UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Análisis de la Calidad de Transmisión en Redes Ópticas mediante Técnicas de Big Data y Machine Learning

Informe de Proyecto de Grado presentado por

Andrés Conti, Marcelo Godoy

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Alberto Castro
Claudina Rattaro

Montevideo, 11 de junio de 2026

Agradecimientos

En primer lugar, queremos agradecer a nuestros tutores, Alberto y Claudina, por confiar en nosotros desde el inicio para la realización de este proyecto. Su guía constante, disponibilidad y acompañamiento a lo largo de todo el proceso fueron fundamentales para orientar el trabajo, superar obstáculos y mantener un rumbo claro en cada etapa.

Agradecemos también a la Facultad de Ingeniería por la formación brindada a lo largo de la carrera y por generar el marco académico y las oportunidades que hicieron posible llevar adelante este proyecto de grado, tanto en lo técnico como en lo humano.

Por último, queremos agradecer a quienes nos acompañaron a lo largo de este recorrido, en especial a nuestras familias, amigos y compañeros. Su apoyo y paciencia, especialmente en los momentos más exigentes, fueron fundamentales para poder completar esta etapa. Gracias.

Resumen

La calidad de transmisión (Quality of Transmission, QoT) es un factor central en la operación de redes ópticas: determina si una nueva conexión puede establecerse con un nivel de degradación aceptable y, por lo tanto, condiciona las decisiones de aprovisionamiento, uso del espectro y desempeño del sistema. Este trabajo diseña, implementa y evalúa un flujo integral *end-to-end* para estimar el QoT de forma *data-driven* sobre un dataset experimental provisto por el Fraunhofer Heinrich Hertz Institute (HHI), que ofrece dos representaciones complementarias del mismo escenario: (i) *Lightpath*, una vista tabular por conexión candidata, y (ii) *Network Status*, una descripción estructurada del estado global de la red al momento de la provisión.

La solución propuesta se apoya en un pipeline de ingeniería de datos por capas (*Bronze–Silver–Gold*) implementado en Python, que utiliza Apache Spark para el procesamiento distribuido y la materialización en formato Apache Parquet. La capa Bronze conserva los datos "crudos" (conversión desde NetCDF), Silver aplica limpieza y normalización, y Gold construye vistas específicas por tarea, incluyendo la definición de etiquetas, la selección de variables y particionados reproducibles de entrenamiento/validación/prueba. El entorno de ejecución se orquesta con Docker Compose (Apache Spark y Hadoop Distributed File System) para exploración y validación, y se complementa con ejecuciones en un clúster de alto desempeño bajo el planificador Simple Linux Utility for Resource Management (SLURM) para transformaciones de alto costo, especialmente relevantes en *Network Status*.

Sobre las vistas *Gold*, se define un protocolo experimental consistente para dos tareas supervisadas: (i) clasificación binaria de QoT y (ii) regresión de métricas continuas como *Optical Signal-to-Noise Ratio* (OSNR) y *Bit Error Rate* (BER). Se evalúan modelos tabulares (Random Forest y perceptrón multicapa, *Multi-Layer Perceptron*) sobre la biblioteca de aprendizaje automático de Apache Spark y redes neuronales en PyTorch sobre entradas derivadas de *Network Status*: una entrada secuencial para redes recurrentes (*Long Short-Term Memory* y *Gated Recurrent Unit*) y una entrada tensorial enlace–frecuencia para una red convolucional profunda (*Deep Convolutional Neural Network*). El ajuste de hiperparámetros se realiza con Optuna; en escenarios desbalanceados, se incorpora la selección de umbral en validación; y el análisis se complementa con interpretabilidad mediante importancias de variables y gradientes integrados.

En *Lightpath*, Random Forest logra un desempeño muy alto en clasificación

(área bajo la curva de precisión–recobrado 0.9987) y en regresión (coeficiente de determinación 0.9967 para OSNR), lo que sugiere que una representación tabular con variables agregadas concentra gran parte de la señal asociada a los efectos físicos relevantes. En *Network Status*, el desempeño es más moderado y sensible al protocolo: Random Forest alcanza un área bajo la curva de precisión–recobrado 0.917 y un perceptrón multicapa en PyTorch obtiene resultados comparables, mientras que modelos con estructura (recurrentes y convolucionales) permiten explorar interacciones más complejas a costa de mayor demanda de cómputo y puesta a punto. En conjunto, el trabajo entrega un pipeline reproducible, artefactos persistidos y evidencia empírica para orientar la elección de representaciones y técnicas de modelado en la estimación de QoT a escala.

Palabras clave: Redes ópticas elásticas, Calidad de transmisión (Quality of Transmission, QoT), Dataset experimental (Fraunhofer HHI), Representaciones *Lightpath* y *Network Status*, Ingeniería de datos a gran escala, Pipeline de datos por capas (Bronze–Silver–Gold), Apache Spark, Hadoop Distributed File System (HDFS), Aprendizaje automático supervisado, Optimización de hiperparámetros, Interpretabilidad de modelos, Reproducibilidad experimental

Glosario

A/D Add/Drop (agregar/quitar canales en un nodo).

API Application Programming Interface.

ASE Amplified Spontaneous Emission (ruido ASE de amplificadores).

AUC-PR Area Under the Precision–Recall Curve.

AUC-ROC Area Under the Receiver Operating Characteristic Curve.

BER Bit Error Rate.

BPSK Binary Phase Shift Keying.

Bronze Capa de datos crudos (ingesta/conversión, mínima transformación).

Bucket Partición/lote del dataset usado para procesamiento y ejecución por-bucket.

BVT Bandwidth Variable Transponder.

CNN Convolutional Neural Network.

CONUS Continental United States (topología de referencia del dataset).

CPU Central Processing Unit.

DCNN Deep Convolutional Neural Network (modelo convolucional profundo).

Docker Plataforma de contenerización (imágenes y contenedores).

EDA Exploratory Data Analysis.

EDFA Erbium-Doped Fiber Amplifier.

EON Elastic Optical Network (Flex-Grid).

ETL Extract–Transform–Load.

F Número de *frequency slots* por enlace (en la grilla espectral).

F1 Medida armónica entre precisión y *recall*.

FEC Forward Error Correction.

Fixed-Grid Asignación espectral en grilla fija (canales de ancho constante).

Flex-Grid Asignación espectral flexible por bloques contiguos de ranuras.

FN False Negative.

FP False Positive.

GRU Gated Recurrent Unit.

GOLD Capa de datos curada y orientada a consumo/experimentos (features/targets listos).

GPU Graphics Processing Unit.

HHI Fraunhofer Heinrich Hertz Institute.

HDFS Hadoop Distributed File System.

HPC High Performance Computing.

ID Identificador (campo de referencia; típicamente no informativo para el modelo).

IG Integrated Gradients (método de atribución de importancia).

I/O Input/Output (entrada/salida).

JOCN Journal of Optical Communications and Networking.

Jupyter Entorno de notebooks para computación interactiva.

KPI Key Performance Indicator.

L Número de enlaces unidireccionales de la topología.

Leakage Fuga de información: uso de variables demasiado cercanas/derivadas del objetivo.

Lightpath Conexión óptica establecida con ruta, espectro y configuración de transmisión.

Line rate Tasa de transmisión del servicio (*bitrate*).

LSTM Long Short-Term Memory.

LUT Lightpath Under Test (conexión bajo evaluación).

MAE Mean Absolute Error.

Medallion Arquitectura de capas Bronze–Silver–Gold para gobernanza y reproducibilidad.

ML Machine Learning.

MLP Multi-Layer Perceptron.

NPZ Formato comprimido de NumPy para persistir tensores/arreglos.

NSD Network State Dataset (representación de estado de red).

OOM Out Of Memory.

OSNR Optical Signal-to-Noise Ratio.

OFC Optical Fiber Communication Conference.

Parquet Formato columnar para almacenamiento eficiente en analítica.

Pipeline Flujo reproducible de pasos de datos y/o modelado.

PLATON Planning Tool for Optical Networks (herramienta de simulación de HHI).

QoT Quality of Transmission (viabilidad/calidad de transmisión).

Q-factor Métrica agregada de calidad, relacionada con separación estadística de niveles.

R^2 Coeficiente de determinación.

RF Random Forest.

RMSE Root Mean Squared Error.

RNN Recurrent Neural Network.

RSA Routing and Spectrum Assignment.

SBATCH Comando de envío de trabajos en Slurm.

Silver Capa de datos depurada/normalizada (limpieza, tipado, consistencia).

Slurm Gestor/planificador de recursos para HPC.

SNR Signal-to-Noise Ratio (y variantes efectivas).

Spark Motor distribuido para procesamiento/analítica (ETL y ML a escala).

Spark ML Biblioteca de aprendizaje automático sobre Apache Spark.

Split Partición de datos (train/val/test o train/test).

Snapshot Estado global de la red en un instante (métricas por enlace/slot).

tbh Largo de secuencia (número de pasos por muestra en la entrada secuencial).

TN True Negative.

TP True Positive.

Throughput Tasa de transferencia/servicio de datos (E/S o red).

WDM Wavelength Division Multiplexing.

Índice general

Capítulo 1

Introducción

En redes ópticas modernas, y en particular en redes ópticas elásticas (*Elastic Optical Networks*, EONs), la provisión de nuevas conexiones requiere decidir de forma rápida y confiable si un nuevo servicio puede establecerse sin comprometer el desempeño del sistema. Esta decisión suele expresarse en términos de *Calidad de Transmisión* (*Quality of Transmission*, QoT): determinar si un *lightpath* —esto es, una conexión óptica entre un origen y un destino, caracterizada por su ruta, la asignación de recursos espectrales y su configuración de transmisión— cumple un umbral operativo de calidad y con qué margen lo hace. En escenarios realistas, la QoT no depende únicamente de la ruta y de los parámetros propios del servicio, sino también del estado dinámico de la red, incluyendo su carga, ocupación y condiciones agregadas de los enlaces.

En este contexto, el aprendizaje automático aparece como una alternativa *data-driven* para aproximar estimaciones de QoT a partir de datos históricos o de simulación. Sin embargo, el desafío no es únicamente "entrenar un modelo": el volumen del material y la complejidad de las representaciones suelen hacer inviable un enfoque *ad hoc* o enteramente interactivo. Por eso, este trabajo se plantea como un flujo integral *end-to-end* que integra ingeniería de datos, análisis exploratorio y experimentación reproducible, desacoplando el costo de transformar datos del ciclo iterativo de modelado.

El estudio se apoya en un dataset sintético de referencia provisto por una colección pública de benchmarking para la estimación de QoT, que ofrece dos representaciones complementarias del mismo fenómeno: *Lightpath* (visión tabular por conexión) y *Network Status* (snapshot del estado global de la red al momento de la provisión). Esta dualidad permite tanto abordajes por conexión, en los que cada muestra se reduce a un vector de atributos, como abordajes condicionados por el contexto global, en los que la estructura interna del snapshot resulta relevante para el modelado. Sobre esta base, se construye un pipeline de datos por capas (*Bronze–Silver–Gold*), diseñado para preservar la trazabilidad, estandarizar las transformaciones y materializar vistas reutilizables orientadas al modelado.

En este capítulo se presenta la motivación y el contexto en el que se desarrolla

el proyecto, se definen los objetivos que orientan el trabajo, se resumen los principales resultados obtenidos y, por último, se describe la organización general del documento.

1.1. Motivación y contexto

En la última década, las redes ópticas se han consolidado como una infraestructura crítica para sostener el crecimiento de la demanda mundial de datos. La expansión de servicios digitales como *streaming* en alta definición, aplicaciones en la nube, videojuegos en línea y el Internet de las Cosas (IoT) incrementa la necesidad de ancho de banda y presiona a los esquemas tradicionales de provisión y administración, impulsando la búsqueda de técnicas que permitan utilizar recursos de manera más eficiente y adaptativa.

En este escenario, las redes ópticas elásticas surgen para atender la necesidad de una asignación más fina y flexible de recursos espectrales ante variaciones de la demanda (López y Velasco, 2016). No obstante, la gestión y optimización de estas redes—especialmente bajo condiciones dinámicas de tráfico y estado de red—introduce desafíos técnicos de gran envergadura, ya que decisiones como ruteo y asignación de espectro (RSA/RMSA) deben tomarse considerando simultáneamente restricciones físicas y operativas.

La estimación de QoT es un ejemplo representativo de ese tipo de decisiones: depende tanto de la configuración del *lightpath* como del contexto y del estado de la red, por lo que resulta natural explorar enfoques *data-driven* capaces de capturar relaciones complejas entre variables. Sin embargo, el reto práctico trasciende al modelo: para que la experimentación sea sostenible, es necesario disponer de datos, transformaciones y artefactos que puedan procesarse y reutilizarse a escala.

Con ese propósito, este trabajo se apoya en un dataset sintético de referencia, generado mediante simulación y provisto por Fraunhofer Heinrich Hertz Institute (HHI) dentro de una colección pública orientada a *benchmarking* para estimación de QoT (Fraunhofer Heinrich Hertz Institute (HHI), s.f.). El uso de esta colección permite discutir resultados en un marco comparable y reproducible y, al mismo tiempo, focalizarse en los aspectos de ingeniería requeridos para operar con representaciones y volúmenes no triviales.

En particular, la colección ofrece dos representaciones complementarias del mismo fenómeno: *Lightpath* y *Network Status*. Ambas presentan implicancias distintas en términos de granularidad, estructura y costo de procesamiento. Esta dualidad permite, por un lado, abordajes tabulares directos para cada conexión y, por otro, escenarios en los que el contexto global y la estructura interna del *snapshot* pasan a formar parte del problema. En consecuencia, transformaciones recurrentes como lectura, filtrado y agregaciones dejan de ser un detalle de implementación y se convierten en un factor determinante para la viabilidad del análisis exploratorio y del modelado.

Un antecedente directo en el trabajo sobre datasets de Fraunhofer HHI es el trabajo de Olmedo (2024) (Olmedo Guillama, 2024), donde se exploran técnicas

de aprendizaje automático para la estimación de QoT a partir de estas colecciones de referencia. Ese antecedente confirma la relevancia del problema de estimar QoT mediante enfoques supervisados sobre datasets de *benchmarking* y muestra el potencial de estas técnicas para capturar relaciones útiles entre variables de configuración, estado de red y calidad de transmisión. Al mismo tiempo, sugiere que cuando crecen el volumen de datos y la complejidad de las representaciones, un tratamiento basado únicamente en herramientas *in-memory* o en flujos *ad hoc* puede convertirse en un factor limitante. En este trabajo, esa observación se toma como punto de partida para abordar la ingeniería de datos y la reproducibilidad como componentes de primera clase.

En ese marco, el manejo eficiente de grandes volúmenes de datos resulta crucial. Herramientas de análisis *in-memory*—por ejemplo, bibliotecas ampliamente usadas en ciencia de datos como *pandas*—son efectivas para análisis acotados, pero no resultan apropiadas cuando el tamaño y la complejidad del dataset requieren procesamiento distribuido y transformaciones intensivas. Por esta razón, la adopción de técnicas de Big Data, como Apache Hadoop y Apache Spark, junto con metodologías de análisis exploratorio de datos (*Exploratory Data Analysis*, EDA), representa un camino necesario para sostener el procesamiento, la validación y la construcción de vistas reutilizables sobre datos a escala.

Además, la complejidad intrínseca de estos sistemas dificulta que la administración y la optimización puedan sostenerse mediante enfoques convencionales centrados en un único servidor de procesamiento: no solo por límites de cómputo y de memoria, sino también por la necesidad de tolerar fallos, absorber variaciones de carga y procesar datos de forma paralela. En contraposición, los sistemas distribuidos proveen un marco idóneo para escalar el procesamiento y el análisis, repartiendo el trabajo entre múltiples nodos y priorizando propiedades clave como particionado, replicación y *fault tolerance*, lo que habilita *pipelines* reproducibles y eficientes en volúmenes masivos (Kleppmann, 2017).

Finalmente, la aplicación de técnicas de aprendizaje automático (*Machine Learning*) se posiciona como un elemento clave en la búsqueda de soluciones robustas y eficientes para la estimación de QoT y la optimización de los recursos ópticos. En la práctica, modelos que anticipen degradaciones o estimen métricas de calidad pueden asistir en la toma de decisiones de aprovisionamiento y contribuir a mejorar el rendimiento global, siempre que su evaluación se realice bajo un protocolo consistente y sobre datos preparados de manera trazable.

En suma, la motivación del proyecto es hacer operativa la experimentación sobre un *benchmark* de QoT: transformar y validar datos a escala de forma reproducible, desacoplar el costo de ingeniería del ciclo iterativo de modelado y habilitar comparaciones controladas entre representaciones y familias de modelos. Bajo esta mirada, el aporte no se reduce al entrenamiento, sino a la construcción de la base metodológica y computacional que permite evaluar enfoques *data-driven* de manera consistente, trazable y repetible en un escenario de complejidad realista.

1.2. Objetivos

El objetivo principal de este trabajo es diseñar, implementar y evaluar un flujo integral *end-to-end* para la estimación *data-driven* de la **Calidad de Transmisión (QoT)** en redes ópticas elásticas (EONs), integrando ingeniería de datos, análisis exploratorio y experimentación reproducible con modelos de aprendizaje supervisado.

En particular, se plantean los siguientes objetivos específicos:

- **Caracterizar el problema y las fuentes de variabilidad** asociadas a la QoT, considerando tanto atributos del *lightpath* (configuración y trayecto) como información de *contexto/estado de red*, y analizar sus implicancias para el modelado.

- **Construir un pipeline de datos por capas (Bronze–Silver–Gold)** que convierta un dataset de gran volumen y estructura heterogénea en vistas aptas para aprendizaje automático, preservando trazabilidad respecto del origen y habilitando re-ejecución controlada bajo condiciones equivalentes.

- **Desacoplar la transformación del ciclo de modelado**, materializando artefactos reutilizables (capas *Silver/Gold*) que permitan entrenar y evaluar modelos sin recomputar transformaciones costosas en cada iteración.

- **Formular y abordar tareas supervisadas** sobre las representaciones disponibles, incluyendo: (i) clasificación de QoT (aceptación/rechazo bajo umbral operativo) y (ii) regresión de métricas físicas continuas (p. ej., OSNR/SNR y BER) como señal cuantitativa para diagnóstico.

- **Establecer líneas base reproducibles y comparables** entre representaciones y familias de modelos, y cuantificar el desempeño y la generalización mediante un protocolo consistente de particionado y métricas estándar, cuidando aspectos como el control de la fuga de información.

- **Analizar los resultados con foco práctico**, identificando escenarios y casos límite en los que el rendimiento se degrada, y discutiendo trade-offs entre desempeño predictivo, interpretabilidad y costo computacional/infraestructura requerida.

Como resultado, se espera entregar (i) un **pipeline reproducible** y trazable que consolide datasets listos para el modelado y (ii) una **evaluación experimental sistemática** que permita comprender la factibilidad y las limitaciones de distintos enfoques para estimar QoT, aportando evidencia útil para trabajos posteriores en operación, planificación o automatización asistida por datos en redes ópticas.

1.3. Resultados del proyecto

Como resultado de este trabajo, se obtiene un flujo integral que transforma el material original del dataset en artefactos consistentes y reutilizables para el modelado. En particular, el pipeline *Bronze–Silver–Gold* permite estandarizar transformaciones, preservar la trazabilidad respecto del origen y materializar vistas orientadas a la experimentación, evitando recomputar operaciones costosas en cada iteración; los scripts, configuraciones y artefactos asociados quedaron versionados en el repositorio del proyecto (Optical Networks ML, 2026). Sobre los conjuntos *Gold*, se evalúa la viabilidad del aprendizaje supervisado en dos tipos de tareas: (i) **clasificación de QoT**, para predecir si una conexión cumple un umbral operativo, y (ii) **regresión de métricas físicas** continuas (OSNR/SNR y BER), que aportan una señal cuantitativa para el diagnóstico, con un protocolo de evaluación consistente que permite comparar representaciones y familias de modelos.

