, alertas generadas, umbral activo e interfaces detectadas.

- **Estado actual por interfaz:** tabla con la probabilidad del último *bucket* y clasificación de riesgo semáforo: **Alto** ($p \geq 0{,}4842$), **Moderado** ($0{,}20 \leq p < 0{,}4842$), **Bajo** ($p < 0{,}20$).
- **Historial por interfaz:** resumen acumulado de alertas, probabilidad máxima y media.
- **Distribución de probabilidades:** histograma de *buckets* por rango de riesgo.
- **Historial cronológico de alertas:** primeras 200 alertas ordenadas por tiempo.
- **Metadatos del análisis:** configuración del modelo, umbral, *window size* y lista de *features*.

El prototipo alcanzó un tiempo de ejecución de ∼43 segundos sobre el dataset completo (∼192,000 líneas) operando exclusivamente en CPU, mostrando compatibilidad técnica con entornos de producción estándar sin requerimientos de hardware especializado. Las Figuras 3.15 y 3.16 muestran el reporte HTML generado sobre el dataset de EMCALI, incluyendo el estado de riesgo semáforo por interfaz y el historial de picos de riesgo detectados. El código fuente del prototipo y un ejemplo del reporte generado (`mi_reporte_actualizado.html`) se encuentran disponibles en el repositorio público del proyecto (Cano Salgado and Zapata Castaño, 2026).

El diseño y la utilidad operativa del reporte fueron sometidos a una revisión informal con el equipo del Centro de Operaciones de Red (NOC) de EMCALI, del que uno de los autores es miembro activo, junto con su jefe directo. La retroalimentación recibida fue favorable respecto a la claridad del sistema de semáforo (Alto / Moderado / Bajo) y a la utilidad de las alertas generadas para priorizar la atención sobre las interfaces en riesgo. Esta revisión constituye una validación preliminar de usabilidad y no reemplaza un estudio formal con protocolo de evaluación, el cual se identifica como una línea de trabajo futuro.



Reporte de Predicción de Fallas

Generado: 2026-05-25 23:34:28 | Archivo: `syslog_colos77.txt` | Período: últimos **30 días** (2026-04-02 → 2026-05-02)

Resumen Ejecutivo

240,663	35,827	0.4842	16
Buckets analizados (1 min c/u)	Buckets con alerta	Threshold de decisión	Interfaces detectadas

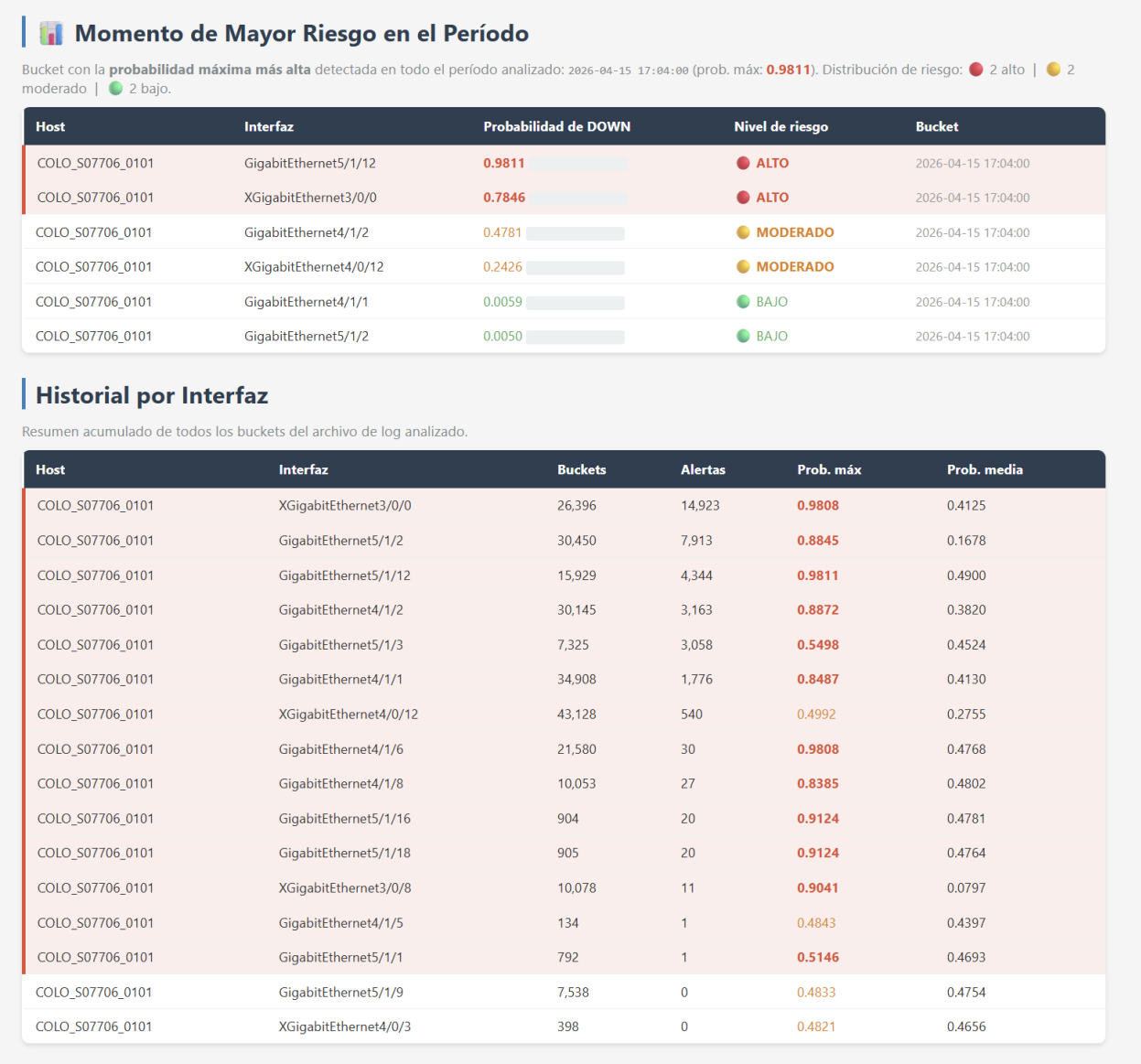
⚠ **Interpretación:** Cada alerta indica que el modelo LSTM predijo un evento **DOWN** (caída de interfaz) dentro de la **próxima hora** a partir de ese bucket. La precisión esperada del modelo es ~50%, lo que significa que aproximadamente la mitad de las alertas pueden ser **falsos positivos**.

Estado Actual — Riesgo de DOWN en la Próxima Hora

Predicción basada en el **último bucket** registrado de cada interfaz. Horizonte de predicción: **60 minutos**. Rango de buckets: `2026-04-08 12:49:00` → `2026-05-02 11:35:00`.

Host	Interfaz	Probabilidad de DOWN	Nivel de riesgo	Bucket
COLO_S07706_0101	GigabitEthernet4/1/8	0.7986	ALTO	2026-04-29 13:13:00
COLO_S07706_0101	XGigabitEthernet3/0/0	0.5277	ALTO	2026-04-24 20:38:00
COLO_S07706_0101	GigabitEthernet4/1/2	0.4843	ALTO	2026-04-25 17:22:00
COLO_S07706_0101	GigabitEthernet5/1/3	0.4833	MODERADO	2026-04-11 19:38:00
COLO_S07706_0101	GigabitEthernet5/1/2	0.4832	MODERADO	2026-05-01 00:02:00
COLO_S07706_0101	GigabitEthernet5/1/16	0.4814	MODERADO	2026-04-08 12:49:00
COLO_S07706_0101	GigabitEthernet5/1/18	0.4814	MODERADO	2026-04-08 12:50:00
COLO_S07706_0101	GigabitEthernet4/1/1	0.4786	MODERADO	2026-04-28 18:21:00
COLO_S07706_0101	XGigabitEthernet4/0/3	0.4780	MODERADO	2026-04-09 20:21:00
COLO_S07706_0101	GigabitEthernet5/1/9	0.4761	MODERADO	2026-04-23 17:53:00
COLO_S07706_0101	GigabitEthernet5/1/1	0.4754	MODERADO	2026-04-08 14:36:00
COLO_S07706_0101	GigabitEthernet5/1/12	0.4744	MODERADO	2026-04-26 18:27:00
COLO_S07706_0101	GigabitEthernet4/1/5	0.4741	MODERADO	2026-04-13 11:56:00
COLO_S07706_0101	GigabitEthernet4/1/6	0.4740	MODERADO	2026-04-30 21:43:00
COLO_S07706_0101	XGigabitEthernet4/0/12	0.2796	MODERADO	2026-05-02 11:35:00
COLO_S07706_0101	XGigabitEthernet3/0/8	0.0783	BAJO	2026-04-14 20:44:00

Figura 3.15: Reporte HTML – resumen ejecutivo y estado actual de riesgo por interfaz. El sistema de semáforo (Alto / Moderado / Bajo) permite al operador del NOC identificar de forma inmediata las interfaces en situación crítica.



Momento de Mayor Riesgo en el Período

Bucket con la **probabilidad máxima más alta** detectada en todo el período analizado: 2026-04-15 17:04:00 (prob. máx: **0.9811**). Distribución de riesgo: 2 alto | 2 moderado | 2 bajo.