Finalmente, el trabajo deja dos conclusiones que guían el resto del informe: por un lado, *Lightpath* se comporta como un baseline tabular sólido, interpretable y de bajo costo; por otro, *Network Status* configura un escenario más exigente donde la elección de representación, el desbalance de clases, el protocolo de particionado/evaluación y el control de fuga de información pasan a ser parte central del problema. En conjunto, estos hallazgos permiten discutir no solo el desempeño predictivo, sino también el costo y la estrategia de infraestructura necesaria para sostener un ciclo experimental realista.

1.4. Organización del documento

El informe se estructura de la siguiente manera:

- El **Capítulo 2** presenta el marco teórico: fundamentos de redes ópticas y QoT, y conceptos de Big Data y Machine Learning relevantes para el enfoque propuesto.
- El **Capítulo 3** describe las herramientas y la infraestructura computacional utilizada, junto con los criterios que guían su configuración y uso.
- El **Capítulo 4** desarrolla la ingeniería de datos: la estructura del dataset, el pipeline *Bronze–Silver–Gold* y el análisis exploratorio para ambas representaciones.
- El **Capítulo 5** presenta el diseño y la metodología experimental, incluyendo el protocolo de entrenamiento y evaluación.
- El **Capítulo 6** presenta los resultados de la experimentación y su análisis, comparando representaciones y modelos bajo condiciones controladas.
- El **Capítulo 7** resume los principales aspectos de la ingeniería de software y de la reproducibilidad del proyecto.

- El **Capítulo 8** sintetiza conclusiones globales, limitaciones y líneas de trabajo a futuro.
- En los **Apéndices** se incluyen detalles operativos complementarios: el Anexo B resume los scripts del pipeline para la representación *Lightpath*, el Anexo C hace lo propio para la representación *Network Status*, y el Anexo D documenta el procedimiento de conexión a la máquina virtual de experimentación.

Capítulo 2

Marco Teórico

Este capítulo establece el marco conceptual necesario para contextualizar el problema de estimación de QoT abordado en el resto del informe. La Figura 2.1 resume el recorrido propuesto: en primer lugar, se introducen los fundamentos del transporte óptico —propagación en fibra, multiplexación WDM/DWDM y noción de *lightpath*— y la evolución hacia redes ópticas elásticas (Flex-Grid/EON), donde la provisión de una conexión requiere tomar decisiones de ruteo, asignación espectral y configuración de la transmisión (p. ej., formato de modulación). Sobre esa base, se define la Calidad de Transmisión (QoT) como criterio de factibilidad y se presentan las métricas más utilizadas para cuantificarla (OSNR, BER y Q-factor), conectando las degradaciones físicas con indicadores observables. Luego, se sintetizan los conceptos de Big Data y Machine Learning relevantes para un enfoque *data-driven*, considerando el tipo de variables disponibles, la formulación de las tareas de aprendizaje y los criterios generales de evaluación. Finalmente, se revisa el estado del arte sobre el uso de aprendizaje automático en redes ópticas, destacando tendencias, aplicaciones frecuentes y desafíos abiertos que motivan este trabajo.

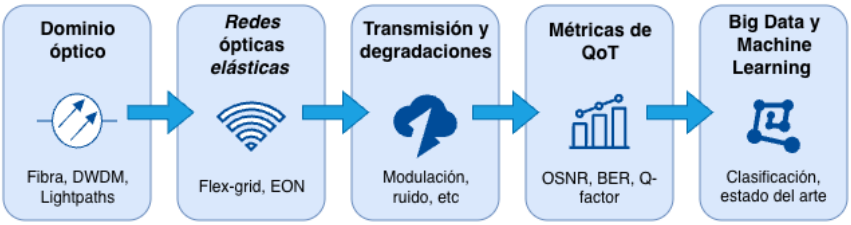


Figura 2.1: Mapa del Marco Teórico. La figura resume la progresión de conceptos que estructura el capítulo: (i) fundamentos del dominio óptico, (ii) redes ópticas elásticas y decisiones de provisión, (iii) transmisión y degradaciones que impactan la señal, (iv) métricas para cuantificar la QoT, y (v) herramientas de Big Data y Machine Learning que se emplean para modelar y predecir QoT.

2.1. Redes ópticas

La mayor parte de los conceptos introductorios presentados en esta sección se basan en el libro *Elastic Optical Networks: Architectures, Technologies, and Control* (López y Velasco, eds.) (López y Velasco, 2016).

La fibra óptica se ha convertido en el medio de transmisión preferido para redes modernas de alta capacidad, especialmente en el transporte troncal. Su adopción se explica por ventajas prácticas frente al cobre: permite cubrir distancias mayores con menor atenuación, ofrece un ancho de banda muy superior y, al no conducir electricidad, es altamente inmune al ruido electromagnético. En términos operativos, esto habilita enlaces de alta velocidad y de larga distancia con una infraestructura más eficiente para el crecimiento sostenido del tráfico (por ejemplo, entre centrales, nodos metropolitanos y centros de datos).

Aun así, transportar información "como luz" no significa que la señal se mantenga perfecta. La señal óptica se degrada gradualmente a medida que recorre la fibra y atraviesa equipos de la red. Por ese motivo, además de entender *qué* es una red óptica, es importante entender *cómo* se organiza el transporte: qué se considera un canal, cómo se multiplica la capacidad y qué significa establecer una conexión extremo a extremo. Estos conceptos serán fundamentales más adelante, cuando se introduzca la noción de QoT y las métricas que la representan.

2.1.1. La fibra óptica y la propagación de la señal

En una red óptica, la información se transmite mediante una señal luminosa que se propaga a través de la fibra óptica. A nivel conceptual, puede pensarse como un "haz" guiado en un medio muy transparente. En la práctica, el canal óptico no es ideal, existen pérdidas, y además surgen efectos que afectan la forma de la señal a lo largo del camino. Por ello, a medida que aumenta la distancia, resulta necesario incorporar dispositivos ópticos intermedios (por ejemplo, amplificadores) para mantener la señal dentro de rangos operativos.

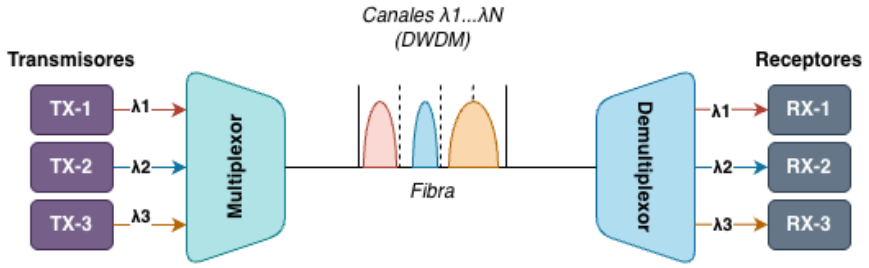
Desde la perspectiva de esta tesis, el punto clave es que el desempeño final de una conexión depende del *camino físico* recorrido y de los *elementos atravesados*. Esta dependencia es la base del concepto de *calidad de transmisión* (QoT), que se formaliza en secciones posteriores.

Multiplexación por longitud de onda: WDM/DWDM

Una sola fibra puede transportar múltiples señales simultáneamente mediante multiplexación por división de la longitud de onda (Wavelength Division Multiplexing, WDM). La idea es simple: si cada señal se transmite en una longitud de onda (o frecuencia) distinta, pueden coexistir en la misma fibra sin interferirse directamente. Cuando el número de canales es grande y el espaciado entre ellos es pequeño, se habla de *Dense WDM* (DWDM). Como se ilustra en la Figura 2.2, en un enlace punto a punto DWDM múltiples transmisores generan canales ópticos en diferentes longitudes de onda ($\lambda_1 \dots \lambda_N$), que se combinan

mediante un multiplexor para propagarse conjuntamente por la fibra, y luego se separan en el extremo receptor mediante un demultiplexor.

DWDM es una de las razones principales por las que las redes ópticas escalan tan bien: para aumentar la capacidad, en muchos casos alcanza con *agregar canales* y reutilizar la misma infraestructura física (fibra, amplificadores y filtros). Esa misma escalabilidad, sin embargo, conlleva desafíos operativos, ya que cada canal debe generarse en el origen, mantener condiciones adecuadas durante el transporte y separarse correctamente en el destino.



Elaboración propia, adaptado de (López y Velasco, 2016).

Figura 2.2: Esquema de enlace punto a punto DWDM: múltiples transmisores ($\lambda_1 \dots \lambda_N$) se multiplexan, se propagan por la fibra y se demultiplexan en el receptor.

De canales a conexiones: qué es un *lightpath*

En la práctica, el objetivo de la red no es "transportar longitudes de onda", sino establecer **conexiones** entre nodos de origen y destino. Una conexión óptica extremo a extremo suele denominarse *lightpath*. Un *lightpath* puede atravesar múltiples enlaces y nodos intermedios, manteniendo (idealmente) el transporte en el dominio óptico para minimizar conversiones al dominio eléctrico.

De manera conceptual, un *lightpath* se puede describir por:

- **Ruta:** la secuencia de enlaces (fibras) y nodos que conectan origen y destino.
- **Recursos ópticos asignados:** el canal o la banda espectral utilizada para transportar la señal.
- **Equipamiento terminal:** transponders en los extremos, responsables de generar y detectar la señal.
- **Configuración de transmisión:** parámetros con los que se transmite la señal (p. ej., modulación, esquema de **FEC** y tasa/*baud rate* o *line rate*), que determinan el compromiso entre eficiencia espectral, alcance y requerimientos de QoT.

Este concepto es especialmente importante para este trabajo: la QoT no se evalúa "en abstracto", sino para un *lightpath específico*, con una ruta y una configuración de transmisión específicas. En otras palabras: el mismo servicio, con una ruta o una configuración distinta, puede presentar una QoT diferente.

Nodos reconfigurables: ROADM en términos funcionales

Para que la red sea flexible en operación, no basta con multiplexar canales, también es necesario *conmutarlos* y *redirigirlos* sin intervención manual. Aquí aparece el concepto de *ROADM* (*Reconfigurable Optical Add-Drop Multiplexer*), un tipo de nodo que permite:

- **Agregar** canales (add) a la fibra desde transponders locales,
- **Extraer** canales (drop) hacia transponders locales,
- **Dejar pasar** canales que atraviesan el nodo hacia otra dirección (*express*).

La Figura 2.3 resume estas funciones a nivel conceptual, distinguiendo el tráfico que atraviesa el nodo (*express*) del que se extrae hacia equipos locales (*drop*) y del que se inserta desde dichos equipos (*add*).

Esta capacidad habilita la reconfiguración remota, la optimización del uso de los recursos y la recuperación ante fallas. Además, el hecho de que la señal atraviese múltiples nodos tiene implicancias directas en el QoT (por ejemplo, debido a efectos de filtrado y acumulación de degradaciones), lo que conecta de forma natural con el análisis posterior del desempeño de la transmisión.

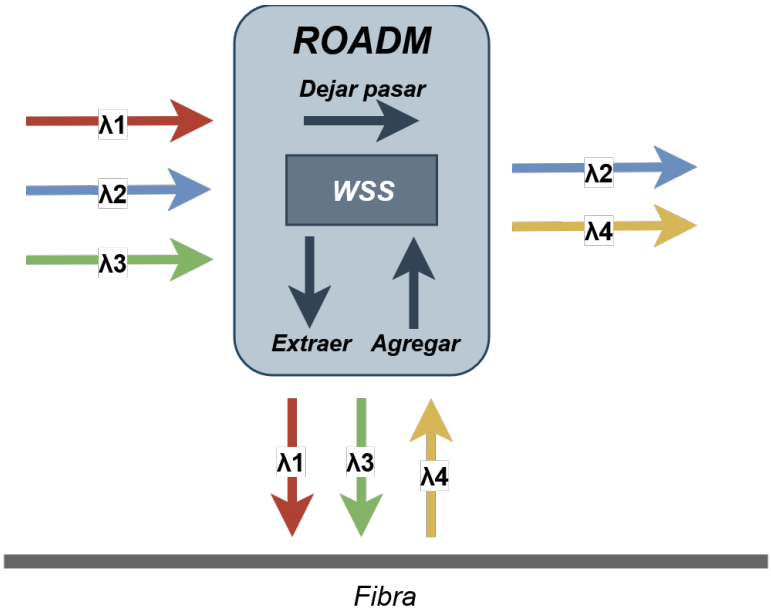
Hasta aquí, la idea clave es que una red óptica puede transportar muchos canales en paralelo (DWDM) y establecer conexiones extremo a extremo (*lightpaths*) que atraviesan nodos reconfigurables (ROADMs). En las siguientes subsecciones se introduce cómo evoluciona el uso del espectro (Fixed-Grid → Flex-Grid/EON) y por qué la configuración de transmisión (por ejemplo, la modulación) y el estado de la red se vuelven determinantes para la QoT.

2.1.2. Planificación y uso del espectro: de Fixed-Grid a Flex-Grid y redes elásticas

En la sección anterior se introdujo la idea de transmitir múltiples canales en paralelo (DWDM) y establecer conexiones extremo a extremo (*lightpaths*). En ese contexto, surge una pregunta natural: *si el espectro óptico es el "recurso" que compartimos en una fibra, ¿cómo se organiza y se asigna ese espectro para crear canales?* La respuesta a esa pregunta ha ido evolucionando con el tiempo, impulsada por el crecimiento del tráfico y la necesidad de mayor flexibilidad operativa.

A grandes rasgos, pueden distinguirse tres etapas conceptuales en la forma de usar el espectro:

- **Fixed-Grid:** canales con ancho y separación fijos.



Elaboración propia, adaptado de (López y Velasco, 2016).

Figura 2.3: Funciones principales de un ROADM: *express* (canales que atraviesan el nodo), *drop* (canales extraídos hacia equipos locales) y *add* (canales agregados desde equipos locales) mediante conmutación selectiva en longitud de onda.

- **Flex-Grid:** el espectro se divide en ranuras más pequeñas (*slices*) que pueden combinarse.
- **Redes ópticas elásticas (EON):** además de flex-grid, se habilita adaptar la transmisión (por ejemplo, modulación y parámetros del transponder) para ajustar los recursos al servicio.

Esta evolución no es un detalle meramente normativo, sino que impacta directamente en el desempeño y la operación de la red. Por un lado, permite aprovechar mejor el espectro; por otro, incrementa el número de configuraciones posibles, lo que vuelve más relevante estimar la QoT de forma rápida y confiable.

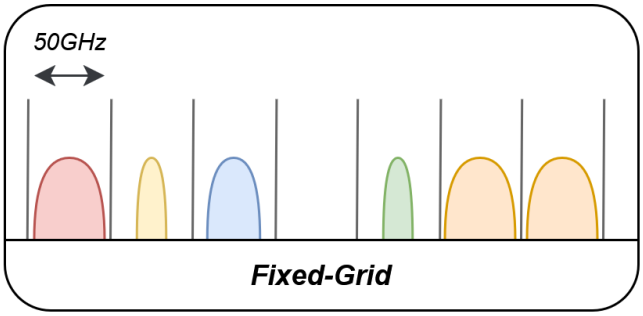
Fixed-Grid: simplicidad a costa de eficiencia

En el esquema **Fixed-Grid**, el espectro se particiona en canales de ancho fijo y centrados en frecuencias predefinidas. En términos prácticos, cuando una conexión necesita transportarse, se le asigna uno de esos canales completos, aunque la demanda real no ocupe toda la capacidad potencial del canal. La Figura 2.4 ilustra este principio: el espectro se divide en *slots* o canales de

ancho constante (por ejemplo, 50 GHz) y cada servicio ocupa un canal completo, independientemente de si utiliza efectivamente todo el ancho disponible.

La ventaja principal de Fixed-Grid es la **simplicidad**, ya que el filtrado y la planificación son más directos, y la interoperabilidad entre equipos se facilita. Sin embargo, con la aparición de servicios heterogéneos (distintas tasas y formatos), comienzan a notarse limitaciones:

- **Desperdicio espectral:** una conexión puede "pagar" un canal completo aunque no lo necesite.
- **Rigidez para crecer:** si se desea aumentar capacidad, muchas veces no hay manera de "ensanchar" un canal; hay que reasignar y reorganizar.



Elaboración propia, adaptado de (López y Velasco, 2016).

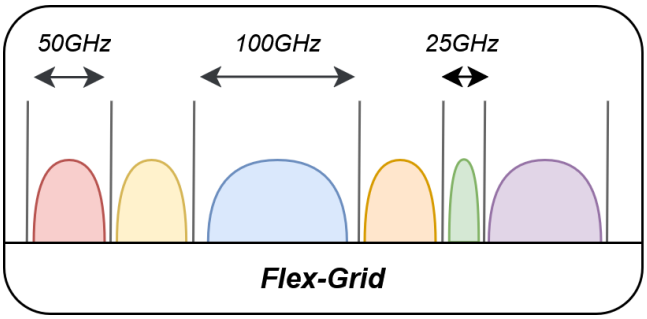
Figura 2.4: Ejemplo de asignación en grilla fija (Fixed-Grid): el espectro se divide en canales de ancho constante (p. ej., 50 GHz) y cada servicio ocupa un canal completo, aun cuando su demanda no requiera todo el ancho disponible, lo que genera ineficiencia espectral.

Flex-Grid: el espectro como bloques combinables

El esquema **Flex-Grid** surge para mejorar la eficiencia en el uso del espectro. En lugar de asignar canales con anchos fijos, el espectro se particiona en ranuras más pequeñas (*slices*) y cada conexión recibe un **bloque contiguo de slices** cuyo tamaño se ajusta a los requerimientos del servicio.

Como se ilustra en la Figura 2.5, esto permite una asignación más flexible: conexiones de menor tasa pueden ocupar menos espectro, mientras que servicios de alta tasa pueden combinar más *slices* para formar un canal más ancho. Una forma habitual de visualizarlo es pensar el espectro como una escala discretizada en bloques pequeños, donde cada conexión reserva un tramo continuo.

Bandas de guarda (guard-bands). En la práctica, entre conexiones vecinas suele reservarse una pequeña separación espectral llamada *guard-band*. Su función es reducir las interferencias y facilitar el filtrado en los nodos. Aunque



Elaboración propia, adaptado de (López y Velasco, 2016).

Figura 2.5: Asignación en Flex-Grid: el espectro se divide en *slices* (ranuras) de menor granularidad y cada servicio recibe un bloque contiguo, ajustado a su demanda, lo que mejora la eficiencia en el uso del espectro.

parezca un detalle, las guard-bands influyen en la eficiencia total y también en la QoT, porque cambian el "espacio útil" que queda disponible para nuevas conexiones.

Redes ópticas elásticas (EON): elasticidad de espectro + elasticidad de transmisión

Las redes ópticas elásticas (EON) toman la idea de flex-grid como base, pero la llevan un paso más allá: no solo se ajusta el ancho espectral, sino que también se ajusta la configuración de transmisión para utilizar los recursos de forma eficiente sin comprometer la viabilidad en QoT (López y Velasco, 2016). En otras palabras, la red busca asignar "solo lo necesario" en espectro y transmitir "de la forma adecuada" en términos de parámetros de transmisión, de modo que la conexión sea eficiente pero también aceptable desde el punto de vista de QoT.

Aquí aparece de forma natural el rol del formato de modulación y de otros parámetros (por ejemplo, tasa de símbolos o FEC), ya que determinan el compromiso entre la eficiencia espectral y la robustez de la transmisión. Al cambiar la modulación, varía tanto la cantidad de información que puede transportarse por unidad de espectro como la sensibilidad de la señal al ruido y a las distorsiones. Por eso, dos conexiones con el mismo bit-rate pueden requerir distinto ancho espectral según la modulación elegida, y, a la vez, esa elección puede ser viable o no en función de la ruta y del estado de la red.

Una consecuencia importante: fragmentación del espectro

Una consecuencia práctica de asignar bloques variables (y liberar conexiones en distintos momentos) es que el espectro puede quedar fragmentado. Es decir, puede haber suficiente espectro libre en total, pero distribuido en "huecos" pequeños que no alcanzan para formar un bloque contiguo del tamaño

necesario para una nueva conexión. Tras la liberación o reacomodo de demandas, el espectro disponible puede quedar particionado, y una nueva solicitud puede bloquearse aun cuando el espectro libre total sea suficiente.

La fragmentación es relevante por dos motivos:

- **Impacta la operación:** aumenta la probabilidad de bloqueo de nuevas solicitudes, incluso cuando la capacidad total disponible sería, en principio, suficiente.

- **Afecta decisiones y QoT:** la red puede verse forzada a usar porciones de espectro menos convenientes (por ejemplo, más cercanas a otros canales o con más filtros acumulados), lo que puede incidir en el desempeño.

La transición Fixed-Grid → Flex-Grid/EON mejora la eficiencia y la flexibilidad, pero también amplía el espacio de decisiones. Para establecer un *lightpath* es necesario seleccionar la ruta, el bloque espectral y, en enfoques elásticos, la configuración de transmisión. Dado que la factibilidad depende de degradaciones físicas y del estado dinámico de la red, estimar QoT se vuelve un componente central. A continuación se profundiza en los parámetros de transmisión (Sección 2.1.3) y luego se introducen las métricas de QoT; más adelante se discute cómo Big Data y Machine Learning permiten modelar esta relación a partir de datos históricos y de monitoreo.

2.1.3. Modulación y transmisión coherente

En las secciones anteriores se describió cómo la red transporta múltiples canales en paralelo (DWDM) y cómo, con Flex-Grid/EON, es posible ajustar el ancho espectral asignado a cada conexión. El siguiente paso para entender la QoT es mirar el extremo de transmisión y recepción: *cómo se convierte una secuencia de bits en una señal óptica* y qué decisiones de configuración influyen en su robustez frente a degradaciones.

De forma general, el transponder toma información digital (bits) y la representa como una señal óptica modulada. El formato de modulación determina cuántos bits se transmiten por símbolo y, por lo tanto, influye en dos aspectos centrales:

- **Eficiencia espectral:** cuánta capacidad se logra por unidad de ancho de banda.

- **Robustez:** qué tan sensible es la transmisión al ruido y a distorsiones acumuladas en la red.

Esta relación es un puente directo hacia QoT: a igualdad de ruta y condiciones de red, una modulación más exigente puede fallar donde una modulación más robusta funciona.

Concepto de modulación y constelaciones

En comunicaciones digitales, modular consiste en mapear grupos de bits a símbolos de una constelación. Por ejemplo, en el formato de modulación QPSK se transmiten 2 bits por símbolo, mientras que en 16-QAM se transmiten 4 bits por símbolo. En términos cualitativos, 16-QAM permite mayor capacidad para un mismo ritmo de símbolos, pero requiere que el receptor distinga puntos de la constelación más cercanos entre sí, lo que aumenta la sensibilidad al ruido. La Figura 2.6 ilustra esta diferencia mediante ejemplos de constelaciones y el efecto del ruido sobre las muestras recibidas (*Phase-shift keying*, s.f.; *Quadrature amplitude modulation*, s.f.).

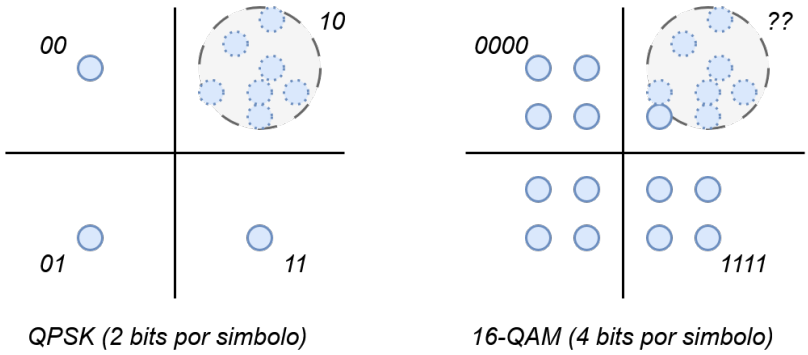


Figura 2.6: Ejemplos de constelaciones: QPSK (2 bits/símbolo) y 16-QAM (4 bits/símbolo). El ruido dispersa las muestras recibidas; en 16-QAM, la menor separación entre símbolos aumenta la probabilidad de confusión en la decisión.

Implicancia para la QoT. Una forma intuitiva de conectar esto con QoT es pensar que, a medida que la señal se degrada, la "nube" de símbolos alrededor de cada punto se dispersa. Cuando la dispersión crece, aumenta la probabilidad de decidir el símbolo incorrecto, lo que se refleja en un aumento de la tasa de error. Por ello, modulaciones de mayor orden suelen requerir mejores condiciones de transmisión para mantener un desempeño aceptable.

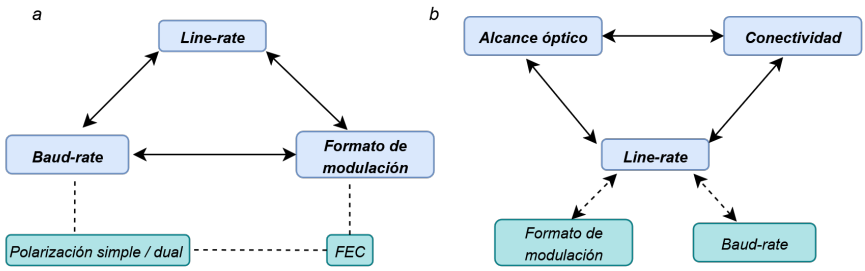
Tasa de símbolo, line-rate y FEC: parámetros que se combinan

Además del formato de modulación, existen otros parámetros que forman parte de la configuración de transmisión. A alto nivel, los más relevantes son:

- **Baud-rate (tasa de símbolos):** cantidad de símbolos por segundo.
- **Line-rate:** tasa bruta resultante (antes de corrección de errores), determinada por (bits por símbolo) × (baud-rate) × (número de polarizaciones, si aplica).

- **FEC (Forward Error Correction):** codificación de corrección de errores que introduce redundancia; reduce la tasa útil, pero aumenta tolerancia a errores.

En redes modernas, estas variables se ajustan en conjunto para lograr un equilibrio entre capacidad, ocupación espectral y robustez. En particular, para un mismo servicio objetivo, distintas combinaciones de (modulación, baud-rate, FEC) pueden resultar en diferentes requerimientos de ancho espectral y diferentes márgenes de QoT. La Figura 2.7 resume esta relación: a nivel del subsistema de transmisión, los parámetros determinan la tasa de línea y el esquema de protección (p. ej., FEC), mientras que a nivel de sistema dichas elecciones condicionan el alcance óptico y, en consecuencia, la conectividad alcanzable.



Elaboración propia, adaptado de (López y Velasco, 2016).

Figura 2.7: Relación conceptual entre parámetros de transmisión en redes ópticas: (a) a nivel de subsistema, la tasa de línea depende del baud-rate y del formato de modulación, e incorpora decisiones como FEC y polarización; (b) a nivel de sistema, estas elecciones impactan el alcance óptico y, por ende, la conectividad de la red.

Transmisión directa vs. transmisión coherente

Existen distintas arquitecturas para transmitir y detectar señales ópticas. En términos generales, se distinguen dos enfoques:

Transmisión directa. En un esquema de detección directa, el receptor mide la potencia óptica (intensidad) y recupera la información a partir de esa medida. Este enfoque puede ser adecuado en ciertas condiciones, pero ofrece menor flexibilidad para compensar degradaciones complejas y para explotar plenamente modulaciones avanzadas.

Transmisión coherente. En sistemas coherentes, el receptor mezcla la señal recibida con un oscilador local y puede recuperar información de amplitud y fase. Esto habilita el uso de modulaciones más densas (por ejemplo, QAM), y permite aplicar procesamiento digital (DSP) para compensar efectos del canal,

como dispersión y otras distorsiones. Por estas razones, la transmisión coherente es una tecnología clave en redes modernas de alta capacidad y es especialmente relevante en el contexto de Flex-Grid/EON.

Trade-off central: capacidad, espectro y alcance

Al combinar flexibilidad espectral con transmisión coherente, aparece un trade-off central para la operación:

- Para **maximizar capacidad** y **reducir ocupación de espectro**, conviene utilizar modulaciones más eficientes (mayor orden).
- Para **maximizar alcance** y tolerar más degradaciones, conviene utilizar modulaciones más robustas y/o FEC más fuerte.

Este equilibrio depende de la ruta (distancia y número de elementos atravesados), del entorno espectral (canales vecinos, guard-bands) y del estado operativo de la red. En consecuencia, dos *lightpaths* con igual tasa objetivo pueden requerir configuraciones distintas para cumplir QoT. Esta observación es importante para este trabajo, ya que el desempeño final no depende de una sola variable, sino de la interacción entre topología, espectro, configuración de transmisión y condiciones de la red.

La Figura 2.8 resume cualitativamente esta relación entre eficiencia espectral y alcance, donde al aumentar el orden de modulación se mejora la eficiencia, pero típicamente disminuye el margen frente a degradaciones y, por tanto, se reduce la distancia alcanzable en comparación con modulaciones más robustas.

La modulación, el baud-rate y la FEC influyen en cómo "se ve" la señal en el receptor y en la tolerancia al ruido y distorsiones. Por ello, es natural introducir métricas que cuantifiquen el margen de transmisión y la probabilidad de error, y que permitan decidir si una conexión es viable o no. En la próxima sección se formaliza el concepto de QoT y se presentan las métricas relevantes, estableciendo el marco que luego será modelado mediante técnicas de Big Data y Machine Learning.

2.1.4. Calidad de Transmisión (QoT): concepto y métricas relevantes

Hasta aquí se presentó el camino conceptual que conecta el transporte óptico con decisiones de configuración (espectro y modulación). El punto clave es que una conexión no se considera válida únicamente porque exista espectro disponible indicando la posibilidad de establecer el *lightpath*, ya que además debe cumplir una **calidad de transmisión** mínima para que el receptor pueda recuperar la información con una tasa de error aceptable.

En redes ópticas reales, la calidad de transmisión depende de múltiples factores y su evaluación suele involucrar tanto mediciones de monitoreo como modelos de ingeniería. En términos generales, puede interpretarse como un criterio de factibilidad: dada una ruta y una configuración de transmisión (por ejemplo,

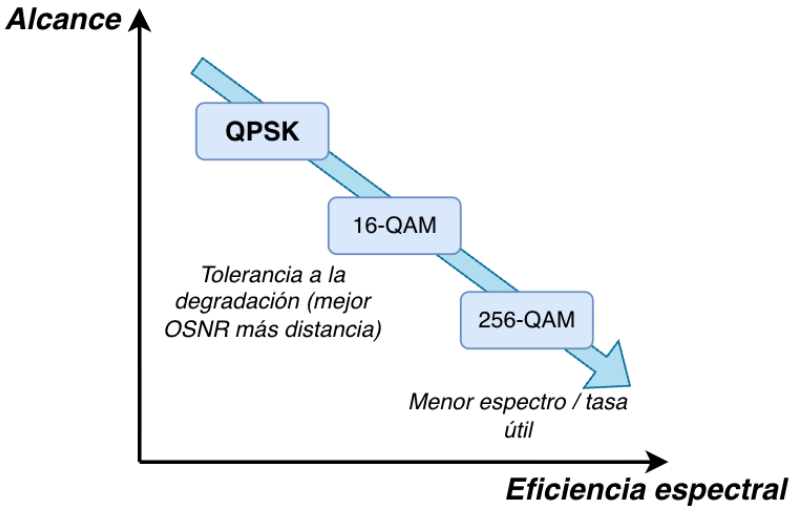


Figura 2.8: Trade-off cualitativo entre alcance y eficiencia espectral: modulaciones de mayor orden (p. ej., 16-QAM, 256-QAM) ofrecen mayor eficiencia espectral, pero suelen requerir condiciones de transmisión más favorables y, por tanto, alcanzan menores distancias que modulaciones más robustas como QPSK.

ancho de canal y formato de modulación), determinar si la conexión alcanzará el desempeño requerido. Esta idea es central en el dominio y es precisamente el tipo de relación que este trabajo busca analizar mediante técnicas de Big Data y Machine Learning.

Por qué la QoT es un problema no trivial

La señal óptica se degrada a lo largo de la ruta por varios mecanismos que pueden actuar simultáneamente. Sin entrar en un desarrollo físico exhaustivo, es útil distinguir dos familias de efectos:

- **Efectos que se acumulan con la distancia y la amplificación:** la atenuación de la señal a lo largo de la fibra requiere el uso de amplificadores ópticos, cuya operación introduce inevitablemente ruido. En particular, los amplificadores ópticos generan *ruido ASE* (*Amplified Spontaneous Emission*), originado por la emisión espontánea del medio amplificador que, al ser amplificada junto con la señal, se manifiesta como un piso de ruido adicional y reduce el margen disponible en el receptor.
- **Efectos asociados a la propagación y al entorno espectral:** dispersión, no linealidades y degradaciones por filtrado al atravesar nodos (por ejemplo, filtrado concatenado). Además, la proximidad a otros canales y el uso de guard-bands influyen en la interferencia y en la robustez.

En redes con Flex-Grid/EON, donde el espectro se asigna en bloques variables y una misma demanda puede admitirse bajo varias configuraciones, la QoT no es un parámetro fijo, sino que depende del *estado de la red* y de las decisiones de provisión (ruta, espectro y configuración de transmisión).

2.1.5. Métricas típicas para cuantificar la QoT

En operación y planificación se utilizan métricas que permiten cuantificar la calidad de la señal y decidir si una conexión es viable. Las métricas más comunes incluyen:

OSNR (Optical Signal-to-Noise Ratio). Relaciona la potencia de la señal con la potencia de ruido óptico (habitualmente asociado a ruido ASE introducido por amplificadores). En términos prácticos, un OSNR bajo indica que la señal llega al receptor con poco margen frente al ruido, aumentando la probabilidad de error. Es una métrica ampliamente utilizada porque es medible y captura un factor dominante en muchos escenarios de transmisión.

SNR (Signal-to-Noise Ratio) y variantes efectivas. En sistemas coherentes, además del OSNR óptico, puede considerarse un SNR efectivo relacionado con la señal ya procesada en el receptor y con el impacto de distorsiones y ruido en banda. Dependiendo del equipamiento y del esquema de monitoreo, pueden reportarse métricas equivalentes o derivadas.

BER (Bit Error Rate). Es la fracción de bits recibidos incorrectamente. Puede definirse antes o después de la corrección de errores (pre-FEC / post-FEC). Un criterio operacional común es verificar que la BER se mantenga por debajo de un umbral (o que la FEC pueda corregir los errores con un margen adecuado).

Q-factor. Es una métrica relacionada con la separación estadística entre niveles de señal y se usa como indicador agregado de desempeño. En la práctica, muchas plataformas reportan Q-factor o métricas equivalentes como proxy de calidad, y se establecen umbrales para operación.

Potencia óptica y pérdidas. Las mediciones de potencia (por ejemplo, potencia transmitida/recibida, pérdidas por tramo, etc.) complementan el análisis, ya que la potencia fuera de rango puede indicar degradaciones, fallas o configuraciones no óptimas.

La Figura 2.9 resume estas métricas y su rol como indicadores complementarios de calidad de transmisión, desde relaciones señal-ruido (OSNR/SNR) hasta medidas directas o derivadas del desempeño de error (BER y Q-factor).

QoT como criterio de aceptación: umbrales y márgenes

En operación, la QoT suele emplearse como un criterio de aceptación basado en umbrales: una conexión se acepta si una métrica (o un conjunto de métricas) cumple un requisito mínimo definido por ingeniería (López y Velasco, 2016). Este tipo de regla es habitual tanto en planificación como en control operacional, ya que permite traducir el desempeño físico de transmisión a una decisión binaria de provisión.

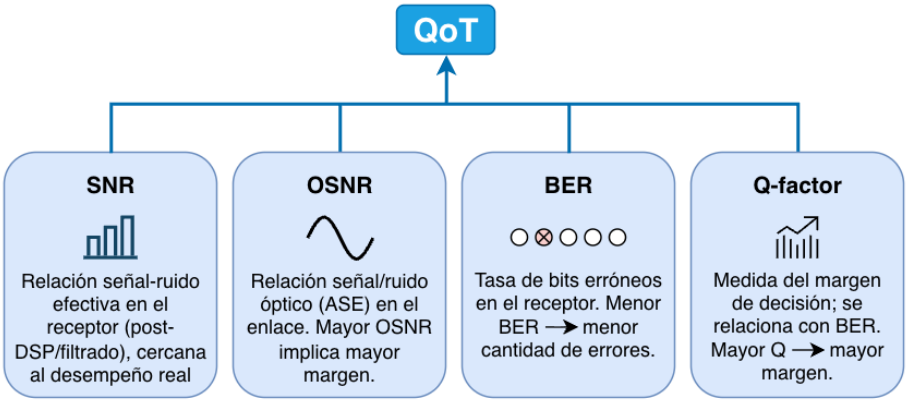


Figura 2.9: Métricas utilizadas para evaluar la QoT: SNR (efectivo), OSNR, BER y Q-factor.

Por ejemplo, se puede aceptar un *lightpath* si la relación señal-ruido disponible supera un valor umbral asociado a la modulación y al esquema de corrección de errores, o si la tasa de error esperada se mantiene dentro de lo corregible por el receptor. Estos umbrales dependen del tipo de *transponder*, del formato de modulación y de políticas de ingeniería (márgenes de diseño). Además, es usual considerar márgenes para cubrir incertidumbres, variaciones temporales y envejecimiento de la infraestructura.

La Figura 2.10 ilustra esta idea: una métrica de calidad se compara contra un umbral para aceptar o rechazar una conexión, lo cual conecta naturalmente con formulaciones de aprendizaje automático orientadas a clasificación (aprobado/rechazado) o a predicción directa de la métrica.

Conexión con el enfoque de esta tesis: de métricas a un problema de modelado

Desde la perspectiva de análisis de datos, la QoT puede formularse como un problema de modelado de distintas maneras, según el tipo de información disponible:

- **Clasificación:** predecir si una conexión cumplirá QoT (apta/no apta) de acuerdo con un umbral sobre una métrica (por ejemplo, BER o Q-factor).
- **Regresión:** estimar directamente el valor de una métrica (por ejemplo, OSNR o Q-factor) y luego aplicar un umbral.
- **Estimación de margen:** predecir cuánto margen queda respecto al umbral, lo que es útil para decisiones más finas de operación.

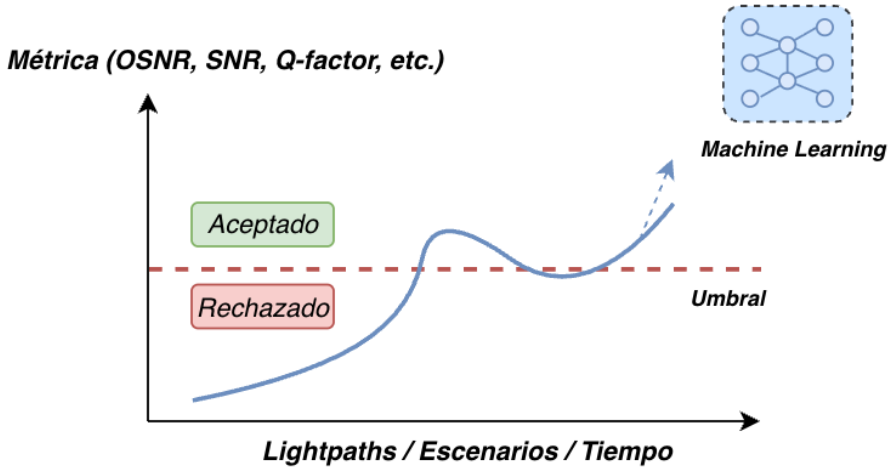


Figura 2.10: QoT como decisión: una métrica de calidad se compara contra un umbral para aceptar o rechazar una conexión. Este criterio induce un problema de clasificación (aprobado/rechazado) o de regresión (predicción de la métrica), que puede abordarse mediante técnicas de *machine learning*.