Host	Interfaz	Probabilidad de DOWN	Nivel de riesgo	Bucket
COLO_S07706_0101	GigabitEthernet5/1/12	0.9811	ALTO	2026-04-15 17:04:00
COLO_S07706_0101	XGigabitEthernet3/0/0	0.7846	ALTO	2026-04-15 17:04:00
COLO_S07706_0101	GigabitEthernet4/1/2	0.4781	MODERADO	2026-04-15 17:04:00
COLO_S07706_0101	XGigabitEthernet4/0/12	0.2426	MODERADO	2026-04-15 17:04:00
COLO_S07706_0101	GigabitEthernet4/1/1	0.0059	BAJO	2026-04-15 17:04:00
COLO_S07706_0101	GigabitEthernet5/1/2	0.0050	BAJO	2026-04-15 17:04:00

Historial por Interfaz

Resumen acumulado de todos los buckets del archivo de log analizado.

Host	Interfaz	Buckets	Alertas	Prob. máx	Prob. media
COLO_S07706_0101	XGigabitEthernet3/0/0	26,396	14,923	0.9808	0.4125
COLO_S07706_0101	GigabitEthernet5/1/2	30,450	7,913	0.8845	0.1678
COLO_S07706_0101	GigabitEthernet5/1/12	15,929	4,344	0.9811	0.4900
COLO_S07706_0101	GigabitEthernet4/1/2	30,145	3,163	0.8872	0.3820
COLO_S07706_0101	GigabitEthernet5/1/3	7,325	3,058	0.5498	0.4524
COLO_S07706_0101	GigabitEthernet4/1/1	34,908	1,776	0.8487	0.4130
COLO_S07706_0101	XGigabitEthernet4/0/12	43,128	540	0.4992	0.2755
COLO_S07706_0101	GigabitEthernet4/1/6	21,580	30	0.9808	0.4768
COLO_S07706_0101	GigabitEthernet4/1/8	10,053	27	0.8385	0.4802
COLO_S07706_0101	GigabitEthernet5/1/16	904	20	0.9124	0.4781
COLO_S07706_0101	GigabitEthernet5/1/18	905	20	0.9124	0.4764
COLO_S07706_0101	XGigabitEthernet3/0/8	10,078	11	0.9041	0.0797
COLO_S07706_0101	GigabitEthernet4/1/5	134	1	0.4843	0.4397
COLO_S07706_0101	GigabitEthernet5/1/1	792	1	0.5146	0.4693
COLO_S07706_0101	GigabitEthernet5/1/9	7,538	0	0.4833	0.4754
COLO_S07706_0101	XGigabitEthernet4/0/3	398	0	0.4821	0.4656

Figura 3.16: Reporte HTML – momento de mayor riesgo detectado e historial acumulado por interfaz. La columna de probabilidad máxima identifica el pico de riesgo dentro del período analizado.

3.2.8. Despliegue conceptual

Aviso: el diseño descrito en este apartado es estrictamente conceptual y no fue implementado ni validado operativamente.

La Figura 3.17 representa una posible integración del prototipo con la infraestructura existente. La fuente de logs proviene del sistema RMS del operador, entregada mediante un colector o repositorio de *syslog*, con el mismo nivel de detalle utilizado en entrenamiento (mensajes crudos de interfaces con `timestamp`, `hostname`, identificador de interfaz, severidad y `message`). Un orquestador (por ejemplo, `cron`) invocaría al CLI `predecir.py` con la ventana de *syslogs* más reciente y publicaría el reporte HTML resultante para consumo del NOC.

Se contemplan además dos líneas de extensión conceptual: un módulo de MLOps para reentrenamiento periódico de baja frecuencia (por ejemplo, trimestral o disparado por evento, dado el volumen de los archivos de entrada que hace inviable un reentrenamiento continuo), versionado y monitoreo de *drift* del modelo; y la ampliación del canal de consumo hacia plataformas de visualización de alarmas (Zabbix, Grafana), sistemas de *ticketing* o notificaciones *push*.

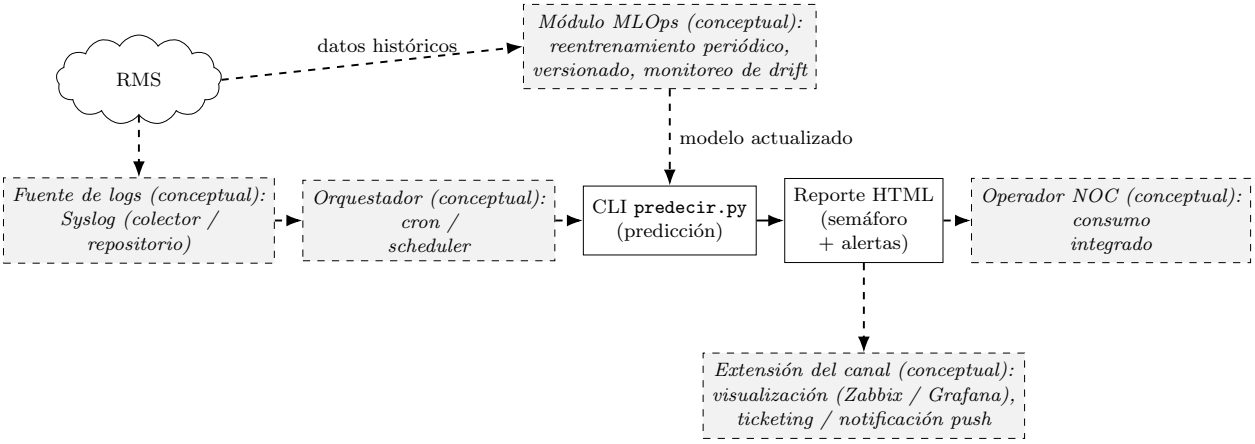


Figura 3.17: Arquitectura conceptual de despliegue del prototipo sobre infraestructura existente **(no implementada, sin validación operativa)**. Los componentes sólidos representan el flujo del prototipo entregado; los componentes discontinuos corresponden a diseño conceptual.

3.3. Resumen del capítulo

Este capítulo detalló el diseño y la implementación del *pipeline* de AIOps en sus ocho fases secuenciales. Se describió cómo un volumen masivo de ∼192k logs crudos de los equipos de la red CORE-MPLS es transformado en un dataset de series temporales mediante la depuración de ruido topológico (exclusión de subinterfaces), la ingeniería de 14 características predictivas y el análisis exploratorio que confirmó la existencia de patrones precursores de falla. El diseño anti-leakage garantiza la integridad temporal del etiquetado, mientras que el entrenamiento multi-serie y la calibración dinámica de umbrales abordan el desafío del desbalance extremo de clases (∼3.4 % de

positivos). La arquitectura multi-modelo permite la comparación objetiva entre enfoques estadísticos y de aprendizaje profundo, y la solución se empaqueta en un prototipo CLI ejecutable que genera reportes HTML autocontenidos para uso operativo en el NOC. Finalmente, el capítulo esboza un diseño de despliegue conceptual (Sección 3.2.8) que representa una posible integración del prototipo con la infraestructura de monitoreo existente, junto con las prácticas de MLOps necesarias para su operación sostenida, sin compromiso de implementación.

Capítulo 4

Evaluación

4.1. Diseño de la evaluación

El protocolo experimental para validar la capacidad predictiva del prototipo AIOps se estructuró garantizando la preservación del orden cronológico de los datos y evitando la contaminación de información del futuro (*look-ahead bias*). A continuación se describen los componentes fundamentales del diseño experimental.

4.1.1. Dataset y partición temporal

El dataset de evaluación corresponde al construido por el *pipeline* de ocho fases descrito en el Capítulo 3: 67,187 *buckets* de series temporales sobre las interfaces de la red CORE-MPLS de EMCALI, cada uno representado por un vector de 14 *features* y una etiqueta binaria ($label = 1$ si se detectó al menos un evento *DOWN* dentro del horizonte de predicción de una hora).

La partición temporal se implementó mediante un esquema **adaptativo** que dividió los datos respetando estrictamente el orden cronológico, sin mezcla aleatoria entre conjuntos:

Cuadro 4.1: Partición temporal del dataset en entrenamiento y prueba, con conteo de *buckets*, positivos y tasa de positivos por partición.

Conjunto	**Buckets**	**Positivos (label=1)**	**Tasa de positivos**
Entrenamiento	51,005	2,161	4.24 %
Prueba (*hold-out*)	16,182	150	0.93 %
Total	**67,187**	**2,311**	**3.44 %**

La diferencia significativa en la tasa de positivos entre entrenamiento (4,24 %) y prueba (0,93 %) constituyó un desafío adicional denominado *covariate shift* temporal, cuyas implicaciones para la transferencia de umbrales se analizan en la Sección 4.2.3.

4.1.2. Estrategia anti-leakage

Para evitar la filtración de información temporal (*data leakage*), el diseño impuso dos restricciones algebraicas en el etiquetado del dataset:

- **Gap temporal** ($\Delta t_{gap} \geq 60$ segundos): separación mínima entre el último dato observado y el inicio del horizonte de predicción, garantizando que ningún *bucket* utilice información síncrona con el evento de falla.

- **Horizonte de predicción** ($\Delta t_{lookahead}$ = 3,600 segundos): ventana de una hora dentro de la cual se buscan eventos *DOWN* para determinar la etiqueta positiva.

Adicionalmente, las estadísticas móviles (`roll_mean`, `roll_std`) y las variables derivadas se calcularon exclusivamente con ventanas expansivas que solo consideraron datos históricos, sin acceso a valores futuros.