En todos los casos, el modelo busca capturar la relación entre variables del *lightpath* (ruta, espectro asignado, modulación/configuración) y el resultado observado en QoT. En redes modernas, esta relación puede estar influenciada por el estado global de la red (carga, canales vecinos, configuración de nodos), lo que hace natural el uso de técnicas de Big Data para integrar múltiples fuentes y el uso de Machine Learning para aprender patrones a partir de históricos.

La Figura 2.11 resume esta formulación: a partir de variables del *lightpath* y del estado de la red, un modelo predice ya sea una métrica de calidad o una decisión de aceptación/rechazo.

Resumen. QoT resume la viabilidad de una conexión óptica en términos de desempeño de transmisión. Se cuantifica mediante métricas como OSNR, BER y Q-factor, y típicamente se evalúa comparando estas métricas con umbrales definidos por el equipamiento y políticas de ingeniería. Dado que la QoT depende de la interacción entre ruta, espectro, modulación y estado de la red, resulta un objetivo natural para enfoques basados en datos. En la siguiente sección se introducen los fundamentos de Big Data y Machine Learning necesarios para abordar este análisis de forma escalable.

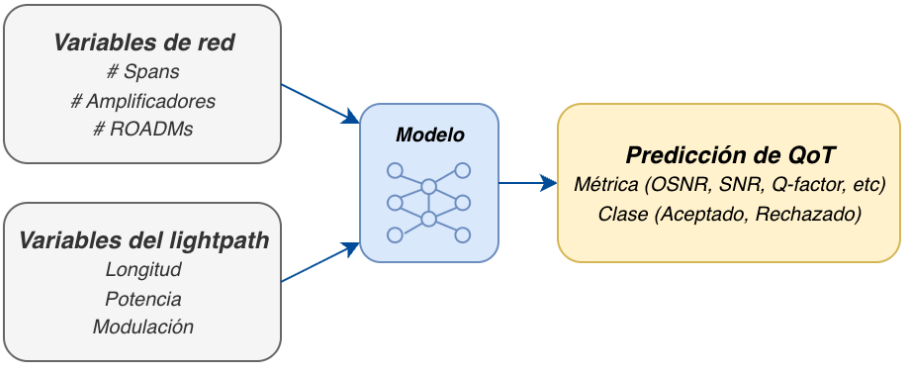


Figura 2.11: Formulación data-driven para estimación de QoT: variables de la red y del *lightpath* alimentan un modelo que predice una métrica (p. ej., OSNR/SNR/Q-factor) o una clase (aceptado/rechazado).

2.2. Big Data y Machine Learning para análisis de la QoT

El análisis *data-driven* de la QoT articula dos dimensiones: el comportamiento físico de la transmisión óptica y la evidencia generada durante la operación de la red. En redes ópticas modernas coexisten múltiples fuentes de información —configuraciones de *lightpaths*, mediciones de potencia, registros de eventos y estados de red— cuyo volumen y granularidad pueden crecer rápidamente.

Incluso en un escenario controlado como el dataset utilizado en este trabajo, la topología subyacente comprende decenas de nodos y enlaces (p. ej., 75 nodos y 99 enlaces), y la generación de tráfico y estados operativos produce un conjunto final de *millones* de muestras para modelado. Esto exige combinar capacidades de almacenamiento y procesamiento a escala (*Big Data*) con la construcción de modelos predictivos (*Machine Learning*).

Esta sección introduce los conceptos necesarios para abordar la QoT desde una perspectiva *data-driven*: (i) qué tipos de datos suelen estar disponibles, (ii) cómo se estructura un pipeline de procesamiento, y (iii) cómo se formula el aprendizaje para predecir QoT. La Figura 2.12 resume este enfoque de extremo a extremo, desde la generación y el monitoreo de datos hasta el entrenamiento de modelos y el uso de predicciones como soporte a decisiones.

2.2.1. Datos disponibles en redes ópticas y su relación con QoT

Desde el punto de vista de modelado, es útil separar la información en tres categorías, según su rol en el problema:

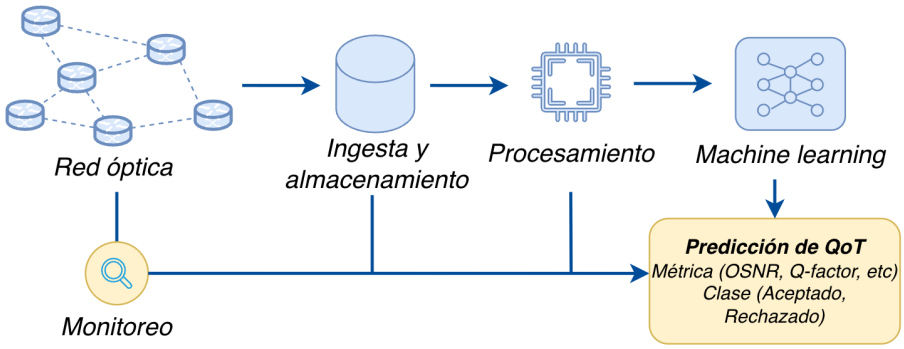


Figura 2.12: Visión general del enfoque del proyecto: la red óptica y su monitoreo generan datos que se ingieren y almacenan; luego se procesan para construir variables y entrenar modelos de *machine learning*, obteniendo una predicción de QoT (métrica o clase) como soporte a decisiones.

Variables de configuración y del servicio. Incluyen parámetros asociados al establecimiento de una conexión: nodos origen/destino, ruta (o enlaces utilizados), espectro asignado (por ejemplo, bloque de slices), ancho de canal, formato de modulación, FEC, tasa objetivo, y cualquier restricción asociada (guard-bands, tipo de transponder, etc.). Estas variables describen *cómo se decidió transmitir*.

Variables de estado y contexto de red. Representan condiciones del sistema en el momento de la conexión: ocupación espectral, canales vecinos, carga, eventos relevantes, y en general todo lo que caracteriza el "entorno" donde se establece el *lightpath*. En redes flexibles, este contexto puede ser determinante debido a fenómenos como fragmentación y filtrado concatenado.

Variables de desempeño u objetivo. Son las métricas utilizadas como salida del modelo: OSNR, SNR, Q-factor, BER (pre/post-FEC), o etiquetas derivadas de umbrales (cumple/no cumple QoT). En esta tesis, estas variables representan el resultado observado que se desea explicar o predecir.

Esta separación es útil porque guía el diseño de variables (features) y clarifica qué información es potencialmente causal (configuración + estado) y cuál es el resultado (QoT). La Figura 2.13 resume esta descomposición en términos de entradas y salidas típicas para un enfoque de estimación de QoT basado en datos.

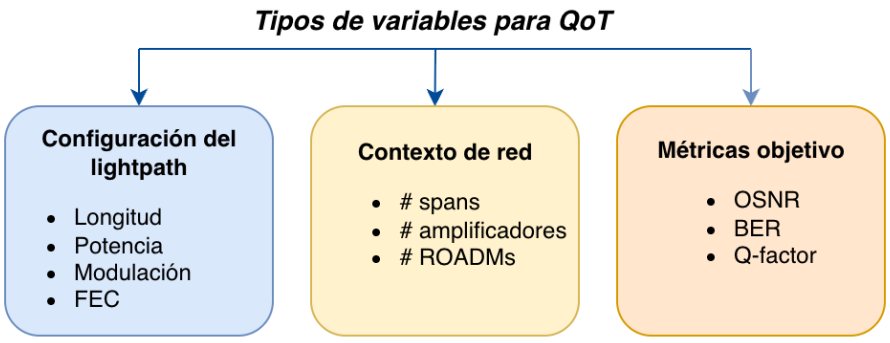


Figura 2.13: Tipos de variables para estimación de QoT: (i) configuración del *lightpath*, (ii) contexto/estado de red, y (iii) métricas objetivo (QoT) como salida del modelo.

2.2.2. Por qué Big Data es relevante en este contexto

En proyectos de análisis de QoT basados en datos, el desafío no suele ser solo entrenar un modelo, sino construir un flujo confiable desde datos crudos hasta variables consistentes y reproducibles. Esto se debe a que:

- Los datos pueden estar distribuidos en múltiples fuentes y formatos (por ejemplo, mediciones periódicas, registros por evento, configuraciones por *lightpath*).
- La escala puede ser alta: muchos enlaces, múltiples canales por enlace, mediciones en el tiempo y múltiples escenarios operativos.
- La preparación de datos para *machine learning* puede requerir operaciones costosas: joins grandes, agregaciones temporales, generación de ventanas, validaciones y cálculo de estadísticas.

En este escenario, las tecnologías de Big Data resultan útiles porque permiten almacenar y procesar volúmenes grandes de información de manera distribuida, manteniendo trazabilidad y facilitando la repetición de experimentos. En particular, es común estructurar el trabajo como un pipeline que separa etapas de ingesta, limpieza/armonización, generación de variables y extracción de datasets listos para entrenamiento.

2.2.3. Pipeline de datos orientado a modelado

Un pipeline orientado a QoT suele incluir, como mínimo, las siguientes etapas (Huyen, s.f.; NCR, DaimlerChrysler, y SPSS, 1999):

1. **Ingesta:** incorporación de datos desde las fuentes originales (simulaciones, mediciones, logs, etc.).

2. **Validación y limpieza:** detección de valores faltantes, inconsistencias, duplicados, unidades, y control de rangos básicos.

3. **Estandarización:** normalización de esquemas, nombres de variables y alineación de índices (por ejemplo, por *lightpath*, enlace, tiempo).

4. **Transformación y enriquecimiento:** joins para incorporar contexto de red, cálculo de agregados, ventanas temporales y variables derivadas.

5. **Construcción del dataset de ML:** selección de features, creación de etiquetas/targets (por ejemplo, QoT binaria por umbral) y particionado para entrenamiento/validación/test.

Un aspecto importante en este tipo de trabajos es la reproducibilidad: que el mismo pipeline aplicado sobre los mismos datos produzca el mismo dataset final. Esto es especialmente relevante cuando se comparan modelos o cuando se evalúa generalización a distintos escenarios.

La Figura 2.14 ilustra este flujo de punta a punta, desde la ingesta y depuración de fuentes heterogéneas hasta la generación de un dataset final listo para entrenar modelos.

2.2.4. Formulación del problema de aprendizaje

Una vez construido el dataset, el objetivo es aprender una función que relacione variables de configuración y contexto con el resultado de QoT. Dependiendo de cómo se defina el target, el problema puede plantearse como:

Clasificación. Se define una etiqueta binaria o multiclase según umbrales sobre una métrica (por ejemplo, "cumple QoT" / "no cumple QoT"). Esta formulación es especialmente útil cuando la decisión operacional es aceptar o rechazar una conexión.

Regresión. Se predice directamente una métrica continua (por ejemplo, OSNR o Q-factor) y luego se compara contra umbrales para decidir viabilidad. Esta opción aporta interpretabilidad adicional en términos de márgenes.

Modelos de incertidumbre o margen. En escenarios operativos puede interesar no solo la predicción puntual, sino un indicador de confianza o margen respecto al umbral, para apoyar decisiones más conservadoras o adaptativas.

2.2.5. Evaluación experimental y consideraciones prácticas

Para que los resultados sean comparables y útiles, la evaluación debe reflejar el uso previsto del modelo. En particular, en problemas de QoT pueden aparecer dos aspectos frecuentes:

Desbalance de clases. Si se define un umbral de QoT, es posible que la mayoría de las conexiones en el dataset queden del lado "viable", generando desbalance. En esos casos, métricas como *accuracy* pueden ser engañosas, y se vuelve más apropiado reportar *precision*, *recall*, *F1-score* o AUC, según el objetivo.

Particionado y generalización. En datos operacionales, el tiempo puede jugar un rol importante: entrenar en un período y evaluar en otro permite observar si el modelo generaliza a condiciones que cambian. También puede interesar evaluar generalización a nuevas topologías, configuraciones o escenarios de carga.

Estas consideraciones se retoman en capítulos posteriores al describir el diseño experimental y las métricas utilizadas en este trabajo.

Resumen. El análisis de QoT mediante Big Data y Machine Learning requiere construir un flujo reproducible desde datos de red hasta un dataset de entrenamiento, y formular el problema como clasificación o regresión de métricas de calidad. En la siguiente sección se revisa el estado del arte de estos enfoques en redes ópticas, destacando aplicaciones típicas, modelos utilizados y desafíos identificados en la literatura.

2.3. Estado del arte: Machine Learning en redes ópticas

En los últimos años, el uso de técnicas de *Machine Learning* (ML) en redes ópticas ha crecido de forma sostenida, impulsado por dos tendencias complementarias: por un lado, el aumento de complejidad en la operación (mayor flexibilidad de configuración, más grados de libertad y más escenarios dinámicos); por otro, la disponibilidad creciente de datos de monitoreo y registros operacionales que permiten construir modelos basados en experiencia histórica. En este contexto, la literatura revisa múltiples aplicaciones, donde el objetivo común es apoyar decisiones de planificación u operación mediante predicciones obtenidas a partir de datos.

Una parte relevante de estos trabajos se concentra en la *Calidad de Transmisión* (QoT), debido a que la aceptación de una conexión depende de si la señal llegará con calidad suficiente al receptor. Diferentes revisiones del área coinciden en que la predicción de QoT es uno de los casos de uso más estudiados, junto con tareas de monitoreo de desempeño y diagnóstico de fallas (Mata y cols., 2018; Rafique y Velasco, 2018; Musumeci y cols., 2019; Ayassi, Triki, Crespi, Minerva, y Laye, 2022; Villa, Tipantuña, Guamán, Arévalo, y Arguero, 2023). La Figura 2.15 resume este panorama, destacando las principales líneas de aplicación de ML que aparecen recurrentemente en la literatura.

2.3.1. Trabajos previos y contexto local

Además de los aportes reportados en la literatura internacional, esta tesis se apoya en antecedentes desarrollados en el ámbito local. En particular, la tesis de Santiago Olmedo (Olmedo Guillama, 2024) estudia la estimación de QoT mediante ML a partir de los *QoT datasets* publicados por el Fraunhofer Heinrich Hertz Institute (HHI), generados con la herramienta de simulación *PLATON*. Dicho antecedente destaca que, aun trabajando con datos sintéticos controlados, el esfuerzo experimental está fuertemente condicionado por decisiones de

ingeniería de datos: organización del volumen, consistencia de variables, trazabilidad de transformaciones y costo de recomputación al iterar sobre escenarios y modelos.

La colección de HHI incluye múltiples *datasets* con distintas configuraciones y niveles de detalle. En esta tesis se adopta como punto de partida el *dataset 01*, por tratarse del conjunto más completo en términos de información disponible y por ofrecer un marco favorable para comparar modelos de manera controlada y luego analizar su transferencia a escenarios alternativos. La descripción detallada de los *datasets*, su estructura y las decisiones de procesamiento se presentan en capítulos posteriores.

Un aspecto relevante de la colección es la coexistencia de dos representaciones complementarias: (i) una vista tabular basada en *lightpaths* (atributos agregados por conexión) y (ii) una vista basada en el *estado de la red* (descripción multidimensional del espectro/ocupación por enlace y frecuencia al aprovisionar un nuevo servicio). Esta dualidad permite formular el problema de QoT desde distintas fuentes de información, pero introduce desafíos distintos en preparación de datos, definición de *features* y selección de arquitecturas. En consecuencia, uno de los objetivos de esta tesis es atacar esas dificultades mediante un *pipeline* reproducible que permita generar datasets comparables (por representación y por escenario), sostener experimentación escalable y documentar resultados de forma sistemática.

2.3.2. Principales casos de uso reportados

En esta sección se resume la literatura de revisión sobre el uso de ML en redes ópticas. En términos generales, estas revisiones coinciden en dos rasgos prácticos del área: (i) la mayoría de los trabajos entrenan y validan modelos sobre datos sintéticos (simulación) o experimentales (laboratorio/*field-trials*), y solo una fracción menor utiliza datos provenientes de redes operativas; y (ii) existe un compromiso entre realismo y flexibilidad: la simulación permite explorar una mayor variedad de escenarios, mientras que los datos reales tienen mayor representatividad pero suelen ser escasos y difíciles de compartir. Estos puntos se discuten en revisiones recientes y también motivan trabajos orientados a compartir información entre dominios preservando privacidad. (Ayassi y cols., 2022; Safari, Shariati, Bergk, y Fischer, 2021c)

A partir de esas revisiones, se pueden agrupar los casos de uso más frecuentes de ML en redes ópticas en cinco categorías (véase la Fig. 2.15):

Predicción de QoT. Consiste en estimar si un *lightpath* (existente o candidato) cumplirá un umbral de calidad, o en predecir directamente una métrica continua para luego decidir su viabilidad. La literatura reporta enfoques que van desde modelos basados en características extremo-a-extremo (por ejemplo, longitud total y cantidad de spans/saltos) hasta representaciones que incorporan información del equipamiento y de la carga espectral (canales co-propagantes), e incluso representaciones del estado de red completo, cuando el objetivo es

capturar interacciones a nivel de red. (Musumeci y cols., 2019; Ayassi y cols., 2022)

Monitoreo de desempeño y estimación de parámetros. Incluye tareas de estimación de indicadores o parámetros relevantes a partir de telemetría y/o señales, con el objetivo de apoyar monitoreo continuo y detección temprana de degradaciones. En las revisiones se destaca que, según el caso, las variables de entrada pueden provenir de mediciones agregadas, de información del trayecto (fibra/equipos) o de salidas de modelos analíticos, y que la factibilidad práctica depende de la disponibilidad de esas variables en escenarios operativos. (Mata y cols., 2018; Ayassi y cols., 2022)

Reconocimiento de modulación y clasificación de señales. Se enfoca en identificar formatos de modulación u otras características de señal, usualmente en el contexto de sistemas coherentes y monitoreo/diagnóstico. Las revisiones incluyen trabajos donde las entradas provienen de información procesada del receptor (por ejemplo, características derivadas del DSP o constelaciones), lo que los posiciona como enfoques cercanos a clasificación de señal más que a predicción puramente topológica. (Rafique y Velasco, 2018; Ayassi y cols., 2022)

Apoyo a control y optimización. Abarca tareas donde el modelo se utiliza para asistir decisiones de operación y control, típicamente relacionadas con el aprovisionamiento y la asignación de recursos (p. ej., RWA/RSA en redes WDM/EON), selección de ruta y espectro, o ajustes de configuración bajo restricciones de QoT. En este grupo, el ML actúa como componente para acelerar o mejorar políticas de decisión (a veces integradas con controladores SDN), buscando mayor eficiencia espectral y automatización del plano de control. (Mata y cols., 2018; Ayassi y cols., 2022)

Diagnóstico, localización de fallas y mantenimiento predictivo. Abarca detección y clasificación de fallas, estimación de causa probable y predicción de eventos futuros. A nivel de datos, este grupo suele apoyarse en combinaciones de telemetría, alarmas, inventario/configuración y registros operacionales; por eso, varias revisiones lo conectan con el desafío de integrar múltiples fuentes y con la necesidad de *pipelines* consistentes para preparar datasets comparables. (Mata y cols., 2018; Ayassi y cols., 2022)

2.3.3. Formulaciones típicas del problema de QoT en la literatura

Los trabajos de QoT suelen formularse de las siguientes maneras:

- **Clasificación binaria:** predecir si una conexión cumple/no cumple QoT (por umbral de BER, Q-factor u otra métrica).