4.1.3. Calibración de umbrales

El umbral de clasificación binaria se calibró de manera dinámica para cada modelo, maximizando el F1-Score sobre un subconjunto de validación definido como el último 20 % del conjunto de entrenamiento. Este enfoque garantizó que el umbral fuera determinado **exclusivamente con datos de entrenamiento**, sin acceso al conjunto de prueba. La calibración se adaptó a la distribución de probabilidades específica de cada modelo, en lugar de utilizar un valor fijo arbitrario de 0,5.

4.1.4. Métricas de evaluación

Se seleccionaron las métricas orientadas a la clase minoritaria definidas en la Sección 2.1.6 (Precision, Recall, F1-Score y PR-AUC), por no ser sensibles a la sobreinflación por verdaderos negativos (VN), los cuales dominan en escenarios de desbalance extremo de clases (He and Garcia, 2009). Se descartó la métrica *Accuracy*, ya que un clasificador trivial que predijera siempre la clase negativa alcanzaría una exactitud del 99,07 % en el conjunto de prueba sin detectar ninguna falla.

En la interpretación de dominio de este problema, VP representa los verdaderos positivos (eventos de indisponibilidad *DOWN* anticipados correctamente), FP los falsos positivos (alertas erróneas que generan fatiga de alarmas en el operador) y FN los falsos negativos (fallas no detectadas que resultan en gestión reactiva). La métrica PR-AUC (*Precision-Recall Area Under the Curve*) se incluyó como indicador de la capacidad de *ranking* de cada modelo, independiente del umbral de decisión seleccionado.

4.1.5. Modelos evaluados

Se evaluaron cinco modelos de clasificación que cubrieron las principales categorías del aprendizaje automático:

1. **ARIMA** (línea base estadística): es un modelo de pronóstico univariado (Box et al., 2015) sobre la variable `event_count_roll_mean` de una interfaz de referencia. El pronóstico continuo se transformó a clasificación binaria mediante puntuación por residuos.

2. **Regresión Logística** (supervisado lineal): línea base explicable con penalización L2 y ponderación inversa de clases (`class_weight="balanced"`).

3. **XGBoost** (supervisado no lineal): *gradient boosting* (Chen and Guestrin, 2016) con ponderación de clases mediante el parámetro `scale_pos_weight`.

4. **Isolation Forest** (no supervisado): detección de anomalías (Liu et al., 2008) sin etiquetas de entrenamiento, con tasa de contaminación calibrada según la prevalencia observada en el conjunto de entrenamiento.

5. **LSTM** (aprendizaje profundo): red neuronal recurrente *Long Short-Term Memory* (Hochreiter and Schmidhuber, 1997) implementada en PyTorch (Paszke et al., 2019), con 128 unidades internas, *Dropout* de 0,3, ventana deslizante de 10 *buckets* (10 minutos), función de pérdida `BCEWithLogitsLoss` ponderada por pesos inversos de clase y *learning rate scheduler*. Entrenada en modo multi-serie sobre las 17 interfaces simultáneamente.

4.2. Resultados de la evaluación

Los cinco modelos fueron entrenados y evaluados sobre un entorno de ejecución unificado (CPU). A continuación se presentan los resultados globales consolidados, el análisis por interfaz y los experimentos complementarios.

4.2.1. Resultados globales

Los resultados consolidados sobre el conjunto de prueba (16,182 *buckets*, 150 positivos, 17 interfaces) se presentan en la Tabla 4.2.

Cuadro 4.2: Consolidado de métricas de rendimiento sobre el conjunto de prueba (17 interfaces).

Modelo	Precision	Recall	F1-Score	PR-AUC	VP	FP	FN
LSTM (PyTorch)	2.76 %	**90.0 %**	**5.36 %**	2.28 %	135	4,750	15
ARIMA	2.24 %	64.0 %	4.32 %	2.00 %	48	2,097	27
Regresión Logística	0 %	0 %	0 %	**11.57 %**	0	0	150
XGBoost	0.63 %	10.0 %	1.19 %	1.46 %	15	2,369	135
Isolation Forest	0.95 %	100 %	1.88 %	0.69 %	150	15,636	0

La Figura 4.1 presenta las curvas Precisión-Recall correspondientes a estos resultados.

4.2.2. Análisis por modelo

- **LSTM:** Obtuvo el mejor F1-Score global (Cuadro 4.2), con el Recall más alto entre los modelos con señal útil. El umbral fue calibrado en el conjunto de entrenamiento ($\sim$0.56), garantizando que el resultado no estuviera inflado. La red LSTM demostró capacidad para capturar patrones de memoria secuencial en las ráfagas de eventos precursores de fallas, superando a los modelos que tratan cada observación de forma independiente.

- **ARIMA:** Segundo mejor F1 (4,32 %), operando de forma univariada sobre la media móvil del conteo de eventos. Detectó el 64 % de las fallas con 2,097 falsos positivos. Su transformación

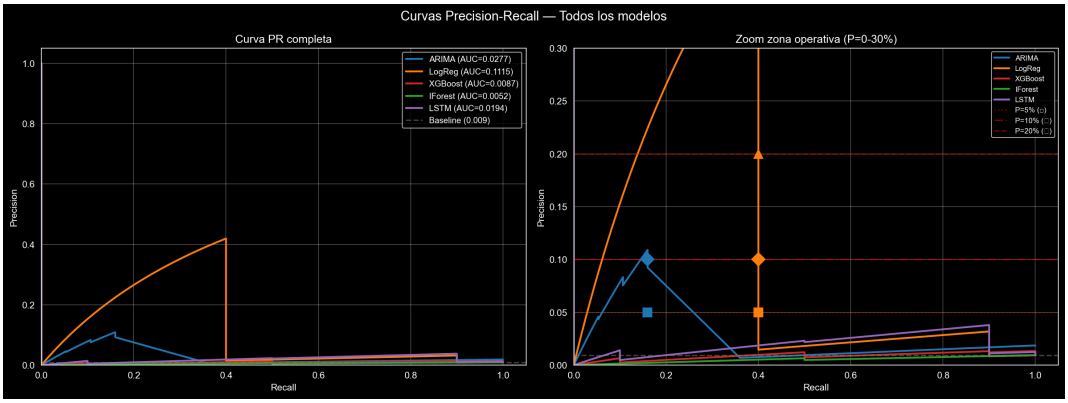


Figura 4.1: Curvas Precisión-Recall (PR) para los cinco modelos evaluados sobre el conjunto de prueba. La curva de Regresión Logística ocupa la posición superior, coherente con su PR-AUC más alto (Cuadro 4.2), pese a no producir predicciones positivas bajo umbral calibrado, evidencia del *covariate shift* temporal. Los valores de AUC mostrados en la leyenda de la gráfica fueron calculados por integración trapezoidal (`sklearn.metrics.auc`), mientras que el Cuadro 4.2 reporta la métrica *Average Precision* (`average_precision_score`), que utiliza interpolación escalonada y es el estándar recomendado; las pequeñas diferencias numéricas entre ambos se deben a este distinto método de integración.

de pronóstico a clasificación binaria resultó más robusta que los modelos de aprendizaje automático clásicos.

- **Regresión Logística:** Presentó el PR-AUC más alto de todos los modelos (11,57 %), lo que indica que logró el mejor *ranking* de probabilidades entre clases. Sin embargo, el umbral calibrado en entrenamiento no produjo ninguna predicción positiva en el conjunto de prueba (0 verdaderos positivos), debido al *covariate shift* temporal entre las distribuciones de entrenamiento y prueba (Cuadro 4.1). Este hallazgo demostró que la Regresión Logística fue el mejor *ranker* pero el peor clasificador en este escenario.

- **XGBoost:** Recall bajo (10 %) con solo 15 de 150 positivos detectados. La corrección de las estadísticas móviles mediante ventanas expansivas alteró la distribución de *features* respecto a versiones anteriores, haciendo que los umbrales calibrados en entrenamiento resultaran excesivamente conservadores para el conjunto de prueba.

- **Isolation Forest:** Recall perfecto (100 %) a costa de marcar 15,636 de 16,182 *buckets* como anomalías (96,6 % del total), resultando en una Precision del 0,95 %. La calibración de la tasa de contaminación basada en la prevalencia del entrenamiento no transfirió adecuadamente a la menor prevalencia del conjunto de prueba (Cuadro 4.1). Este modelo no aportó señal discriminativa útil.

4.2.3. Análisis por interfaz: Hallazgo principal

El análisis de las métricas del modelo LSTM desagregadas por interfaz reveló el **hallazgo principal** de la investigación. Los resultados globales ($F1 = 5{,}36\,\%$, $Precision = 2{,}76\,\%$) estaban **diluidos** por la agregación de 17 interfaces con características heterogéneas. Al evaluar el rendimiento en la interfaz con mayor densidad de telemetría óptica, se obtuvieron resultados operativamente significativos (Cuadro 4.3):

Cuadro 4.3: Métricas LSTM por interfaz representativa (umbral global calibrado en entrenamiento).