- **Clasificación multiclase:** predecir niveles discretos de calidad o márgenes de QoT.

- **Regresión:** estimar directamente el valor de una métrica continua y aplicar un umbral posterior.

En general, la elección depende de la disponibilidad de etiquetas y de cómo se definan los requisitos operacionales. Por ejemplo, si el objetivo es decidir admisión de solicitudes, la clasificación binaria se alinea naturalmente con el proceso de aceptación/rechazo; en cambio, si interesa estimar margen o priorizar reconfiguraciones, la regresión puede aportar información más rica. (Musumeci y cols., 2019; Ayassi y cols., 2022)

2.3.4. Tipos de variables utilizadas como entrada

Los surveys reportan que las variables de entrada para modelos de ML en QoT suelen combinar:

- **Características del *lightpath*:** longitud total, número de saltos, enlaces atravesados, frecuencia central o posición espectral, ancho espectral asignado, potencia, modulación/FEC (si está disponible).

- **Características de red y contexto:** ocupación del espectro, interferencia de canales vecinos, rutas coexistentes, condiciones de carga.

- **Características físicas derivadas:** agregados o proxies de degradación acumulada (por ejemplo, estimaciones o indicadores relacionados con OSNR/no linealidades).

Un patrón recurrente es que la performance del modelo depende fuertemente de qué tan bien se capture el *contexto* donde opera la conexión, especialmente en escenarios dinámicos y en redes flexibles donde los canales vecinos y la fragmentación varían con el tiempo. (Musumeci y cols., 2019; Ayassi y cols., 2022; Villa y cols., 2023)

2.3.5. Modelos utilizados y tendencias

En cuanto a modelos, la literatura abarca desde métodos clásicos hasta enfoques más recientes. Los *surveys* reportan el uso frecuente de:

- **Modelos clásicos supervisados:** regresión lineal; máquinas de vectores de soporte (SVM), que buscan un hiperplano de separación (o una función de regresión) maximizando margen; árboles y *ensembles* (por ejemplo, Random Forest y Gradient Boosting), que combinan múltiples árboles para mejorar desempeño; y k-NN, que predice a partir de la similitud con ejemplos cercanos.

- **Redes neuronales:** perceptrones multicapa (MLP), que aprenden funciones no lineales a partir de capas de neuronas completamente conectadas; y, en algunos trabajos, arquitecturas más estructuradas cuando la entrada lo justifica (por ejemplo, representaciones matriciales o basadas en topología para capturar dependencias a nivel de red).

- **Aprendizaje no supervisado/anomalías:** técnicas de *clustering* y detección de outliers para diagnóstico, segmentación de condiciones operativas y monitoreo.

Una tendencia destacada es el aumento de enfoques que intentan explotar la estructura de red (por ejemplo, incorporando representación topológica o agregando información contextual), ya que la QoT depende no solo de un único enlace sino del camino completo y de su entorno. (Ayassi y cols., 2022; Villa y cols., 2023)

2.3.6. Desafíos abiertos identificados

Más allá de los avances, los *surveys* coinciden en varios desafíos que condicionan la adopción práctica. La Figura 2.16 resume los ejes principales que aparecen de forma recurrente en la literatura.

Calidad y disponibilidad de datos. En redes reales pueden existir mediciones faltantes, cambios de instrumentación, diferentes granularidades de monitoreo y sesgos en los datos observados (por ejemplo, pocas muestras de fallas o de casos "no viables"). (Mata y cols., 2018; Ayassi y cols., 2022)

Generalización y transferencia. Modelos entrenados en una topología o configuración pueden degradar su desempeño al cambiar el escenario (nuevas rutas, nuevas modulaciones, distinta carga, o cambios de equipamiento). Se reporta interés creciente en evaluar generalización y en técnicas de transferencia. (Musumeci y cols., 2019; Villa y cols., 2023)

Interpretabilidad y uso operacional. Para integrar ML en procesos de operación, resulta valioso comprender por qué el modelo predice un resultado (por ejemplo, qué factores empujan un *lightpath* hacia no viabilidad), y cómo se incorpora la predicción en un flujo de decisión robusto. (Ayassi y cols., 2022)

Integración con control y automatización. Finalmente, un tema transversal es cómo conectar la predicción (QoT/monitoring) con acciones concretas de control y gestión, manteniendo consistencia con restricciones de red y políticas operacionales. (Ayassi y cols., 2022; Villa y cols., 2023)

Resumen. El estado del arte muestra que la aplicación de ML en redes ópticas es un área activa, con la predicción de QoT como uno de los casos de uso más relevantes. Los trabajos difieren en formulación (clasificación/regresión), variables utilizadas y modelos aplicados, pero comparten desafíos vinculados a datos, generalización e integración operacional. Este contexto motiva el enfoque de esta tesis: construir un pipeline reproducible de datos y evaluar modelos de ML para análisis/predicción de QoT en escenarios representativos. La Tabla 2.1 sintetiza, de manera comparativa, los tipos de datos, representaciones y modelos reportados en referencias seleccionadas (incluyendo el antecedente local).

Tabla 2.1: Comparación resumida de referencias representativas: tipo de datos/dataset, representación del problema y modelos utilizados. Las revisiones (*surveys*) agrupan múltiples trabajos, por lo que se reportan tendencias generales.

Referencia	**Datos / dataset**	**Representación de entrada**	**Formulación**	**Modelos reportados**
(Mata y cols., 2018; Rafique y Velasco, 2018; Musumeci y cols., 2019; Ayassi y cols., 2022; Villa y cols., 2023)	Mixto: simulación, laboratorio y, en menor medida, datos operativos (según survey)	Mixto: *features* por *lightpath*, telemetría/monitoreo, señales del receptor, contexto de red	Clasificación y regresión (QoT y tareas afines)	Clásicos (SVM, RF, boosting, k-NN), redes neuronales (MLP y variantes), no supervisado/anomalías
(Olmedo Guillama, 2024)	HHI/PLATON (*QoT datasets*); foco en dataset 01 como base experimental	Dual: tabular por *lightpath* y estado de red (LUT / vista multidimensional)	QoT como clasificación/-regresión (según métrica/umbral)	Baselines supervisados y comparación de enfoques sobre ambas representaciones
(Safari y cols., 2021a)	Simulación a escala de red (HHI/-PLATON; escenarios controlados)	Estado de red a nivel red (*tensores*/matrices por enlace-frecuencia)	Clasificación (viabilidad QoT)	Red neuronal convolucional (DCNN)
(Bergk y cols., 2022a)	Colección/curación de datasets para QoT (énfasis en calidad y visualización)	Comparación de representaciones y métricas; foco en calidad del dataset	N/A (artículo orientado a dataset/QA)	N/A (no centrado en modelo específico)
(Safari y cols., 2021c)	Multi-dominio / multi-vendor (datos distribuidos entre dominios, con restricciones de privacidad)	Estadísticas/compartición segura para entrenamiento colaborativo	Entrenamiento distribuido (según tarea)	Mecanismos de cómputo seguro para habilitar entrenamiento/-compartición

2.4. Resumen del capítulo

Este capítulo presentó los conceptos necesarios para enmarcar el objetivo del proyecto: analizar la *Calidad de Transmisión* (QoT) en redes ópticas utilizando técnicas de Big Data y Machine Learning. Primero se introdujeron los fundamentos de redes ópticas, destacando por qué la fibra y la multiplexación por longitud de onda permiten transportar grandes volúmenes de información, y cómo se establecen conexiones extremo a extremo mediante *lightpaths* y nodos reconfigurables. Luego se describió la evolución en el uso del espectro desde esquemas de grilla fija hacia Flex-Grid y redes ópticas elásticas, mostrando cómo aumenta la eficiencia y flexibilidad, pero también la complejidad operativa. A continuación se explicó el rol de la modulación y la transmisión coherente, resaltando el compromiso entre capacidad, ocupación espectral y robustez, y cómo estas decisiones influyen directamente en la calidad alcanzable.

Sobre esa base, se definió QoT como criterio de viabilidad de una conexión óptica y se introdujeron métricas relevantes para cuantificarla (por ejemplo, OSNR, BER y Q-factor), junto con la noción de umbrales y márgenes utilizados en planificación y operación. Posteriormente se abordó cómo un enfoque basado en datos permite formular el problema de QoT como tareas de clasificación o regresión, y por qué es necesario un pipeline reproducible de procesamiento a escala para construir datasets consistentes. Finalmente, se revisó el estado del arte del uso de Machine Learning en redes ópticas, identificando aplicaciones frecuentes, tipos de variables utilizadas, modelos empleados y desafíos abiertos, lo que motiva el enfoque adoptado en esta tesis.

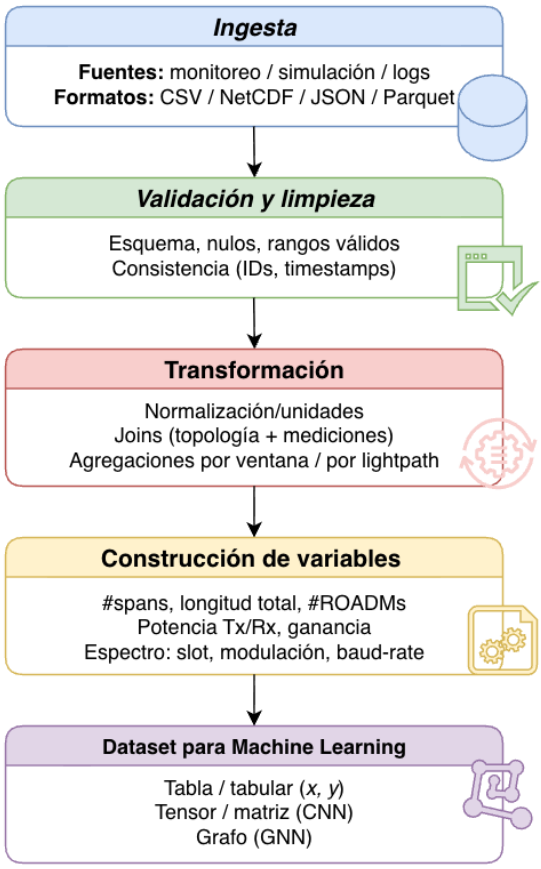


Figura 2.14: Pipeline típico de datos para QoT: ingesta de fuentes heterogéneas, validación/limpieza, transformaciones y agregaciones, construcción de variables (feature engineering) y generación del dataset final para modelos de ML (tabular, tensor o grafo).

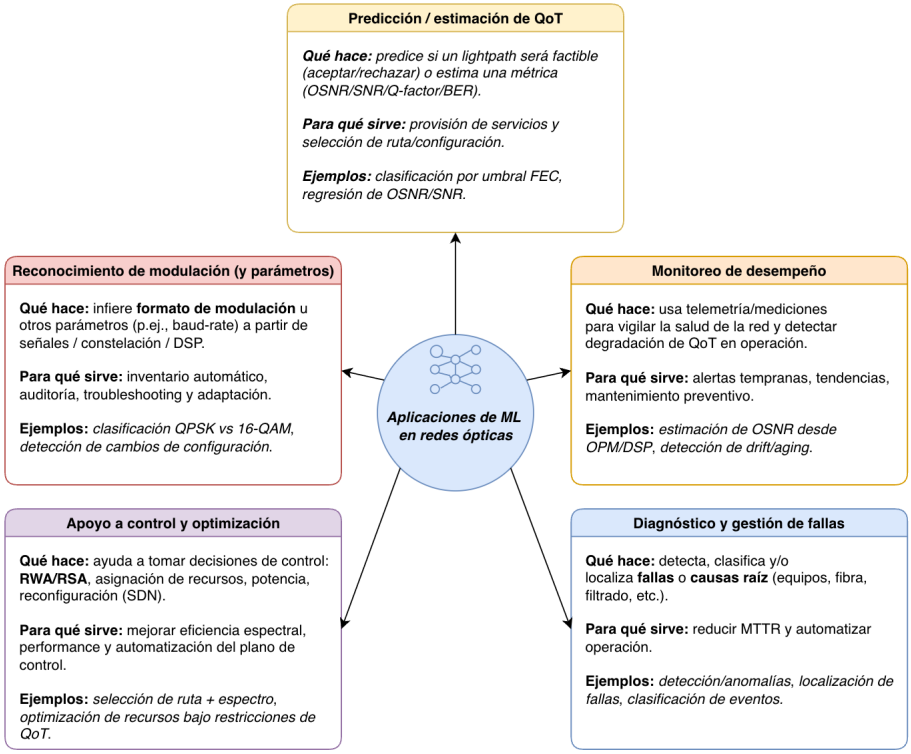


Figura 2.15: Mapa de aplicaciones de *machine learning* en redes ópticas: predicción/estimación de QoT, monitoreo de desempeño, reconocimiento de modulación y parámetros, diagnóstico y gestión de fallas, y apoyo a control y optimización.

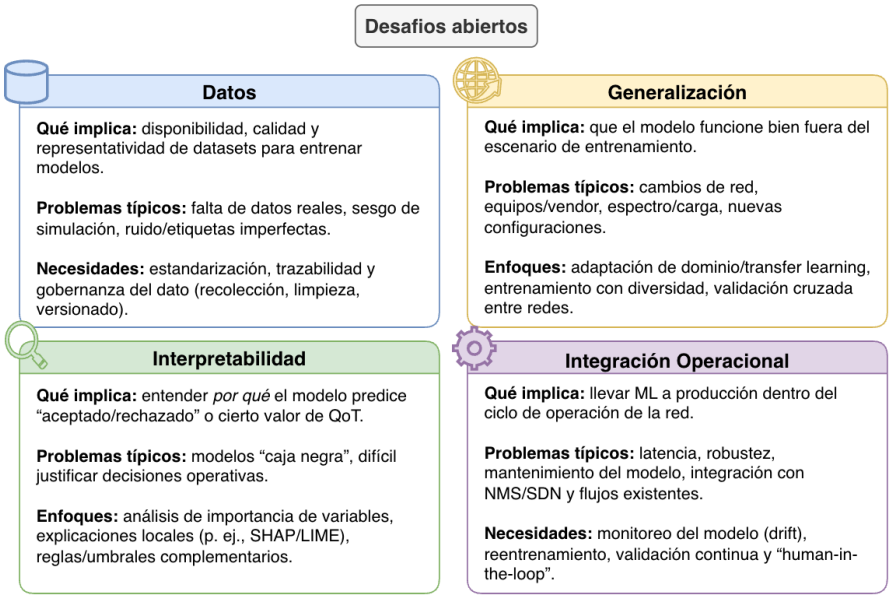


Figura 2.16: Desafíos abiertos en el uso de *machine learning* para QoT en redes ópticas: disponibilidad/calidad de datos, capacidad de generalización, interpretabilidad de los modelos e integración operacional en el ciclo de operación de la red.

Capítulo 3

Ingeniería de datos para QoT: dataset, EDA e infraestructura

Este capítulo establece la base técnica del trabajo. En primer lugar, se caracteriza el *dataset* experimental empleado, describiendo su procedencia, sus dos representaciones (*lightpath* y *network status*) y el alcance de la información disponible. En segundo lugar, se presentan las decisiones de ingeniería adoptadas para operar con datos de gran volumen de forma reproducible: conversión de formato, organización en almacenamiento distribuido, y definición de una plataforma de cómputo capaz de ejecutar transformaciones intensivas.

Sobre esta base se introduce el *pipeline* de datos que habilita la experimentación: ingesta, estandarización estructural, controles de calidad y materialización de vistas intermedias para evitar recomputaciones costosas. Finalmente, se reporta un análisis exploratorio de datos (EDA) orientado a validar supuestos y caracterizar el *estado del dato* (rangos plausibles, valores centinela, distribución de clases y relaciones entre métricas), transformando dichos hallazgos en criterios concretos que se aplican en las etapas posteriores de preparación y modelado.

El objetivo es doble: (i) dejar documentadas las condiciones bajo las cuales se procesan y utilizan los datos, asegurando trazabilidad y reproducibilidad; y (ii) fundamentar, con evidencia empírica, las elecciones metodológicas que se emplean en la etapa de experimentación, incluyendo la adopción de una estrategia de capas *Bronze–Silver–Gold* y la ejecución distribuida en la infraestructura de cómputo utilizada.

3.1. Origen del Dataset y Contexto Experimental

El conjunto de datos utilizado en este trabajo fue publicado por el *Fraunhofer Heinrich Hertz Institute* (HHI) en el marco de una línea de investigación orientada a facilitar la evaluación de técnicas de *Machine Learning* aplicadas a redes ópticas, particularmente en problemas vinculados a *Quality of Transmission* (QoT). A diferencia de datasets construidos a partir de mediciones de campo, el material provisto por HHI corresponde a un dataset **sintético**, generado mediante simulación bajo supuestos y parámetros controlados. Esto aporta dos ventajas relevantes para este proyecto: por un lado, permite cubrir una amplia variedad de estados de red y condiciones de operación; por otro, habilita trazabilidad sobre el origen de las variables y consistencia en la generación de métricas objetivo. El dataset se encuentra acompañado por documentación técnica y publicaciones asociadas que detallan su proceso de construcción, su estructura y las variables disponibles (Bergk, Shariati, Safari, y Fischer, 2022b).

La generación del dataset fue realizado con la herramienta de simulación *PLATON* (*Planning Tool for Optical Networks*), desarrollada por HHI. En términos generales, *PLATON* permite simular escenarios realistas de planificación y operación en redes ópticas, incorporando elementos de interés para QoT: selección de rutas, asignación espectral y modelado físico de degradaciones que afectan la señal a lo largo del trayecto (p. ej., acumulación de ruido, efectos asociados a la longitud de los enlaces y características de amplificación). A partir de esta simulación se producen métricas de calidad como OSNR (*Optical Signal-to-Noise Ratio*), SNR (*Signal-to-Noise Ratio*) y BER (*Bit Error Rate*), además de etiquetas binarias asociadas a la viabilidad de una conexión bajo criterios de QoT (Fraunhofer Heinrich-Hertz-Institut (HHI), s.f.). Estas variables permiten abordar tanto tareas de clasificación supervisada (QoT válida / no válida) como, alternativamente, tareas de regresión sobre métricas continuas, dependiendo del objetivo de modelado y del diseño experimental (Bergk y cols., 2022b).

Para construir los escenarios de simulación se utilizó como red de referencia la topología *CONUS* (*Continental United States*), ampliamente empleada en la literatura para evaluar algoritmos de planificación y asignación de recursos en redes ópticas. La topología considerada está compuesta por **75 nodos** y **99 enlaces bidireccionales**, con una distribución geográfica inspirada en ubicaciones reales de Estados Unidos. Sobre esta base se ejecutaron **ocho simulaciones** independientes (0 a 7). Cada simulación introduce variaciones en condiciones de carga y configuraciones de red, con el objetivo de cubrir un espectro amplio de estados operativos: desde situaciones relativamente livianas hasta escenarios más exigentes en términos de utilización de recursos y degradación de señal. Como resultado, el conjunto final abarca millones de muestras y captura diversidad suficiente como para estudiar patrones en la QoT bajo condiciones heterogéneas.

En cuanto al respaldo documental, el dataset se apoya en una combinación de *datasheets* y publicaciones donde se describen el procedimiento de generación,

los supuestos de simulación y el significado de las variables. En este trabajo se toman dichas fuentes como referencia para interpretar unidades, dominios plausibles y convenciones de codificación presentes en los datos (por ejemplo, la presencia de valores especiales o rangos atípicos identificados durante el análisis exploratorio) (Bergk y cols., 2022b). Esto es particularmente relevante porque, aun tratándose de un dataset sintético, la complejidad de la simulación y el volumen del material hacen imprescindible validar supuestos antes de avanzar hacia el modelado.