Interfaz	Samples	Pos.	Prev.	Prec.	Recall	F1
`XGigabitEthernet3/0/0`	4,123	75	1.82 %	**50.0 %**	**80.0 %**	**61.54 %**
`Eth-Trunk34` (no óptica)	4,113	75	1.82 %	1.83 %	100 %	3.60 %

La interfaz `XGigabitEthernet3/0/0` del equipo `COLO_S07706_0101` alcanzó una **Precisión del** 50 %, un **Recall del** 80 % y un **F1-Score de** 61,54 %. Esto significó que de cada 100 alertas generadas por el modelo, 50 correspondían a fallas reales, y que el sistema fue capaz de anticipar 4 de cada 5 eventos de degradación de potencia óptica con una hora de antelación. Este resultado es consistente con la viabilidad operativa del prototipo para el Centro de Operaciones de Red (NOC) de EMCALI, sujeta a la confirmación mediante un estudio formal de usabilidad.

Dado el tamaño reducido de la muestra positiva (75 eventos positivos de un total de 4 123 observaciones en el conjunto de prueba), se estimó la incertidumbre del F1-Score mediante *bootstrap* no paramétrico: se generaron 10 000 remuestreos con reemplazo del conjunto de prueba, se recalculó el F1 en cada uno, y se tomó el intervalo entre los percentiles 2.5 y 97.5 de la distribución resultante, obteniéndose un intervalo de confianza al 95 % de [52,9 %, 69,0 %]. Como referencia inferior, un clasificador trivial que predijera todos los *buckets* como positivos obtendría un recall del 100 % (marcaría los 75 positivos) y una precision igual a la prevalencia de la clase ($75/4{,}123 = 1{,}82\,\%$), lo que equivale a un F1 de apenas 3,6 %. Que el límite inferior del intervalo (52,9 %) supere ampliamente ese piso confirma que el desempeño observado refleja una capacidad discriminativa real y no un artefacto del tamaño muestral.

En contraste, la interfaz `Eth-Trunk34`, de tipo *link aggregation* (no óptica), obtuvo una Precision de apenas 1,83 % (equivalente a la prevalencia de la clase positiva), indicando que el modelo no aportó información discriminativa sobre esta interfaz. La causa raíz fue la ausencia de separación natural en la distribución de probabilidades: mientras que en `XGigabitEthernet3/0/0` las muestras negativas presentaron probabilidades claramente inferiores al umbral ($< 0{,}5$) y las positivas superiores ($> 0{,}6$), en `Eth-Trunk34` todas las muestras, positivas y negativas, recibieron probabilidades superiores al umbral global ($> 0{,}56$), resultando en la predicción masiva de falsos positivos.

Factores que explicaron el rendimiento diferenciado:

1. **Densidad de telemetría óptica:** `XGigabitEthernet3/0/0` presentó un 99,6 % de datos no nulos en la variable `rx_power_dbm`, proporcionando al modelo una señal óptica continua y densa. Las interfaces sin telemetría óptica carecieron de esta fuente de información.

2. **Patrones de degradación pre-falla:** En la interfaz óptica se observaron incrementos graduales en `event_count` y `warning_count` previos a los eventos *DOWN*, constituyendo patrones temporales que la red LSTM pudo capturar mediante sus compuertas de memoria.

3. **Naturaleza de la interfaz:** Las interfaces de *link aggregation* como `Eth-Trunk34` agrupan múltiples enlaces físicos, generando un comportamiento de eventos cualitativamente diferente al de las interfaces ópticas individuales.

Es preciso hacer explícito, sin embargo, que la interfaz de referencia se seleccionó por ser el caso más representativo dentro del dominio de aplicación del enfoque, y no por comparación entre interfaces equivalentes: el conjunto de 17 interfaces analizadas incluye agregaciones lógicas y puertos con telemetría óptica escasa o nula, para los cuales el sistema no está diseñado, pues su capacidad predictiva depende de la existencia de eventos de potencia óptica. En esa medida, los resultados son extrapolables a otras interfaces en tanto dispongan de una densidad de telemetría óptica comparable, y la degradación observada en el experimento de umbral por interfaz (Sección 4.2.6) refleja principalmente el comportamiento esperado del modelo fuera de su dominio de aplicación, y no una falla de generalización dentro de él.

La Figura 4.2 presenta la línea temporal de predicciones de los cinco modelos sobre el conjunto de prueba completo (16,182 *buckets*, 17 interfaces concatenadas). Esta visualización permite comparar el comportamiento temporal de cada modelo: mientras ARIMA e Isolation Forest generan predicciones positivas de forma casi constante a lo largo de toda la serie (explicando sus altas tasas de falsos positivos), el modelo LSTM es el único que exhibe una estructura temporal diferenciada, donde la probabilidad asignada se eleva en las regiones donde se concentran las fallas reales y desciende por debajo del umbral en las zonas silenciosas. Este comportamiento evidencia la capacidad de las compuertas de memoria de la red LSTM para discriminar entre regímenes temporales de actividad normal y de degradación inminente.

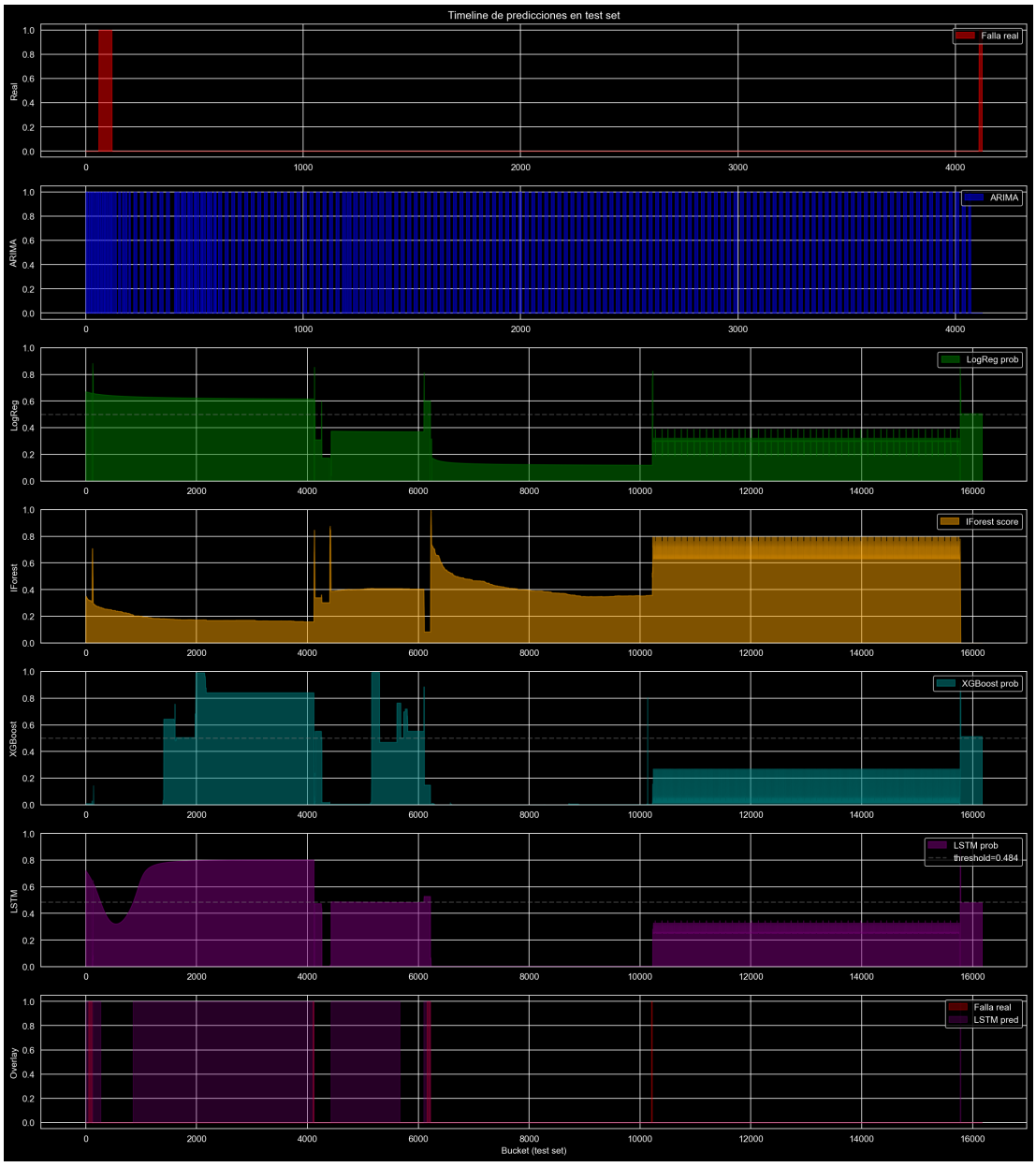


Figura 4.2: Línea temporal de predicciones sobre el conjunto de prueba completo (16,182 *buckets*, 17 interfaces). El panel superior muestra las fallas reales; los paneles siguientes presentan las probabilidades de cada modelo. El LSTM es el único que reduce la probabilidad por debajo del umbral (línea punteada) en regiones sin fallas, evidenciando discriminación temporal. ARIMA e Isolation Forest predicen positivo de forma casi uniforme.

4.2.4. Importancia de *features*

El análisis de los coeficientes absolutos promedio de la Regresión Logística, seleccionada por su interpretabilidad, reveló la contribución relativa de cada variable predictora (Cuadro 4.4):

Cuadro 4.4: Importancia de *features* por coeficientes absolutos de Regresión Logística.