En síntesis, este dataset provee un entorno controlado para estudiar la relación entre condiciones físicas y topológicas de la red, utilización de recursos y métricas de calidad de transmisión. Esta combinación lo vuelve adecuado para realizar análisis exploratorio con criterio de calidad de datos y, posteriormente, entrenar y evaluar modelos predictivos bajo escenarios diversos. En las secciones siguientes se detalla la estructura del dataset (incluyendo sus dos representaciones principales) y se documentan las decisiones de ingeniería adoptadas para su tratamiento a escala dentro del pipeline de datos del proyecto.

3.2. Estructura del Dataset

El dataset provisto por Fraunhofer HHI presenta una particularidad clave: cada conjunto de simulaciones se publica en **dos representaciones complementarias** del mismo fenómeno, denominadas *Network Status* y *Lightpath* (Bergk y cols., 2022b). Estas dos vistas no son "datasets distintos", sino dos maneras de codificar *el mismo estado de red* y el mismo evento de aprovisionamiento de un *lightpath*. En particular, el esquema de indexación es consistente entre ambas representaciones: una muestra con un índice dado refiere al mismo escenario de red, variando únicamente la forma de descripción (atributos del camino bajo prueba versus descripción estructurada del contexto global). Esta dualidad es central para el diseño metodológico del trabajo, porque habilita dos familias de abordajes: (i) modelos *per-connection* (tabulares) que predicen QoT a partir de los atributos del *lightpath* candidato, y (ii) modelos condicionados por el *contexto global*, que incorporan el estado completo de la red al momento de la provisión para capturar interacciones y efectos de carga que no quedan reflejados en la vista tabular.

3.2.1. Representación *lightpath* (*per-connection*, tabular)

La representación *Lightpath* se centra en describir la conexión óptica bajo prueba y un conjunto acotado de variables de contexto que resultan relevantes para esa conexión. En esta vista, cada muestra se codifica como un **vector** de características: un registro tabular que resume (i) propiedades del trayecto y su configuración, y (ii) algunos descriptores del estado de los enlaces por los que el *lightpath* atraviesa (Bergk y cols., 2022b). En el paper, esta estructura se formaliza como un conjunto de D muestras $x^{(d)} \in \mathbb{R}^N$ (dataset $X \in \mathbb{R}^{D \times N}$), donde cada muestra agrupa N *features* escalares.

Además, la colección de QoT datasets define que la vista *Lightpath* incluye **31 features** y **4 atributos de meta-data**, totalizando $N = 35$ variables por muestra (en forma de "feature vectors" según la nomenclatura del artículo). Estas variables cubren distintos tipos de información:

- **Atributos de la conexión/ruta**: por ejemplo, identificadores de conexión y nodos origen/destino, longitud total del trayecto, cantidad de enlaces y spans, etc.
- **Configuración del transmisor/transceptor**: por ejemplo, frecuencia central, modulación y tasas de línea asociadas.
- **Descriptores del estado de red relevantes al trayecto**: por ejemplo, medidas asociadas a ocupación/ interferencias y estadísticas agregadas en enlaces del camino.

En la práctica, esta representación es especialmente útil para construir líneas base robustas con modelos clásicos para datos tabulares (p. ej., Random Forest o Multilayer Perceptron), y para entender qué tanto de la QoT puede explicarse *sin* necesidad de modelar explícitamente el contexto completo del estado global de la red.

3.2.2. Representación Network Status (estado global, estructurada)

La representación *Network Status* describe el estado completo de la red al momento en que se aprovisiona un nuevo *lightpath*. En particular, una muestra de *Network Status* "encapsula" el conjunto de *lightpaths* activos y el nuevo *lightpath* en el instante de provisión (Bergk y cols., 2022b). En contraste con *Lightpath*, aquí no se busca resumir el estado en un único vector, sino preservar una estructura ordenada que refleje el contexto de red.

Según el paper citado, la estructura de *Network Status* puede entenderse como un conjunto de D muestras, donde cada muestra incluye N matrices de características de dimensión $(L \times F)$, con L el número de enlaces de la red y F el número de features representadas a nivel de red. Adicionalmente, la colección se complementa con un descriptor de topología de dimensión $(L \times M)$, donde M corresponde a features topológicas (por enlace). Esta forma de representar el estado favorece modelos que explotan la estructura del problema (p. ej., modelos que consumen tensores o entradas estructuradas) y permite abordar la estimación de QoT incorporando explícitamente el contexto global de la red. Para el detalle fino de qué matrices/features se incluyen, el artículo remite a los *datasheets* públicos de la colección. La Figura 3.1 resume visualmente esta organización.

Desde el punto de vista de este trabajo, esta representación es relevante porque introduce explícitamente el contexto: ocupación y utilización a nivel de enlaces/slots, condiciones agregadas de tráfico y variables físicas/topológicas que condicionan la QoT. Esto permite estudiar, en etapas posteriores, modelos que

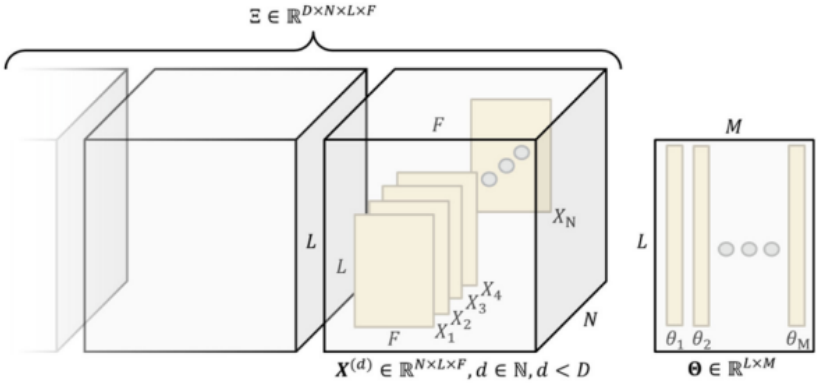


Figura 3.1: Esquema conceptual de la representación *Network Status*. El dataset se organiza como un conjunto de D muestras; cada muestra $x^{(d)}$ contiene N matrices de dimensión $L \times F$, donde L es el número de enlaces de la red y F la cantidad de variables representadas a nivel de red. En forma complementaria, la topología se describe mediante $\theta \in \mathbb{R}^{L\times M}$, con M atributos topológicos por enlace. A diferencia de la vista *Lightpath*, esta representación preserva explícitamente el contexto global de la red al momento de aprovisionar una nueva conexión. Reproducido de (Bergk y cols., 2022b).

no sólo predicen "si el *lightpath* es viable", sino *por qué* y *en qué condiciones de red* se vuelve inviable.

3.2.3. Variables objetivo incluidas (regresión y clasificación)

Ambas representaciones se publican con un conjunto común de **variables objetivo** ($T = 4$), lo cual habilita tareas tanto de regresión como de clasificación supervisada. En la notación del paper, cada muestra $x^{(d)}$ está rotulada con $y_t^{(d)}$ para $t \leq T$:

- **QoT (clase binaria, y_1):** etiqueta binaria derivada del cumplimiento de un umbral de FEC, definido sobre el BER pre-FEC. En particular, y_1 depende del umbral BER_{th} y se construye comparando y_4 (BER) contra dicho umbral.
- **OSNR (y_2):** relación señal-ruido óptica asociada al *lightpath*.
- **SNR (y_3):** relación señal-ruido asociada al *lightpath*.
- **BER pre-FEC (y_4):** tasa de error de bit previa a FEC para el *lightpath* correspondiente.

Este diseño hace explícito que el mismo dataset puede usarse para: (i) **clasificación** (predicción de QoT válida/no válida vía y_1), o (ii) **regresión** (predicción de OSNR/SNR/BER vía y_2, y_3, y_4), lo cual resulta útil para comparar enfoques y para analizar compromisos entre interpretabilidad, desempeño y costo de preparación de datos. En las secciones siguientes se retoma esta dualidad al definir el pipeline de ingeniería (cómo se materializan vistas "listas para modelado") y al justificar decisiones de EDA y selección de variables.

3.3. Características de la Red Simulada

El conjunto de datos utilizado en este trabajo se construye sobre una red óptica simulada con supuestos y parámetros orientados a representar un escenario operativo realista. La topología seleccionada, junto con la parametrización física de los enlaces y la generación de tráfico, busca capturar condiciones típicas de una red de transporte de gran escala y su impacto sobre la calidad de transmisión (QoT) (Fraunhofer Heinrich Hertz Institute (HHI), s.f.).

3.3.1. Topología CONUS

La red simulada corresponde a la topología CONUS (*Continental United States*), una configuración ampliamente utilizada como banco de pruebas en la literatura para evaluar planificación, ruteo y operación en redes ópticas de larga distancia (de Lima y Pavani, 2021). Su atractivo principal es que combina (i) una distribución geográfica realista, (ii) heterogeneidad en longitudes de enlace y (iii) una conectividad no uniforme (zonas más densas y otras más dispersas), reflejando la estructura típica de un *backbone* nacional.

Desde el punto de vista estructural, la topología puede interpretarse como un grafo $G = (V, E)$, donde cada nodo $v \in V$ representa un punto de presencia (PoP) o sitio de conmutación, y cada enlace $e \in E$ representa un tramo de fibra óptica que interconecta dos nodos (modelado como conexión bidireccional). En esta colección, la topología se describe por:

- **75 nodos**, distribuidos a lo largo del territorio continental de Estados Unidos.
- **99 enlaces**, que conectan pares de nodos mediante tramos de fibra.

Esta escala produce una red relativamente *dispersa* (grado medio $\approx 2|E|/|V| = 198/75 \simeq 2{,}64$), lo cual es consistente con topologías de *backbone*, donde se busca un equilibrio: suficiente conectividad para contar con rutas alternativas, sin convertir la red en una malla excesivamente densa. La Figura 3.2 ilustra la ubicación aproximada de los nodos y la conectividad entre ellos.

Esta estructura induce enlaces y rutas de longitudes muy diferentes. En redes ópticas, esa variabilidad se traduce en degradaciones físicas (p. ej., atenuación y dispersión) que impactan directamente en métricas como OSNR, SNR o BER.

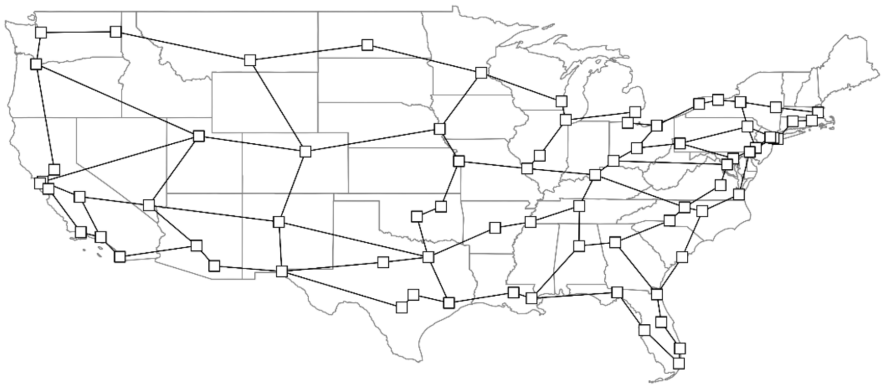


Figura 3.2: Topología CONUS utilizada en la simulación: nodos (cuadrados) y enlaces (líneas) sobre un mapa de referencia. La variabilidad geográfica induce enlaces de distinta longitud y, por ende, diferentes condiciones físicas de transmisión.

Por ello, la topología condiciona la dificultad del problema: no todas las conexiones enfrentan el mismo nivel de exigencia y parte de la variación del dataset se explica por las distancias y rutas asociadas.

3.3.2. Parámetros físicos considerados

Durante la simulación de los *lightpaths* se contemplan parámetros físicos que afectan directamente la degradación de la señal a lo largo del trayecto. Entre los más relevantes se encuentran:

- **Longitud del enlace** (km), con impacto directo en atenuación y dispersión.
- **Atenuación** (dB/km), asociada a la pérdida de potencia por unidad de distancia.
- **Dispersión cromática (CD)**, que genera ensanchamiento de pulso por diferencias de velocidad entre longitudes de onda.
- **Dispersión por modo de polarización (PMD)**, que introduce distorsión adicional por efectos de birrefringencia.
- **Número de amplificadores (EDFA)**, que compensan pérdidas pero introducen ruido acumulado.
- **Tipo de modulación**, con distintos requisitos de OSNR/SNR y tolerancias a degradaciones.

- **Configuración espectral**, incluyendo granularidad (flex-grid), frecuencia central y cantidad de *slots* asignados.

Estos parámetros se calculan o se derivan para cada conexión simulada y se reflejan en el conjunto de variables disponibles en ambas representaciones (*Lightpath* y *Network Status*), ya sea de forma directa o a través de agregaciones/atributos asociados al estado de red (Bergk y cols., 2022b).

3.3.3. Características del tráfico generado

La simulación incorpora un proceso de generación de tráfico orientado a cubrir una variedad de estados operativos de la red, con el fin de incluir tanto casos favorables como situaciones exigentes. En términos generales:

- **Pares origen–destino pseudoaleatorios**, cubriendo trayectorias diversas dentro de la topología.
- **Demandas heterogéneas**, con tasas de transmisión variables que reflejan distintos perfiles de servicio.
- **Niveles de carga diferenciados**, a través de múltiples simulaciones con configuraciones distintas, desde estados relativamente livianos hasta escenarios cercanos a saturación.
- **Asignación condicionada por el estado actual**, donde cada nueva conexión se evalúa respecto al estado de la red (p. ej., disponibilidad espectral y condiciones de QoT), generando tanto asignaciones exitosas como fallidas.

En conjunto, estos elementos permiten que el dataset contenga ejemplos variados y comparables bajo supuestos controlados, lo cual resulta útil para analizar patrones en QoT y entrenar modelos que generalicen a condiciones operativas diversas (Bergk y cols., 2022b).

3.4. Infraestructura computacional y entornos de ejecución

El procesamiento del dataset y la construcción de artefactos listos para modelado exige tomar decisiones explícitas sobre la infraestructura de cómputo. En este trabajo se emplearon dos entornos complementarios, diseñados para responder a necesidades distintas: por un lado, un ambiente controlado que facilita la iteración sobre transformaciones, la depuración y la validación de supuestos; y por otro, una plataforma distribuida orientada a ejecutar cargas de mayor costo computacional, aprovechando paralelismo y capacidad de E/S.

En conjunto, ambos entornos permiten separar la definición y verificación del flujo de datos (consistencia, integridad y parametrización) de su ejecución sobre volúmenes sustanciales. A continuación se describen las características

principales de cada entorno, su forma de acceso y operación, y los criterios adoptados para seleccionar uno u otro según el tipo de tarea.

3.4.1. Entorno de ejecución contenerizado en máquina virtual

Como parte de la infraestructura empleada, se dispuso de una máquina virtual (VM) que ejecuta un stack distribuido *contenedorizado*. El objetivo de este entorno fue ofrecer una base controlada y reproducible para ejecutar el motor de procesamiento, validar transformaciones y sostener tareas de exploración e inspección del dataset sin depender de configuraciones particulares del host.

El despliegue se orquestó mediante *Docker Compose*, que declara de forma explícita los servicios del motor de cómputo (*Spark*), el almacenamiento distribuido (*HDFS*) y una capa de interacción (notebooks/UI) utilizada para análisis. En esta solución, los contenedores se organizan en una red interna (*Docker bridge network*) que habilita la comunicación entre componentes. La Figura 3.3 resume la arquitectura desplegada en la VM.

La Tabla 3.1 resume parámetros representativos de configuración utilizados en este entorno. Estos valores se definieron en archivos declarativos (configuración de Spark/HDFS y variables de entorno) y se ajustaron de forma iterativa para poder ejecutar transformaciones intensivas (agrupamientos, agregaciones y reestructuración de datos) con comportamiento estable.

Tabla 3.1: Parámetros principales de configuración de Spark y HDFS.

Componente	**Parámetro**	**Valor**
Spark Master	`spark.default.parallelism`	12
Spark Worker	`SPARK_WORKER_CORES`	8
Spark Worker	`SPARK_WORKER_MEMORY`	8g
Executor	`spark.executor.memory`	48g
Executor	`spark.executor.cores`	4
Driver	`spark.driver.memory`	24g
HDFS	`dfs.replication`	2
HDFS	`dfs.blocksize`	128MB

Desde el punto de vista operativo, este entorno aportó dos ventajas prácticas: **reproducibilidad** (mismas versiones y dependencias al reinstanciar el stack) y **aislamiento** (evitar dependencias de máquina difíciles de auditar). No obstante, al ejecutar transformaciones de alto costo sobre volúmenes sustanciales —en particular para la representación *network status*— el factor limitante pasa a ser el costo de mover datos: operaciones con *shuffle* intensivo tienden a disparar *spill* a disco y múltiples lecturas/escrituras intermedias, volviendo el proceso sensible a memoria y E/S.

Como referencia, en una VM con **16 CPU** y **92 GB de memoria**, la conversión completa del dataset de *network status* desde su formato de origen hacia una primera materialización en almacenamiento distribuido insumió del

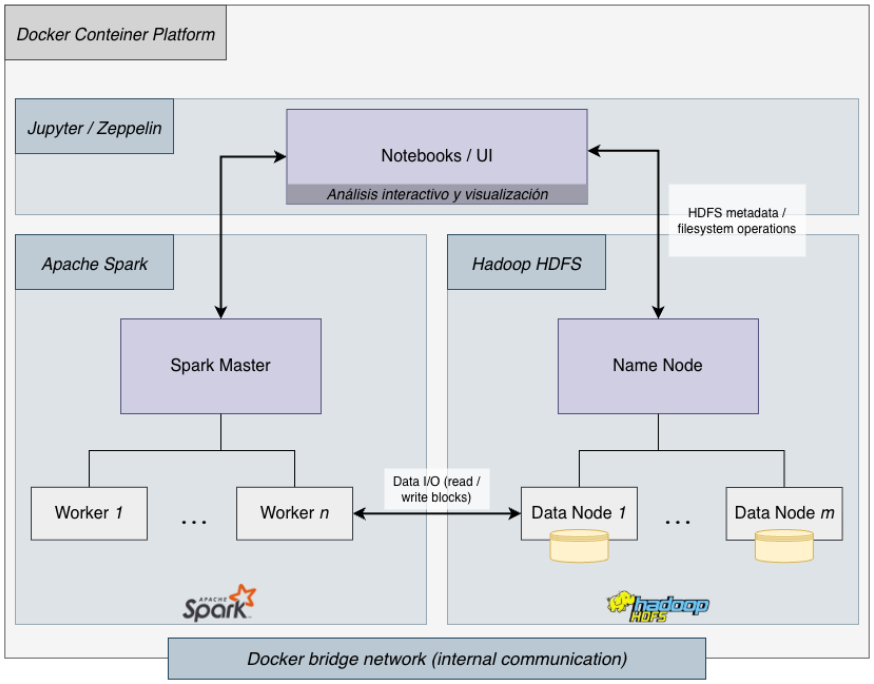


Figura 3.3: Arquitectura del entorno distribuido en VM: *Docker Compose* orquesta Spark (Master/Workers), HDFS (NameNode/DataNodes) y una capa de interacción (Jupyter/Zeppelin) sobre una red interna.

orden de ∼**10 días**. Este resultado motivó el uso de una plataforma distribuida para las etapas de mayor demanda computacional, descrita en la subsección siguiente.

Acceso a la VM

El acceso a la VM se realizó de forma remota mediante SSH utilizando un *jump host* institucional. Para operar el stack se empleó *port forwarding* hacia los servicios con interfaz web (por ejemplo, paneles de HDFS y Spark, y la capa de interacción), de modo de mantener el despliegue aislado y accesible sin exponer puertos de manera pública. Adicionalmente, dado que la VM no cuenta con salida directa a Internet, se configuró el acceso mediante el proxy institucional para habilitar la descarga de dependencias y la ejecución de tareas que requieren conectividad. El procedimiento concreto de conexión y tunelización se documenta en el Anexo D.

3.4.2. Entorno de procesamiento distribuido a escala (ClusterUY)

Para las etapas de mayor costo computacional se utilizó *ClusterUY* (*ClusterUY*, s.f.), una infraestructura de cómputo de alto desempeño orientada a la ejecución de trabajos por lotes y a la asignación explícita de recursos. Este entorno resulta especialmente adecuado cuando el volumen de datos y el patrón de acceso (lecturas/escrituras repetidas, materializaciones intermedias y redistribuciones globales) vuelven inestable una ejecución prolongada en un contexto interactivo.

En nuestro caso, la motivación fue principalmente operativa: varias transformaciones del pipeline implican movimientos intensivos de datos entre particiones (por ejemplo, *joins*, agregaciones y **reestructuración de datos** a nivel de tablas), lo que incrementa el costo de *shuffle* y hace que el desempeño dependa fuertemente de memoria y E/S. En un entorno batch, estos procesos se encapsulan como ejecuciones independientes, con recursos dedicados y parámetros controlados, facilitando ejecuciones repetibles y comparables.

El envío y la gestión de ejecuciones se realizaron mediante *Slurm* (*Slurm Workload Manager: Documentation*, s.f.), que actúa como planificador de trabajos y gestor de recursos. El flujo general consiste en acceder al nodo de login, preparar el comando de ejecución (por ejemplo, `spark-submit`) y enviar el job con `sbatch`, declarando CPU, memoria y límites de tiempo acordes a cada etapa. Slurm asigna nodos de cómputo, instancia el proceso *driver* y coordina la ejecución distribuida a través de *executors*.

La Figura 3.4 resume la arquitectura adoptada. La entrada/salida de datasets y artefactos (datasets intermedios, modelos y resultados) se realiza sobre almacenamiento compartido, mientras que los temporales de Spark —incluyendo directorios asociados a *spill* y *shuffle*— se derivan a un área de trabajo temporal (*scratch*) disponible en el cluster. Este punto es relevante: al ubicar temporales pesados en *scratch*, se reducen cuellos de botella de E/S sobre el almacenamiento persistente y se mejora la estabilidad de las etapas más sensibles del pipeline.

Desde el punto de vista metodológico, en línea con enfoques que modelan el procesamiento como un conjunto de etapas declarativas, parametrizables y re-ejecutables en forma de *workflow* (Köster y Rahmann, 2012), se adoptó un esquema de ejecuciones **por trabajos**, donde cada etapa del pipeline se define como una unidad parametrizable (rutas de entrada/salida, particionado por *buckets*, configuración de recursos y directorios temporales). En la práctica, esto permitió: (i) evitar dependencias de sesiones interactivas extensas, (ii) conservar trazas de ejecución por corrida (tiempos, tamaños de salida y conteos relevantes) como evidencia para comparar variantes, y (iii) aprovechar de forma sistemática el almacenamiento temporal del cluster para reducir la fricción asociada a *shuffle* y materializaciones intermedias.

A nivel de implementación, las ejecuciones se encapsularon en scripts y *wrappers* de envío (combinando `sbatch` con `spark-submit`), de modo que una corrida quede definida por el artefacto de código invocado, sus parámetros y los recursos asignados por el planificador. Este criterio, coherente con prácticas orientadas a reproducibilidad operacional (misma definición de corrida, mismos parámetros

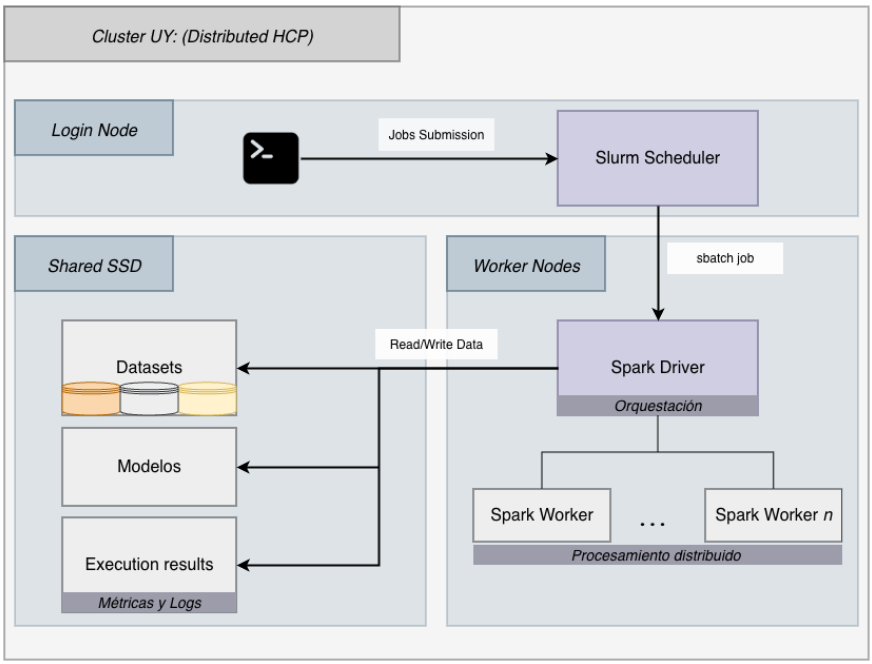


Figura 3.4: Arquitectura de ejecución batch en ClusterUY: las ejecuciones se envían desde el nodo de login mediante `sbatch`/`spark-submit`, el planificador Slurm asigna recursos en nodos de cómputo, y se ejecuta un *Spark Driver* que coordina *Spark Workers/Executors*. La E/S se realiza sobre almacenamiento compartido (datasets, modelos y resultados), utilizando *scratch* para temporales y etapas con *spill/shuffle*.

y trazas comparables) (Di Tommaso y cols., 2017), transformó el cómputo pesado en un proceso controlado y repetible, complementario al entorno en VM, que se mantuvo como base para inspección y validación en escenarios de menor escala.

3.4.3. Comparación y criterios de uso

La elección de plataforma no se trató de "tener dos lugares para correr lo mismo", sino de asignar cada tipo de tarea al entorno que reduce fricción y riesgo. En términos generales, la VM ofrece un espacio controlado para iterar y validar (configuración estable, servicios siempre disponibles y acceso directo a interfaces), mientras que el cluster habilita ejecuciones de mayor escala con recursos dimensionados y un modo de operación orientado a trabajos por lotes.

En este proyecto, los criterios que guiaron el uso de cada entorno pueden

resumirse en cuatro ejes:

- **Escalabilidad:** capacidad de procesar volúmenes grandes y construir artefactos intermedios sin depender de que los datos "entren" completamente en memoria.
- **Eficiencia de E/S:** minimizar el costo de lecturas/escrituras intermedias y reducir cuellos de botella asociados a *shuffle* y materializaciones.
- **Reproducibilidad:** configuración explícita del entorno y posibilidad de re-ejecutar etapas bajo condiciones comparables (parámetros, recursos y salidas).
- **Capacidad de verificación:** facilidad para inspeccionar resultados intermedios y detectar desvíos tempranos del esquema, dominios o supuestos del pipeline.

Bajo estos criterios, la VM se utilizó preferentemente para inspección, validación y ajustes finos sobre subconjuntos o vistas previas, así como para estabilizar transformaciones antes de ejecutarlas en escala. En contraste, ClusterUY se reservó para las etapas en las que el costo principal proviene del movimiento de datos (redistribuciones, agregaciones a gran escala y reestructuración de tablas), donde el modo batch y la disponibilidad de mayores recursos vuelven la ejecución más previsible y manejable.

La Tabla 3.2 presenta escenarios representativos que ilustran el orden de magnitud de los costos involucrados en cada plataforma. Estos valores deben interpretarse como referencias operativas (dependientes de particionado, carga del sistema y parámetros de ejecución), pero resultan útiles para justificar la separación de responsabilidades entre entornos.

Tabla 3.2: Comparación representativa de tiempos de ejecución entre VM y ClusterUY.

Proceso	**VM**	**ClusterUY**
Generación de la capa **Silver** para la representación *network status*. *Recursos de referencia:* VM (16 CPU, 92 GB) vs. ClusterUY (64 CPU, 200 GB).	∼10 días	<10 h

En síntesis, la combinación de ambos entornos permitió mantener un ciclo de iteración razonable para validar y depurar el pipeline, sin que las etapas de cómputo pesado impusieran tiempos de espera incompatibles con la experimentación. Esta complementariedad se vuelve especialmente relevante al pasar de exploraciones iniciales a la materialización sistemática de capas reutilizables, tema central de la sección siguiente.

3.5. Pipeline de datos: *Bronze–Silver–Gold*

Esta sección describe el **pipeline de datos** construido para transformar el material original del dataset en artefactos consistentes y reutilizables para modelado. El foco no está puesto en una secuencia histórica de ejecución, sino en la **estructura lógica** del proceso: qué responsabilidades cumple cada etapa, qué productos genera y qué decisiones de ingeniería permiten escalar el procesamiento manteniendo trazabilidad y reproducibilidad. En particular, se adopta un esquema por capas (*Bronze–Silver–Gold*) para desacoplar el costo de transformación del ciclo de experimentación, de modo que el entrenamiento y la evaluación puedan iterar sobre salidas estables sin re-ejecutar etapas pesadas.

3.5.1. Objetivos y restricciones de diseño

El diseño del pipeline estuvo guiado por un objetivo central: **convertir un dataset de gran volumen y estructura compleja en representaciones aptas para aprendizaje automático** sin perder trazabilidad respecto del origen. En términos prácticos, esto implicó construir un flujo que preserve la semántica de los datos, produzca salidas determinísticas y permita re-ejecutar etapas bajo condiciones equivalentes (mismas transformaciones, mismos parámetros y mismo esquema).

A partir de ese objetivo, se definieron los siguientes principios de diseño:

- **Trazabilidad end-to-end:** cada artefacto de salida debe poder vincularse con precisión al conjunto de transformaciones aplicadas (filtros, columnas derivadas y supuestos), manteniendo el linaje desde el dataset original hasta las vistas listas para modelado.

- **Reproducibilidad operacional:** el pipeline debe ejecutarse de forma controlada y repetible, con configuración explícita (rutas, particionado, parámetros de Spark) y sin depender de pasos manuales difíciles de auditar.

- **Desacoplamiento entre transformación y modelado:** las etapas intensivas en cómputo y E/S deben ejecutarse una vez y materializarse como capas reutilizables, de modo que la experimentación sobre modelos no requiera recomputar transformaciones costosas.

- **Escalabilidad y eficiencia:** el pipeline debe ser compatible con ejecución distribuida, minimizando cuellos de botella de E/S y controlando el impacto de operaciones con *shuffle* sobre memoria y disco.

- **Compatibilidad con múltiples tareas y modelos:** la salida no se reduce a una única tabla "final", sino que debe permitir derivar vistas adaptadas a distintos enfoques de modelado (por ejemplo, salidas tabulares para modelos clásicos y salidas estructuradas cuando corresponde).

Estas restricciones se traducen en una estrategia por capas (Bronze–Silver–Gold), donde cada capa cumple una responsabilidad bien definida y produce artefactos versionables. En las subsecciones siguientes se describe esta organización y las decisiones adoptadas para que el pipeline sea estable y escalable.

3.5.2. Visión general end-to-end y esquema medallón

El artefacto central que habilita la experimentación de este trabajo es el **pipeline de datos**. Su rol es transformar el material original del dataset en un conjunto de salidas **consistentes, trazables y reutilizables** para modelado, de forma que el ciclo de entrenamiento y evaluación no dependa de transformaciones *ad-hoc* ni de recomputar etapas costosas en cada iteración.

Para estructurar este proceso se adopta un esquema por capas **Bronze–Silver–Gold**, común en arquitecturas de datos modernas (conocido como *medallion architecture*) (Databricks, 2020). Esta organización separa responsabilidades y explicita contratos entre etapas:

- **Bronze**: preserva una copia fiel del origen (minimizando decisiones irreversibles).
- **Silver**: normaliza, depura y estandariza la estructura para análisis y construcción de representaciones.
- **Gold**: materializa vistas *orientadas a modelado* (no necesariamente únicas), priorizando eficiencia y comparabilidad.

La Figura 3.5 resume la solución end-to-end, desde el dataset (y sus dos representaciones principales) hasta el pipeline de datos por capas y el posterior pipeline de aprendizaje automático. A su vez, la Figura 3.6 sintetiza las responsabilidades de cada capa y el tipo de artefactos que se materializan para garantizar trazabilidad, persistencia y reutilización durante la experimentación.

Un punto clave de diseño es que **Bronze y Silver se definen por representación** (p. ej., *network status* por un lado y *lightpath* por otro) y se mantienen agnósticas al modelo. En cambio, **Gold no es único**: se construyen distintas variantes según la tarea y el tipo de algoritmo a evaluar (tabular, secuencial, tensores, etc.), preservando siempre trazabilidad hacia Silver/Bronze.

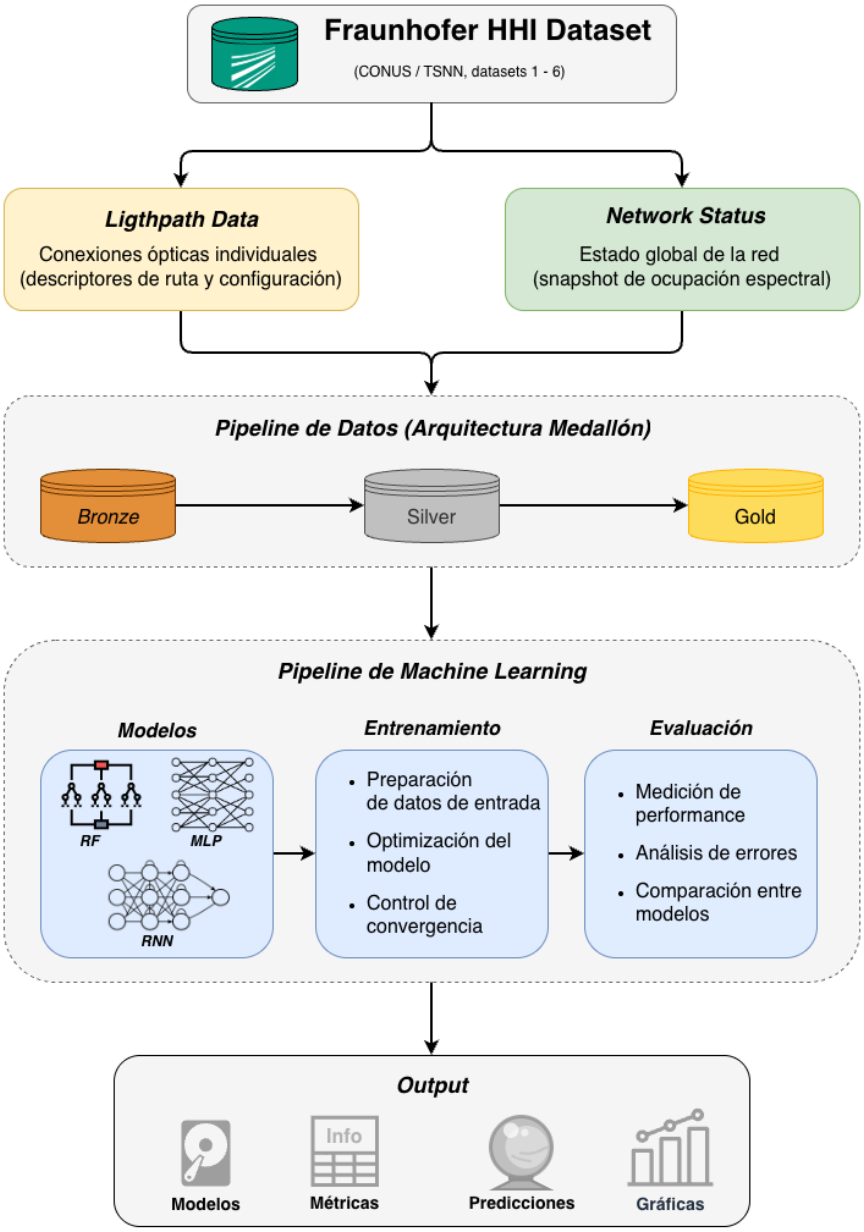


Figura 3.5: Visión general de la solución: desde el dataset Fraunhofer HHI (representaciones *Lightpath* y *Network Status*) hasta el pipeline de datos (*Bronze–Silver–Gold*) y el pipeline de ML (entrenamiento, evaluación y generación de artefactos).

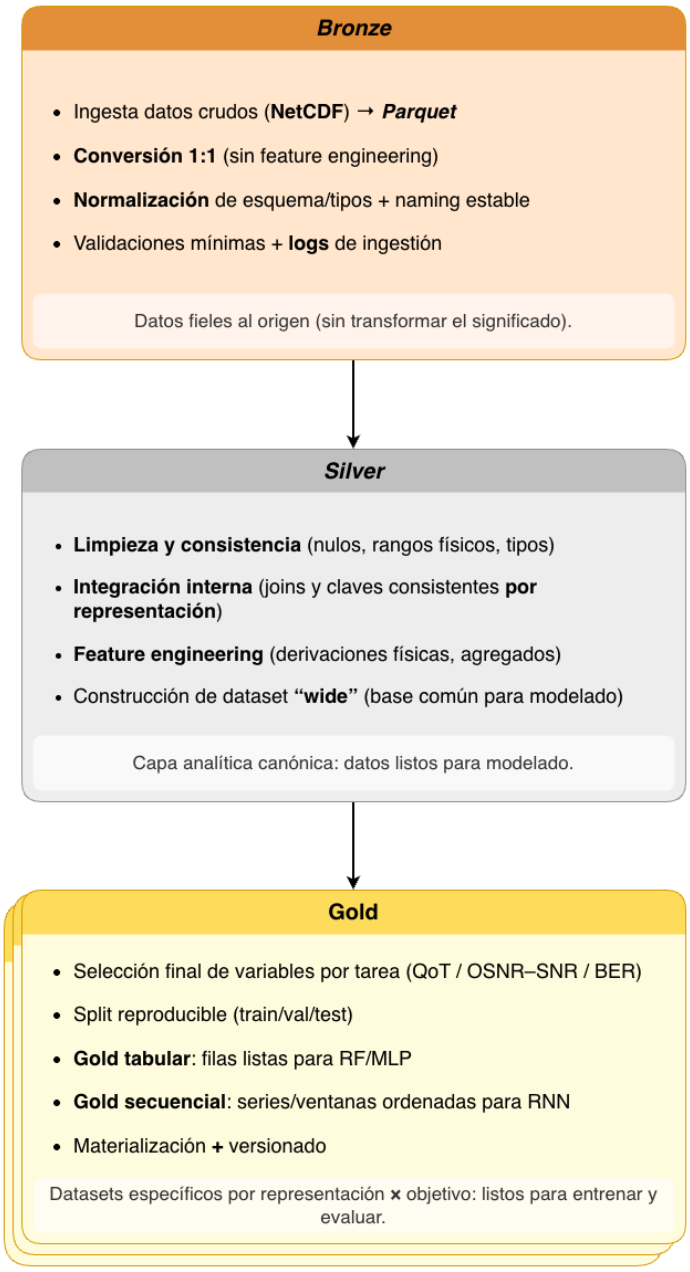


Figura 3.6: Pipeline de datos (*Bronze–Silver–Gold*): responsabilidades por capa y artefactos generados para soportar trazabilidad, persistencia y reutilización en la experimentación.

3.5.3. Capa Bronze: ingesta y preservación fiel del origen

Qué se conserva y por qué

La capa **Bronze** se define como una materialización **fiel** del dataset original en un formato columnar eficiente (Parquet), evitando introducir decisiones de limpieza (imputación de nulos, eliminación de outliers, normalización) o agregación que puedan "tapar" el origen. En esta etapa se priorizan dos objetivos: (i) **trazabilidad** (poder auditar y reconstruir el camino hasta los datos originales) y (ii) **recomputabilidad** (poder rehacer etapas posteriores sin reingestar desde fuentes más difíciles de manipular).