Feature	Coeficiente absoluto	Tipo
`event_count_accel`	1.0254	Transformación (segunda derivada)
`down_count`	0.9263	Conteo directo
`event_count_roll_std`	0.7026	Estadístico móvil
`event_count_roll_mean`	0.6013	Estadístico móvil
`event_count`	0.2847	Conteo directo
`rx_power_dbm`	0.003	Telemetría óptica

Un hallazgo relevante fue que la variable de potencia óptica (`rx_power_dbm`), central en el alcance del proyecto, presentó la menor importancia predictiva individual (coeficiente = 0,003). Esto es coherente con el comportamiento diferenciado por interfaz descrito en la Sección 4.2.3: más que actuar como predictor directo de fallas, la potencia óptica operó como indicador de la **calidad de la telemetría**, pues su presencia densa habilitó al modelo para aprovechar las variables derivadas combinadas (`anomaly_score` y `power_trend`, definidas en el Capítulo 3) que sí contribuyeron a la discriminación. La Figura 4.3 ilustra esta jerarquía de importancias.

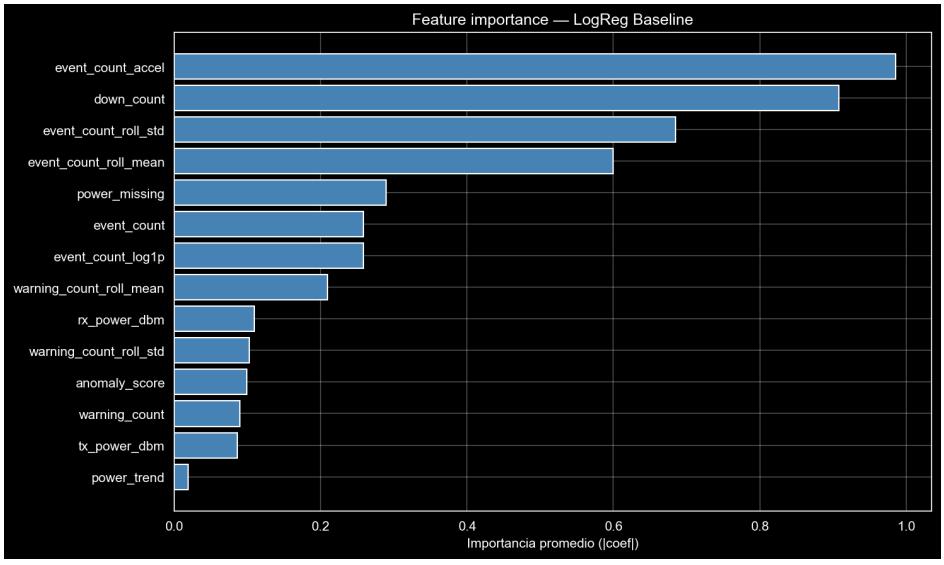


Figura 4.3: Importancia de *features* por magnitud del coeficiente absoluto promedio de la Regresión Logística. Las variables de aceleración de eventos (`event_count_accel`) y conteo de caídas (`down_count`) dominan la contribución predictiva; la potencia óptica presenta el coeficiente más bajo pese a ser la señal distintiva de las interfaces ópticas.

4.2.5. Experimento: entrenamiento *single-interface* vs. *multi-serie*

Se realizó un experimento complementario para evaluar la hipótesis de que el entrenamiento sobre una sola interfaz (la de mayor número de positivos) podría mejorar el rendimiento al eliminar el ruido de interfaces con comportamiento heterogéneo.

Cuadro 4.5: Comparación de F1-Score entre modo multi-serie (17 interfaces) y *single-interface*.

Modelo	Multi-serie F1	Single-interface F1	Variación
LSTM	**5.36 %**	3.60 %	−33 %
ARIMA	4.32 %	4.32 %	0 %
XGBoost	1.19 %	3.60 %	+203 %
Isolation Forest	1.88 %	3.58 %	+90 %

Resultado: La hipótesis fue refutada. El modo *single-interface* **degradó** al modelo LSTM, el mejor clasificador, de $F1 = 5{,}36\,\%$ a $F1 = 3{,}60\,\%$ (una reducción del 33 %). En el modo *single-interface*, todos los modelos de aprendizaje automático tendieron a predecir la totalidad de las muestras como positivas, alcanzando $Recall = 100\,\%$ pero $Precision \approx 1{,}83\,\%$ (equivalente a la prevalencia del conjunto de prueba).

Las causas identificadas fueron:

1. **Pérdida de regularización natural:** En el modo multi-serie, la diversidad de comportamientos entre las 17 interfaces actuó como regularizador implícito, forzando al modelo a aprender patrones generalizables. Con una sola interfaz, el modelo se sobreajustó a los patrones del entrenamiento.

2. ***Covariate shift* amplificado:** La diferencia de prevalencia entre entrenamiento (6,67 % en la interfaz seleccionada) y prueba (1,82 %) fue más severa que en el modo multi-serie, haciendo que los umbrales calibrados en entrenamiento no transfirieran al conjunto de prueba.

Este resultado confirmó que el **modo multi-serie fue superior** y se adoptó como configuración definitiva del prototipo.

4.2.6. Experimento: umbral por interfaz

Se exploró la posibilidad de calibrar umbrales independientes por interfaz, con el objetivo de adaptar el punto de corte a la distribución de probabilidades específica de cada serie temporal. Se implementaron dos iteraciones:

1. **Iteración 1 (maximización de F1 en validación por interfaz):** El F1 global del LSTM descendió de 5,36 % a 3,59 %. Con solo ∼15 positivos por interfaz en validación, la maximización de F1 favoreció sistemáticamente predecir todo como positivo.

2. **Iteración 2 (objetivo de Precision $\geq 5\,\%$ en entrenamiento por interfaz):** El F1 global descendió aún más, a 1,92 %. El *covariate shift* entre entrenamiento ($\sim$4 % positivos) y prueba ($\sim$1 % positivos) hizo que los umbrales calibrados en entrenamiento fueran demasiado permisivos en el conjunto de prueba.

Conclusión: El umbral por interfaz no fue viable con el volumen de datos disponible. El umbral **global** funcionó para `XGigabitEthernet3/0/0` porque la separación de probabilidades en esa interfaz fue **natural**, independiente del punto de corte específico. En interfaces sin señal discriminativa, ningún umbral, global o local, podía resolver la falta de información predictiva en las *features* disponibles.

4.3. Resumen del capítulo

En este capítulo se presentó la validación empírica del prototipo predictivo de AIOps. Se evaluaron cinco modelos de distinta naturaleza algorítmica sobre 67,187 *buckets* de series temporales extraídos de la red CORE-MPLS de EMCALI, bajo un diseño experimental con separación temporal estricta y calibración de umbrales exclusivamente sobre datos de entrenamiento.

A nivel global (17 interfaces), todos los modelos presentaron precisiones inferiores al 5 %, explicado por el desbalance extremo de clases, el *covariate shift* temporal y la heterogeneidad entre interfaces. Sin embargo, el análisis desagregado reveló que el modelo LSTM alcanzó un **F1-Score de** 61,54 % (IC 95 % [52,9 %, 69,0 %] mediante *bootstrap* no paramétrico) en la interfaz óptica `XGigabitEthernet3/0/0`, un resultado operativamente significativo que aporta evidencia sobre la capacidad del prototipo para generar alertas tempranas con una hora de anticipación en interfaces con telemetría óptica densa.

Los experimentos complementarios confirmaron que el modo multi-serie fue superior al *single-interface* y que la calibración de umbrales por interfaz no fue viable con el volumen de datos disponible. El hallazgo sobre la importancia diferenciada de la variable de potencia óptica, baja importancia predictiva individual pero alta correlación con la calidad de la telemetría, constituyó un aporte para futuros trabajos en predicción de fallas en redes ópticas.

Capítulo 5

Conclusiones

La ejecución del presente proyecto permitió cumplir de manera satisfactoria el objetivo general, orientado al desarrollo de un prototipo de clasificación de anomalías en archivos LOGFILES de la red CORE-MPLS de EMCALI, con el propósito de apoyar la toma de decisiones del equipo de soporte técnico frente a eventos de potencia óptica.

En relación con el primer objetivo específico, se realizó un análisis exhaustivo de los registros syslog provenientes de los equipos de la red CORE-MPLS de EMCALI. Este proceso permitió identificar eventos relevantes asociados a la potencia óptica en las interfaces de comunicación, evidenciando la naturaleza heterogénea y no estructurada de los datos, así como la presencia de patrones significativos en los registros.

Respecto al segundo objetivo específico, se definieron criterios claros para la identificación de anomalías en los eventos de potencia óptica. Se determinó que las fallas se manifiestan mediante ráfagas logarítmicas de advertencias específicas y cambios acelerados en variables físicas. Este hallazgo permitió establecer formalmente los conceptos de *Gap* y *Lookahead*, reduciendo el riesgo de sesgo por filtración de datos (*data leakage*) y mejorando la calidad del proceso de etiquetado.

En cuanto al tercer objetivo específico, se llevó a cabo la selección de modelos de clasificación adecuados, evaluando tanto enfoques estadísticos tradicionales como métodos no supervisados. Los resultados evidenciaron que dichos modelos no son suficientes para abordar la complejidad multidimensional ni el desbalance extremo de los datos. En este contexto, se identificó que las redes neuronales recurrentes tipo LSTM son más apropiadas para capturar la dinámica temporal de los eventos.

En relación con el cuarto objetivo específico, se evaluó el desempeño de los modelos mediante métricas de precisión, recall y F1-score. El hallazgo principal de la evaluación fue que el modelo LSTM, al analizar su rendimiento desagregado por interfaz, alcanzó un **F1-Score de** 61,54 % (Recall del 80 %) en la interfaz óptica `XGigabitEthernet3/0/0`, logrando un equilibrio operativamente útil entre la detección de anomalías y la reducción de falsas alarmas: el sistema fue capaz de anticipar 4 de cada 5 eventos de degradación de potencia óptica con una hora de antelación. A nivel global, en cambio, el desempeño se vio diluido por la inclusión de interfaces sin telemetría óptica densa, donde el modelo no disponía de información predictiva suficiente para discriminar entre estados normales y anómalos; el detalle cuantitativo de estas métricas se presenta en el capítulo de evaluación.

Finalmente, en cumplimiento del quinto objetivo específico, se desarrolló un prototipo funcional implementado en Python, estructurado como un pipeline modular e integrado mediante una interfaz de línea de comandos. Este prototipo permite la identificación anticipada de anomalías en los LOGFILES, habilitando un enfoque proactivo en la gestión de la red. En particular, el sistema

proporciona una ventana de anticipación aproximada de una hora para la ejecución de acciones preventivas, como el desvío de tráfico MPLS, antes de que se presente una interrupción total del servicio. Este resultado representa un avance significativo frente a los esquemas reactivos tradicionales, contribuyendo a mejorar la disponibilidad y confiabilidad de la red de telecomunicaciones.

En conjunto, los resultados obtenidos demuestran la viabilidad técnica del uso de modelos de aprendizaje profundo para la clasificación de anomalías en entornos de redes de telecomunicaciones y aportan evidencia preliminar de su viabilidad operativa sobre el escenario evaluado, respaldada por la validación informal del prototipo con el equipo del NOC de EMCALI del que uno de los autores forma parte. La confirmación plena de su aplicabilidad en contextos reales de operación requeriría un estudio formal de usabilidad y un despliegue sobre un conjunto ampliado de nodos.

5.1. Relación con la Literatura Existente

Los hallazgos de esta tesis se alinean y expanden las corrientes contemporáneas del paradigma de *Artificial Intelligence for IT Operations* (AIOps). La literatura clásica en minería de logs (v.g., arquitecturas basadas en agrupamientos estáticos o reglas heurísticas de correlación de eventos) suele fallar en entornos de redes densas debido a la rigidez ante la naturaleza cambiante del tráfico.

Este estudio valida las conclusiones de investigaciones internacionales recientes que posicionan a las arquitecturas de aprendizaje profundo secuencial (Deep Learning) como el estándar de rendimiento para la predicción de fallas en redes de transporte óptico. Al incorporar la técnica de funciones de pérdida ponderadas inversamente en PyTorch en lugar de implementar sobremuestreos sintéticos tradicionales como SMOTE, los cuales alteran artificialmente la coherencia temporal de las series de tiempo, esta investigación aporta una solución metodológica limpia y reproducible para el manejo del desbalance extremo en *syslogs* de telecomunicaciones sin distorsionar la física del problema de degradación de potencia.

5.2. Implicaciones Prácticas

La adopción del prototipo desarrollado tiene el potencial de impactar positivamente la eficiencia operativa y los Acuerdos de Niveles de Servicio (SLA) de EMCALI E.I.C.E. E.S.P.:

- **Reducción del MTTR (*Mean Time To Repair*):** Al automatizar la extracción de características complejas (como la aceleración de logs y tendencias lineales de potencia óptica), el NOC disminuye el tiempo invertido en el diagnóstico inicial forense del problema.
- **Optimización de Recursos en Campo:** El reporte operativo HTML generado automáticamente por la CLI provee una clasificación de riesgo semáforo (Alto, Moderado, Bajo), permitiendo programar cuadrillas técnicas de mantenimiento preventivo en la planta externa óptica exclusivamente sobre los nodos degradados de forma priorizada.
- **Mitigación del Impacto al Usuario Final:** Disponer de una hora de ventana predictiva faculta a los protocolos de enrutamiento dinámico de la red CORE-MPLS para re-enrutar el tráfico de datos de manera controlada, reduciendo el impacto operativo de las caídas intempestivas de enlaces troncales de conectividad corporativa y residencial.
- **Posicionamiento Estratégico en la Transformación Digital:** La integración del paradigma AIOps en las operaciones de EMCALI representa un avance concreto hacia la madurez digital en la gestión de redes, situando a la empresa entre los operadores de telecomunicaciones del sector público colombiano que adoptan modelos de inteligencia artificial en sus procesos de soporte crítico. Esta capacidad fortalece la competitividad institucional y sienta una base metodológica replicable para futuras iniciativas de automatización en la organización.
- **Escalabilidad a Otros Servicios de la Empresa:** La arquitectura modular del *pipeline*, centrada en la transformación de registros de texto libre en ventanas temporales de

características, no está acoplada exclusivamente a los eventos de potencia óptica de la red CORE-MPLS. Con los ajustes correspondientes en el módulo de *parsing* y en el esquema de etiquetado, la metodología es aplicable a otros activos críticos gestionados por EMCALI.

5.3. Limitaciones de la Investigación

A pesar de los resultados alcanzados en la interfaz óptica validada, se reconocen las siguientes limitaciones técnicas en el presente estudio:

1. **Dependencia de la Densidad del Log Histórico:** El pipeline es altamente sensible a la consistencia de los servidores de centralización de logs (Syslog Daemons). Interrupciones en el canal de telemetría UDP/TCP de los enrutadores que provoquen la pérdida de registros sesgan el cálculo de las estadísticas móviles del modelo.

2. **Especificidad del Proveedor de Hardware:** El módulo de parseo y filtrado topológico se calibró específicamente para la semántica y convenciones de eventos de los sistemas operativos de los equipos de la red CORE-MPLS. Su despliegue sobre entornos de otros fabricantes (v.g., Cisco, Juniper) requiere una fase previa de re-ingeniería y mapeo de las expresiones de texto.

3. **Fenómenos Estocásticos Impredecibles:** El sistema está diseñado para capturar degradaciones graduales de potencia óptica (v.g., atenuación por suciedad, envejecimiento del láser o dobleces macroscópicos del hilo de fibra). Anomalías de carácter puramente estocástico o instantáneo, tales como un corte físico de fibra por obra civil o vandalismo, quedan fuera del alcance del modelado predictivo al no presentar firmas previas en los registros de eventos.

4. **Volumen y Cobertura del Conjunto de Datos:** En la operación regular de EMCALI este nivel de detalle de telemetría óptica no representaba una ganancia inmediata, por lo que no se mantenía activo ni se persistía. Para el proyecto se habilitó de manera temporal sobre hosts representativos, con un dispositivo dedicado a almacenar el flujo de *syslog* resultante, y se desactivó al concluir la captura. El volumen autorizado, condicionado por costo, quedó restringido a dos equipos frente a los múltiples nodos que componen la topología real de la red CORE-MPLS de EMCALI. Si bien estos equipos fueron seleccionados por su representatividad operativa, los resultados constituyen evidencia sobre el desempeño del enfoque en dichos casos y no una caracterización cuantitativa exhaustiva de la red en su conjunto, cuya validación requeriría una habilitación de la telemetría a mayor escala.

Capítulo 6

Riesgos Éticos

En este capítulo se describe la forma en que fueron identificados, evaluados y gestionados los riesgos éticos asociados al desarrollo del proyecto *"Prototipo de identificación y clasificación de anomalías provenientes de logfiles de la red CORE-MPLS de EMCALI para el evento de potencia óptica de la interfaz de comunicación"*. A continuación se presentan los principales riesgos identificados y las medidas adoptadas para su gestión.

6.1. Uso responsable de la información

Los datos fueron utilizados exclusivamente con fines académicos y de investigación, evitando cualquier uso indebido o no autorizado.

6.2. Privacidad y confidencialidad

Los archivos *logfiles* generados por los switches de la red CORE-MPLS contienen información sensible relacionada con la infraestructura tecnológica de EMCALI, incluyendo direcciones IP, estados operativos y parámetros de comunicación. Un manejo inadecuado de estos datos podría haber comprometido la seguridad de la red y la continuidad del servicio.

Con el fin de mitigar este riesgo, se implementaron diversas medidas de control. En primer lugar, se restringió el acceso a la información únicamente a los ingenieros autorizados, garantizando un manejo controlado de los datos. Asimismo, el procesamiento de la información se realizó en entornos controlados y seguros, evitando la exposición a riesgos externos.

Adicionalmente, se aseguró la exclusión de información sensible en los resultados y productos derivados del proyecto, con el propósito de preservar la confidencialidad de la infraestructura tecnológica. Finalmente, se cumplió con las políticas internas de la organización y con la normativa colombiana vigente en materia de protección de datos personales, en particular la Ley 1581 de 2012 (Congreso de la República de Colombia (2012)).

6.3. Sesgos en el modelo

El desempeño del prototipo dependió directamente de la calidad y representatividad de los datos utilizados durante las fases de entrenamiento y validación. Se identificó que el uso de información incompleta o desbalanceada podría generar sesgos en el modelo, ocasionando falsos positivos o negativos en la detección de anomalías, afectando así la eficiencia operativa del sistema.

Para gestionar este riesgo se adoptaron decisiones metodológicas orientadas a una evaluación honesta del modelo: el uso de métricas robustas al desbalance de clases en lugar de la exactitud, una separación temporal estricta entre entrenamiento y prueba que evitó la filtración de información futura, el análisis desagregado por interfaz para distinguir en qué condiciones el modelo opera con señal discriminativa real, y la calibración del umbral de decisión sobre un conjunto de validación independiente. El detalle de estas decisiones y sus resultados se presenta en el capítulo de evaluación (Sección 4.2.1).

Estas estrategias permitieron reducir el impacto de los sesgos en el modelo, fortaleciendo la confiabilidad de los resultados y su aplicabilidad en un contexto operativo real.

6.4. Transparencia y explicabilidad

La adopción de modelos de aprendizaje profundo en contextos operativos críticos exigió que sus decisiones fueran auditables y comprensibles para los ingenieros del Centro de Operaciones de Red (NOC), quienes actúan como usuarios finales del sistema.

Para garantizar la transparencia, el prototipo genera reportes analíticos interpretables que exponen de forma trazable el riesgo por interfaz, se apoyó en un modelo interpretable para analizar la influencia de las variables sobre las predicciones, y se documentó el *pipeline* y los criterios de etiquetado para asegurar la reproducibilidad y la auditabilidad del proceso. La descripción técnica de estos mecanismos se encuentra en el capítulo de desarrollo (Sección 3.2). En todo caso, las decisiones operativas finales permanecieron bajo la responsabilidad del personal técnico humano del NOC: el sistema se utilizó como herramienta de apoyo a la toma de decisiones y no como un mecanismo autónomo de ejecución de acciones sobre la red.

Estas medidas permitieron asegurar la transparencia y explicabilidad del modelo, fortaleciendo la confianza en su uso dentro de un entorno operativo real.

6.5. Impacto operativo

Las alertas generadas por el prototipo presentaron el potencial de afectar la operación de la red en caso de no ser gestionadas adecuadamente. Se identificó que un volumen elevado de falsas alarmas podría generar fatiga de alertas en el equipo de soporte, reduciendo la confianza en el sistema y disminuyendo la efectividad de la respuesta ante fallas reales.

Para mitigar este riesgo, la clasificación de riesgo se priorizó de modo que el personal técnico pudiera concentrar su atención en las interfaces de mayor criticidad, reduciendo el esfuerzo cognitivo de la revisión manual. Además, el prototipo se validó en un entorno experimental antes de cualquier despliegue, con seguimiento de la tasa de falsas alarmas, y su diseño modular permite actualizar el modelo con nuevos datos históricos sin alterar la interfaz de usuario, facilitando la adaptación continua ante cambios en el comportamiento de la red. El detalle del reporte de riesgo y del protocolo de validación se presenta en los capítulos de desarrollo y evaluación (Secciones 3.2 y 4.2.1).

Estas medidas permitieron mitigar el impacto operativo del prototipo, favoreciendo su integración como herramienta de apoyo confiable en la gestión de la red.

La gestión integral de estos riesgos aseguró que el desarrollo del prototipo se realizara bajo principios éticos sólidos, alineados con las buenas prácticas en ingeniería de software y en el manejo responsable de la información, minimizando posibles impactos negativos y garantizando la integridad del trabajo realizado.

Capítulo 7

Lecciones Aprendidas

7.1. Lecciones Aprendidas

El ciclo de vida del desarrollo de este proyecto aportó aprendizajes críticos en la intersección de la ciencia de datos y la ingeniería de telecomunicaciones:

- **La Importancia Crítica de la Limpieza Topológica:** En las fases iniciales de la investigación, el volumen masivo de logs lógicos inflaba artificialmente las métricas. La implementación del filtro restrictivo de subinterfaces lógicas en el módulo de parseo evidenció que en problemas de infraestructura física, el conocimiento experto del dominio de redes es más determinante para el éxito del modelo que la complejidad algorítmica en sí misma.
- **El Balance entre Complejidad y Despliegue Operativo:** Diseñar una red LSTM profunda representó un reto de optimización de código. La decisión técnica de restringir el pipeline a ejecución exclusiva en CPU e implementar serialización estricta mediante `.pt` y `.pkl` demostró que los modelos de inteligencia artificial aplicados a la industria deben diseñarse pensando en las restricciones del entorno de producción host, priorizando la ligereza y la independencia de librerías propietarias sobre la maximización marginal de métricas a costa de hardware inaccesible.
- **El Desplazamiento Covariante como Advertencia de Evaluación:** La diferencia estadística entre la distribución de positivos en el conjunto de entrenamiento (4,24 %) y en el de prueba (0,93 %) constituyó una lección de rigor metodológico: la evaluación de modelos de series de tiempo sobre infraestructura real debe contemplar explícitamente la posibilidad de un cambio en la distribución temporal de los datos (*covariate shift*). La evaluación por interfaz resultó más informativa que las métricas globales, evidenciando que un modelo con Recall global del 90 % puede resultar operativamente inútil en interfaces sin telemetría óptica densa, mientras que en aquellas donde los datos físicos están disponibles el rendimiento es cualitativamente superior.

Capítulo 8

Trabajos Futuros

8.1. Trabajos Futuros

Como líneas de continuidad científica y tecnológica derivadas de esta tesis, se proponen los siguientes desarrollos futuros:

- **Migración hacia Arquitecturas de Aprendizaje por Transferencia y Transformadores (*Transformers*):** Investigar el rendimiento de modelos basados en mecanismos de atención (como *LogBERT* o variantes de *GPT* ajustadas a dominios de TI) para evaluar si la atención contextual mejora la detección de alertas tempranas frente a las neuronas LSTM sin incrementar críticamente el costo computacional.
- **Integración de Telemetría Streaming Directa (gRPC):** Sustituir la ingesta basada en archivos de logs estáticos por un pipeline de procesamiento de flujos en tiempo real continuo utilizando el protocolo gRPC Network Management Interface (gNMI), capturando directamente las métricas del transceptor óptico digital (DOM) sin pasar por el formateo intermedio de texto libre del *syslog*.
- **Mecanismos de Mitigación Automatizada (*Closed-Loop Automation*):** Extender el script CLI para que, además de generar el reporte operativo en formato HTML, interactúe de forma autónoma con las API de los controladores de red de EMCALI (Redes Definidas por Software - SDN), desencadenando de manera automática políticas de aislamiento preventivo en la interfaz de más alto riesgo.
- **Extensión a Entornos Multi-Fabricante y Nuevos Tipos de Falla:** Generalizar el módulo de parseo y filtrado topológico mediante una capa de abstracción configurable que permita mapear la semántica de eventos de otros proveedores de equipos de red (Cisco, Juniper, ZTE). De manera paralela, ampliar el catálogo de anomalías detectables más allá de la degradación de potencia óptica, incorporando patrones de falla lógica (oscilaciones de BGP, *flapping* de interfaces) e inestabilidades en la capa de transporte MPLS.
- **Adopción de prácticas de MLOps:** Incorporar el ciclo de vida operativo del modelo esbozado en la Sección 3.2.8 (reentrenamiento periódico, versionado y monitoreo de *drift* en producción).

Glosario

ACCURACY Métrica del porcentaje de clasificaciones correctas de un modelo de aprendizaje supervisado.

AIOps (*Artificial Intelligence for IT Operations* / Inteligencia Artificial para las Operaciones de TI), paradigma que aplica técnicas de *big data* y aprendizaje automático a la gestión de la infraestructura tecnológica de una organización. Estructura un flujo continuo de cuatro etapas: recolección agnóstica de datos (métricas, logs, eventos, alertas y trazas), parseo y estructuración automatizada, análisis y detección mediante *machine learning*, y automatización de respuestas (orquestación).

ARIMA (*AutoRegressive Integrated Moving Average*), modelo estadístico univariado empleado para series temporales que presentan tendencia pero carecen de componentes estacionales (ciclos repetitivos fijos). Se define por tres parámetros (p, d, q), correspondientes a los términos autorregresivo (AR, p), de integración o diferenciación (I, d) y de media móvil (MA, q).

BUCKET (o cubo/contenedor), bloque de tiempo de duración fija en el que se agrupan los registros de *log* crudos para convertirlos en métricas numéricas. Al dividir el tiempo continuo en intervalos discretos regulares (por ejemplo, de 60 segundos), se obtiene la estructura tabular uniforme que requieren los modelos de aprendizaje automático.

CELDA En el contexto de una red LSTM, la celda (*cell*) es la unidad funcional básica de la red neuronal. Cada celda procesa el estado correspondiente a un paso temporal de la secuencia (un *bucket*), manteniendo y actualizando la memoria interna que se propaga a lo largo de la serie.

CLI (Command Line Interface / Interfaz de Línea de Comandos), herramienta de software que permite al usuario interactuar con un sistema mediante comandos de texto introducidos en una terminal, sin necesidad de una interfaz gráfica.

COVARIATE SHIFT Fenómeno estadístico donde la distribución de las variables de entrada (*features*) cambia entre los datos de entrenamiento y los datos de prueba o producción, aunque la relación entre las variables y la variable objetivo permanece teóricamente igual. Afecta la capacidad de transferencia de los umbrales calibrados durante el entrenamiento.

DATA LEAKAGE (Filtración de datos), ocurre cuando información del futuro o del conjunto de prueba se filtra inadvertidamente hacia el proceso de entrenamiento del modelo, produciendo estimaciones de rendimiento artificialmente infladas que no se reproducen en producción. En series temporales se previene mediante la separación estricta del orden cronológico y el uso de un GAP temporal.

GAP En el contexto de AIOps, análisis de logs y predicción de fallas en series temporales, el GAP (brecha o desfase temporal) es el margen de tiempo obligatorio que se deja entre la ventana de datos que se está analizando (el pasado) y el momento exacto en el que se espera que ocurra el evento/falla (el futuro).

IP Internet Protocol address, una dirección IP (dirección de protocolo de Internet) es una serie de números asignados a cada dispositivo conectado a una red informática o a Internet. Las direcciones IP identifican y diferencian los miles de millones de dispositivos en línea, incluidos los ordenadores y los teléfonos móviles, y ayudan a esos dispositivos a comunicarse entre sí.

LOGFILE También conocidos como archivos de registro, son archivos que registran secuencialmente todos los eventos o acciones que afectan a un proceso particular, como una aplicación o la actividad de una red informática. Estos archivos proporcionan una evidencia del comportamiento del sistema.

LSTM (Long Short-Term Memory / Memoria de Corto-Largo Plazo), tipo de red neuronal recurrente (RNN) diseñada para procesar, analizar y predecir datos secuenciales o series temporales. A diferencia de los modelos que analizan cada registro de forma aislada, la LSTM captura el orden y la evolución temporal de los eventos mediante compuertas de memoria, lo que le permite reconocer patrones precursores (por ejemplo, que la ocurrencia de un evento seguido de otro en un intervalo corto anticipe una falla posterior).

MÉTRICAS Son elementos cuantificables que surgen de una o varias mediciones obtenidas de fuentes con las que disponemos o podemos generar. Las métricas miden cualquier actividad o tarea, independientemente de los objetivos definidos.

MPLS o Multiprotocol Label Switching, es una tecnología de red avanzada que combina elementos de las redes conmutadas por circuitos y las redes conmutadas por paquetes. Su objetivo principal es facilitar la transmisión eficiente de datos a través de una red, minimizando la latencia y optimizando la velocidad. La red MPLS está ideada para agrupar diferentes tipos de datos transmitidos mediante la misma red y para mandar paquetes de información de forma rápida. Esta tecnología está sobre todo indicada para los servicios WAN y para soluciones de privacidad como una VPN.

MTTR (Mean Time To Repair / Tiempo Medio de Reparación), métrica operativa que mide el tiempo promedio necesario para restaurar un servicio o sistema tras una falla. Es un indicador clave de la eficiencia del equipo de soporte técnico y de la calidad de los procedimientos de respuesta a incidentes.

NOC (Network Operations Center / Centro de Operaciones de Red), lugar centralizado desde donde el equipo técnico de telecomunicaciones monitorea, gestiona y resuelve incidentes en la infraestructura de red de una organización en tiempo real.

PARSEO El término proviene del inglés *parsing* y se refiere al análisis de una secuencia de símbolos, ya sea texto, código o datos, con el objetivo de entender su estructura y organizarla de

manera comprensible para máquinas o humanos. En programación, este proceso permite que un programa interprete correctamente la información contenida en archivos de texto plano como los *syslog* de equipos de red.

PIPELINE (Línea de producción / Tubería de datos), es un conjunto de pasos secuenciales y automatizados que transforman los datos en bruto (como logs de texto plano en un archivo syslog) en información limpia, estructurada y lista para que un modelo de Inteligencia Artificial (como LSTM) pueda hacer predicciones, o puedan mostrar métricas en tiempo real.

PROTOTIPO Es una representación temprana y funcional de un producto o sistema, diseñada para proporcionar una vista preliminar y tangible de cómo se verá y se comportará el producto final. Se trata de un modelo inicial que encapsula las características clave, funcionalidades y aspectos visuales del producto, permitiendo a los diseñadores, desarrolladores y equipos de proyectos explorar conceptos, probar ideas y recopilar retroalimentación antes de avanzar hacia la producción completa.

REGEX (Expresión regular), es un conjunto de caracteres que actúa como un patrón para identificar coincidencias dentro de cadenas de texto. Su función principal es buscar, extraer, validar o reemplazar texto de forma avanzada, permitiendo automatizar tareas que serían complejas con búsquedas simples.

RMS Red Multi Servicios, una red multiservicio en telecomunicaciones es una infraestructura que puede transportar cualquier tipo de servicio sobre cualquier infraestructura; las redes multiservicio son la columna vertebral en que se soporta el mundo IP, y, por tanto, el mundo de las nuevas tecnologías.

SARIMA (*Seasonal ARIMA*), extensión de ARIMA que incorpora un segundo grupo de parámetros estacionales $(P, D, Q)_s$, los cuales replican los componentes autorregresivo, de integración y de media móvil sobre un período estacional fijo definido por s.

SFP Un (Small Form-Factor Pluggable), es un transceptor óptico intercambiable que permite la conexión de dispositivos de red a través de cables de fibra óptica o cobre. El SFP es un módulo que se utiliza en equipos de red como switches, routers y servidores. Su función principal es convertir señales eléctricas en señales ópticas y viceversa, permitiendo la transmisión de datos a través de diferentes tipos de cables. Esto proporciona una gran flexibilidad en la configuración de redes, ya que se pueden adaptar a diversas necesidades de velocidad y distancia simplemente cambiando el módulo SFP en lugar de reemplazar todo el hardware.

SLA (Service Level Agreement / Acuerdo de Nivel de Servicio), contrato entre un proveedor de servicios y un cliente que define el nivel de servicio esperado, incluyendo métricas de disponibilidad, tiempos de respuesta y penalizaciones económicas por incumplimiento.

SMOTE (Synthetic Minority Over-sampling Technique / Técnica de Sobremuestreo de Minoría Sintética), técnica que genera ejemplos sintéticos de la clase minoritaria interpolando entre

muestras existentes, utilizada para mitigar el problema del desbalance de clases en conjuntos de datos. En series temporales su aplicación requiere precaución para no introducir *data leakage*.

SYSLOG Protocolo estándar (RFC 5424) para el envío y almacenamiento de mensajes de registro de eventos generados por equipos de red, servidores y sistemas operativos. Los mensajes syslog contienen información de severidad, instalación, marca temporal y descripción del evento.

VPN (Virtual Private Network / Red Privada Virtual), es una conexión segura y cifrada entre dispositivos a través de una red pública. En el contexto de MPLS, las VPN permiten a las empresas extender su red privada sobre infraestructura compartida, garantizando el aislamiento del tráfico y la privacidad de las comunicaciones.

Bibliografía

Belzarena, P. and Aspirot, L. (2010). End-to-end quality of service seen by applications: A statistical learning approach. *Computer Networks*, 54(17):2993–3004.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, Hoboken, New Jersey, 5th edition.

Cano Salgado, D. and Zapata Castaño, W. (2026). Prototipo AIOps – pipeline de predicción de fallas en interfaces de red ópticas mediante LSTM. Repositorio del código fuente del proyecto de grado. Disponible en https://github.com/Kans29/Tesis_Prediccion_LogFiles. Consultado en junio de 2026.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, New York, NY, USA. Association for Computing Machinery.

Congreso de la República de Colombia (2012). Ley 1581 de 2012: Por la cual se dictan disposiciones generales para la protección de datos personales. Diario Oficial No. 48.587. https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981.

Du, M., Li, F., Zheng, G., and Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1285–1298. ACM.

Elastic NV (2021). The elastic stack: Elasticsearch, kibana, beats, and logstash. https://www.elastic.co/what-is/elk-stack.

Gartner (2017). Market guide for AIOps platforms. Research Note G00322184, Gartner Research.

Gerhards, R. (2009). The syslog protocol. RFC 5424, Internet Engineering Task Force.

Ghuli, P. (2018). Anomaly detection in log records. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(1):343–347.

Graylog, Inc. (2026). Graylog — centralized log management and siem. https://graylog.org/. Accedido: 2026-06-01.

Gómez, G., Fernández, A., Martínez Tagliafico, S., García González, G., Acuña, J., and Casas, P. (2020). Detección de anomalías en sistemas de telecomunicaciones mediante métodos de aprendizaje continuo. Proyecto de investigación, Universidad de la República, Facultad de Ingeniería (Udelar FI).

He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.

He, S., Zhu, J., He, P., and Lyu, M. R. (2016). Experience report: System log analysis for anomaly detection. In *Proceedings of the 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pages 207–218. IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hussein, S. A. and Répás, S. R. (2024). Anomaly detection in log files based on machine learning techniques. *Journal of Electrical Systems*, 20(3s):1299–1311.

IBM Corporation (2021). Ibm qradar: Siem security intelligence. https://www.ibm.com/security/security-intelligence/qradar.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.

LogRhythm Inc. (2021). Nextgen siem, log management, network monitoring, user & entity behavior (ueba). https://logrhythm.com/.

López-Avila, L., Acosta-Mendoza, N., and Gago-Alonso, A. (2019). Detección de anomalías basada en aprendizaje profundo: Revisión. *Revista Cubana de Ciencias Informáticas*, 13(3):107–123.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Seabold, S. and Perktold, J. (2010). statsmodels: Econometric and statistical modeling with Python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 92–96.

Splunk Inc. (2021). The data-to-everything platform. https://www.splunk.com/.

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., Chen, J., He, X., Yao, R., Lou, J.-G., Chintalapati, M., Shen, F., and Zhang, D. (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 27th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 807–817. ACM.