En términos prácticos, Bronze funciona como la **fuente canónica** del pipeline: la representación más cercana al origen que actúa como referencia estable para todas las etapas posteriores. Cualquier transformación aplicada en Silver o Gold puede rastrearse hasta Bronze, y si una etapa intermedia necesita rehacerse, Bronze provee el punto de partida sin necesidad de volver a la fuente original. Sobre esta base se construyen normalizaciones, validaciones y representaciones para análisis y modelado.

Estructura de los datos (por representación)

Para *network status*, Bronze queda naturalmente representado como una *vista larga*: cada fila corresponde a una combinación `(sample, link, freq, nt_feat, metric)` con los valores asociados. Esta decisión permite preservar el contenido original sin forzar tempranamente una estructura tabular "ancha", que se construye recién en Silver. A modo ilustrativo, la Tabla 3.3 resume el esquema conceptual mínimo que caracteriza esta capa para *network status*.

Tabla 3.3: Esquema conceptual mínimo de *network status* en Bronze.

Campo	Tipo	Rol
`sample`	BIGINT	Identificador de muestra (estado global).
`link`	INT/BIGINT	Índice de enlace.
`freq`	DOUBLE	Frecuencia/slot (grilla espectral).
`lp_feat`	STRING	Nombre del atributo asociado al *lightpath*.
`nt_feat`	STRING	Nombre del atributo topológico/de red.
`metric`	STRING	Métrica física u objetivo (`osnr`, `snr`, `ber`, `class`).
`data`	DOUBLE	Valor crudo asociado a `lp_feat`.
`ntd`	DOUBLE	Valor crudo asociado a `nt_feat`.
`target`	DOUBLE/INT	Valor de la métrica indicada en `metric`.

Para *lightpath*, Bronze sigue el mismo principio: materializar el origen en un formato eficiente, manteniendo el significado de los campos y evitando normalizaciones irreversibles. Esto permite que las decisiones de estandarización y selección de features queden explícitas aguas abajo, donde pueden documentarse y versionarse.

Costos, volumen y lecciones aprendidas

Materializar Bronze es, típicamente, una de las etapas más costosas cuando el dataset es voluminoso. En nuestro caso, el tamaño y la estructura de *network status* hacen que esta capa imponga un costo inicial alto, pero amortizable: al persistir el resultado, las etapas posteriores pueden iterar sobre Parquet distribuido, con lecturas selectivas por columnas y sin volver a pagar el costo de ingesta desde el formato original. Como referencia operativa, la conversión inicial y persistencia de *network status* hacia una primera materialización del pipeline tomó del orden de **∼10 días** en el entorno de VM (Sección 3.4.1), lo que motivó reservar la infraestructura distribuida para las etapas más intensivas.

Además, preservar Bronze como réplica fiel reduce ambigüedades: ante divergencias en resultados o cambios de criterios en etapas posteriores, siempre existe un "piso" estable desde el cual recomputar y comparar. En este sentido, el costo inicial se justifica por el beneficio metodológico y operativo: habilita iteración sobre capas derivadas (Silver/Gold) bajo condiciones controladas y con trazabilidad explícita.

3.5.4. Capa Silver: normalización, limpieza y estandarización

Transformaciones estructurales (long → wide cuando aplica)

La capa **Silver** toma Bronze y produce una tabla *lista para análisis* con una granularidad estable por unidad de observación. En *network status*, el objetivo es obtener **una fila por** `(sample, link, freq)` que contenga, en columnas numéricas: (i) métricas físicas y etiqueta, y (ii) contexto topológico y atributos del *lightpath*.

En términos operativos, Silver realiza una **reestructuración de datos** que consolida información que en Bronze aparece distribuida en filas (por `metric`, `nt_feat` o `lp_feat`). Por ejemplo:

- consolida métricas físicas en columnas $\{m\_\texttt{class}, m\_\texttt{osnr}, m\_\texttt{snr}, m\_\texttt{ber}\}$;
- aterriza atributos topológicos en columnas numéricas (p. ej., `len_km`, `nspans`, `src_degree`, `dst_degree`);
- incorpora atributos del *lightpath* (p. ej., `conn_id`, `mod_order`, `path_len`);
- deriva índices auxiliares (p. ej., `freq_idx`) y campos de procedencia (p. ej., `sample_bucket`) para trazabilidad.

El efecto práctico es que EDA y construcción de representaciones pasan a operar sobre un esquema "ancho" estable, evitando reestructuraciones pesadas repetidas y estandarizando el acceso a features.

Limpieza y controles de calidad (data contracts)

Silver es también el punto natural para aplicar **limpieza de datos** e introducir **validaciones** y controles de calidad, ya que aquí se explicitan y corrigen los supuestos del modelado. Estas verificaciones se definen de forma manual a partir de conocimiento de dominio: valores físicamente imposibles (por ejemplo, OSNR negativos o tasas de error mayores a 1), inconsistencias estructurales esperables en la topología, y restricciones de completitud mínima para que una observación sea utilizable en el modelado.

- presencia y tipo de columnas esperadas;
- rangos plausibles para métricas (por ejemplo, evitar valores de relleno o fuera de dominio);
- consistencia de claves (`sample, link, freq`) y ausencia de duplicados por unidad de observación;
- completitud mínima de features críticas para cada representación.

Estas verificaciones actúan como contratos que delimitan "qué significa" que un dataset esté listo para ser consumido aguas abajo.

Particionado/estrategias para shuffle y memoria

Dado el volumen de *network status*, la construcción de Silver requiere decisiones de ejecución orientadas a estabilidad: particionado coherente (por ejemplo por buckets), escritura en Parquet con tamaños de archivo adecuados, y separación en unidades de trabajo que eviten recomputar todo ante fallas. Estas estrategias no cambian el significado de los datos, pero sí determinan la viabilidad de ejecutar el pipeline de forma reproducible sobre infraestructura distribuida.

En síntesis, Silver define la **representación canónica para análisis**: una base estable, validada y estandarizada desde la cual se derivan vistas Gold específicas para modelado.

3.5.5. Capa Gold: vistas orientadas a modelado

Gold tabular (modelos clásicos y comparables)

La capa **Gold** transforma Silver en datasets **directamente consumibles** por modelos, priorizando eficiencia de lectura, estabilidad del esquema y comparabilidad entre corridas. Una primera familia corresponde a **Gold tabular**: datasets en formato tabular (p. ej., Parquet) con una fila por unidad de aprendizaje y columnas numéricas estables, adecuados para modelos clásicos de ML (árboles, ensembles, regresores y clasificadores tradicionales).

En esta familia, el diseño explicita qué features y métricas se incluyen, y cómo se construyen las particiones necesarias para entrenamiento/validación, manteniendo trazabilidad hacia Silver.

Gold estructurado (secuencias/tensores para modelos con sesgo inductivo)

La segunda familia corresponde a representaciones **estructuradas** (secuencias o tensores), donde la información se organiza por muestra y por índices (por ejemplo, frecuencia/enlace). Estas vistas se materializan típicamente como arreglos densos o artefactos compactos (p. ej., `.npz`), y son especialmente útiles para modelos que explotan estructura (por ejemplo redes con entradas secuenciales o matriciales).

Desde el punto de vista metodológico, este es el puente entre "datos listos para analizar" (Silver) y "datos listos para entrenar" (Gold): la estructura final deja de estar definida por la conveniencia del almacenamiento y pasa a estar definida por el **patrón de consumo del modelo**.

Por qué Gold no es único (por tarea/modelo) y cómo se versiona

A diferencia de Bronze/Silver, Gold **no es una única vista**: existen múltiples Gold según representación, tarea y familia de modelos. Conceptualmente, cada Gold encapsula: (i) una **selección explícita** de variables (features/targets) y (ii) una **estructura de datos** que favorece el consumo eficiente en entrenamiento.

Para garantizar comparabilidad, cada variante de Gold se **versiona** (por ejemplo, por configuración de features, estrategia de particionado o definición de índices), de modo que los resultados reportados puedan trazarse a una salida específica del pipeline.

Un efecto directo de materializar Gold es desacoplar el costo de transformación del costo de modelado: el ciclo pasa de "transformar + entrenar" a "leer Gold + entrenar", habilitando iteración real sobre modelos e hiperparámetros. Como referencia, una prueba de entrenamiento desde Gold (1 lote) tomó del orden de minutos, y un entrenamiento de mayor escala (70 lotes) se completó en pocas horas sobre el entorno distribuido.

3.6. Análisis Exploratorio de Datos

El análisis exploratorio de datos (EDA, por sus siglas en inglés) es una etapa clave en este trabajo porque cumple dos funciones en paralelo. Por un lado, permite entender cómo es realmente el dato con el que se va a trabajar: su granularidad, sus campos, la relación entre variables y la forma en que se codifican las métricas y atributos de red. Por otro lado, actúa como una primera instancia de control de calidad: detecta valores y rangos fuera del dominio físico, inconsistencias de codificación y desbalances de clases que, si no se identifican temprano, terminan afectando tanto al *pipeline* como a la evaluación posterior de modelos.

Además del resultado descriptivo, el EDA también cumple un rol de ingeniería: en un dataset con volumen y estructura no triviales, los primeros análisis sirven para dimensionar el costo real de transformaciones que luego se volverán

recurrentes (por ejemplo, pasos que requieren reordenar grandes volúmenes, consolidar información y materializar resultados intermedios), identificar cuellos de botella de E/S y anticipar qué parte del flujo puede ejecutarse de forma interactiva (notebooks) y qué parte debe migrarse a un procesamiento por lotes con recursos explícitamente dimensionados. Este aprendizaje conecta directamente con las decisiones de plataforma y ejecución descritas en la Sección 3.4, y con la lógica de materialización y reutilización del pipeline Bronze–Silver–Gold presentada en este mismo capítulo.

Aunque el dataset de HHI incluye dos representaciones (*lightpath* y *network status*), en esta sección se comienza por *lightpath* por ser la vista más básica y directa: resume la conexión a nivel de *lightpath* con variables agregadas, lo que facilita validar rápidamente dominios, distribuciones y relaciones generales. Luego se aborda *network status*, que es la vista más exigente en volumen y estructura, y por lo tanto la que condiciona con más fuerza tanto el costo computacional como las reglas de limpieza/estandarización necesarias antes de construir datasets listos para modelado. En ambos casos, el análisis se apoya en notebooks exploratorios y se complementa con reportes automáticos generados con `ydata-profiling`, de los cuales se extraen métricas y capturas como evidencia de calidad y estructura del dato (Clemente y cols., 2023).

3.6.1. Objetivo, alcance y artefactos

El objetivo de esta sección es caracterizar empíricamente el dataset (estructura, distribución y calidad) y, a partir de esa evidencia, derivar decisiones concretas para el *pipeline* de datos y la experimentación posterior. En particular, el EDA se utiliza para responder un conjunto acotado de preguntas guía que atraviesan ambas representaciones:

- **Estructura real del dato:** ¿qué representa una observación en cada vista (*lightpath* y *network status*) y qué implicancias tiene esa granularidad al construir variables y datasets de entrada?
- **Distribuciones y dominio:** ¿qué rangos y formas de distribución presentan las métricas (p. ej., OSNR/SNR/BER) y los atributos de red, y cuáles se apartan de lo esperable desde el punto de vista físico o de la simulación?
- **Variable objetivo y desbalance:** ¿cómo se distribuye la variable objetivo y qué tan marcado es el desbalance entre clases, de modo de anticipar impactos en muestreo, particionado y métricas de evaluación?
- **Calidad y codificaciones especiales:** ¿existen valores de relleno, faltantes implícitos o codificaciones no evidentes que deban tratarse explícitamente antes de realizar agregaciones, correlaciones o transformaciones masivas?

- **Redundancia y dependencias:** ¿qué relaciones (correlaciones) aparecen entre variables y métricas, y qué sugiere esto sobre redundancia, selección de *features* y riesgos de fuga de información?
- **Costo computacional:** ¿qué transformaciones resultan críticas en términos de cómputo/E/S y cómo condicionan la estrategia de materialización en Bronze–Silver–Gold y el entorno de ejecución (Sección 3.4)?

El alcance del EDA se concentra en dos líneas de evidencia complementarias. Por un lado, se realizan análisis ad-hoc en notebooks (agregaciones, verificaciones y visualizaciones orientadas a hipótesis). Por otro lado, se incorporan reportes automáticos generados con `ydata-profiling` (Clemente y cols., 2023), que aportan un resumen sistemático y reproducible de estadísticas del dataset, alertas de calidad, patrones de faltantes y relaciones entre variables.

Como entregables, esta sección se apoya en los siguientes artefactos:

- **Notebook EDA *Lightpath*:** consultas y visualizaciones enfocadas en dominios, distribuciones y consistencia a nivel de conexión.
- **Notebook EDA *Network Status*:** análisis de volumen/estructura, distribución de métricas, desbalance, y validación del costo de transformaciones necesarias para construir vistas de modelado.
- **Reporte *YData Profiling* (*lightpath*) y reporte *YData Profiling* (Network Status):** a partir de los cuales se incorporan capturas seleccionadas (*dataset statistics*, *alerts*, *missing values* y *correlations*) como evidencia directa de estructura y calidad del dato.

3.6.2. EDA para la representación *Lightpath*

En el marco de la estructura del dataset presentada en la Sección 3.2, *Lightpath* se interpreta como la representación más "compacta" del problema: cada registro describe un *lightpath* aprovisionado y condensa, en pocas variables, información de ruta, parámetros de transmisión y calidad observada. Esta simplicidad relativa la vuelve un punto de partida natural para el EDA: permite chequear dominios, rangos y relaciones generales con iteraciones rápidas, antes de abordar *Network Status*, donde el estado global de la red incrementa de forma marcada tanto el volumen como la complejidad estructural.

Análisis del dataset y variables relevantes

El análisis se realizó sobre un *preview* de la vista *Lightpath*, mediante consultas exploratorias en notebook y un reporte automático de *ydata-profiling*. Este último sintetiza, de forma reproducible, estadísticas descriptivas, alertas y relaciones relevantes entre variables, y sirve como evidencia empírica para justificar decisiones de limpieza y selección de *features*. En particular, el reporte permite validar, en una sola pasada, el tamaño del *preview*, los tipos inferidos y chequeos globales (completitud, cardinalidades y alertas), lo cual resulta útil

como *sanity check* previo a inspecciones más finas; la Figura 3.7 resume este panorama general.

En términos de estructura, *Lightpath* se comporta como un dataset tabular clásico: una fila por conexión, con variables mayormente numéricas y un conjunto pequeño de atributos categóricos. A grandes rasgos, las variables pueden agruparse en las siguientes familias:

- **Topología y ruta:** variables que capturan la distancia recorrida y la "forma" de la ruta (p.ej. `path_len`, `avg_link_len`, `min_link_len`, `max_link_len`, `num_links`, `num_spans`).
- **Espectro y configuración de transmisión:** parámetros asociados al canal y a la modulación (p.ej. `freq`, `mod_order`, `lp_linerate`, `conn_linerate`).
- **Calidad de transmisión (variables objetivo):** métricas continuas y su etiqueta discreta (p.ej. `y_osnr`, `y_snr`, `y_ber`, `y_class`).

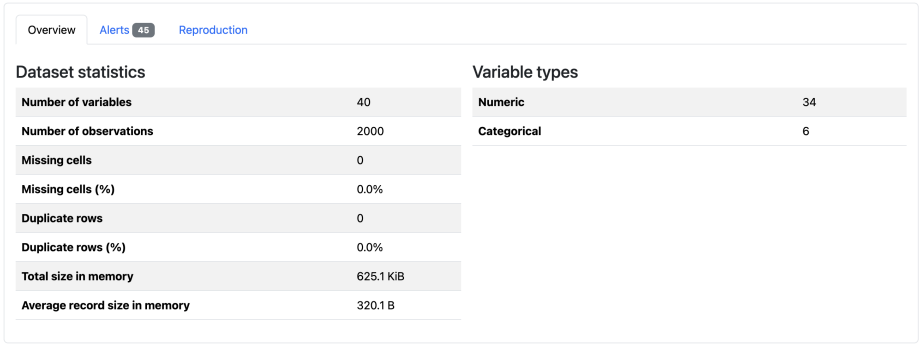


Figura 3.7: Resumen del *preview* *Lightpath* a partir del reporte automático (`ydata-profiling`): tamaño, tipos de variables y chequeos globales.

Distribuciones, rangos y validaciones de consistencia

Una vez verificada la estructura, el EDA se enfocó en contrastar rangos y distribuciones contra lo esperable desde el dominio. En la familia de ruta/topología, `path_len` presenta una dispersión marcada, con valores que cubren desde trayectos cortos hasta rutas largas; esta variabilidad es consistente con una topología donde no todas las conexiones enfrentan el mismo nivel de exigencia. La Figura 3.8 ilustra esta heterogeneidad y sugiere que el dataset incluye casos de distinta "dificultad" (en términos de distancia y degradaciones acumuladas), lo que luego se refleja en las variables de calidad.

De forma complementaria, `num_links` y `num_spans` ayudan a caracterizar la complejidad de la ruta más allá de la distancia total: en el *preview* tienden a moverse en conjunto con las variables de longitud, reforzando la idea de

que varias de estas variables capturan, en esencia, una misma señal estructural (longitud/segmentación de la ruta), con diferentes agregaciones.

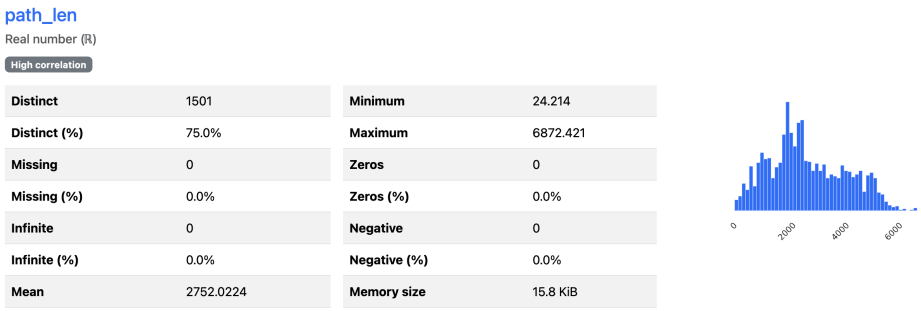


Figura 3.8: Distribución de `path_len` (longitud total de ruta) en el *preview Lightpath*.

En la configuración de transmisión, variables como `mod_order` y `lp_linerate` permiten corroborar que el dataset cubre un conjunto discreto de opciones de modulación y tasas, con frecuencias relativas diferentes entre categorías. La Figura 3.9 muestra estas distribuciones y confirma dos aspectos prácticos: (i) que las codificaciones son consistentes con un conjunto finito de configuraciones y (ii) que algunas combinaciones pueden estar subrepresentadas, lo cual anticipa la necesidad de cuidados metodológicos posteriores (por ejemplo, estratificación o validaciones que eviten que particiones pequeñas queden dominadas por pocos casos).

Finalmente, sobre las variables objetivo (`y_osnr`, `y_snr`, `y_ber`) se verificó que no existan valores faltantes en el *preview* y que sus rangos sean plausibles, en el sentido de no evidenciar valores fuera de dominio ni codificaciones inconsistentes. La Figura 3.10 resume estas distribuciones y, además, el conteo por clase de `y_class`, donde se observa tempranamente el **desbalance de clases**: existe una clase mayoritaria y una o más clases minoritarias con menor representación. Este hallazgo condiciona tanto la elección de métricas de evaluación (p.ej., métricas robustas a desbalance) como decisiones posteriores de particionado (p.ej., particiones estratificadas) y/o técnicas de balanceo si el objetivo es mejorar cobertura en las clases raras.

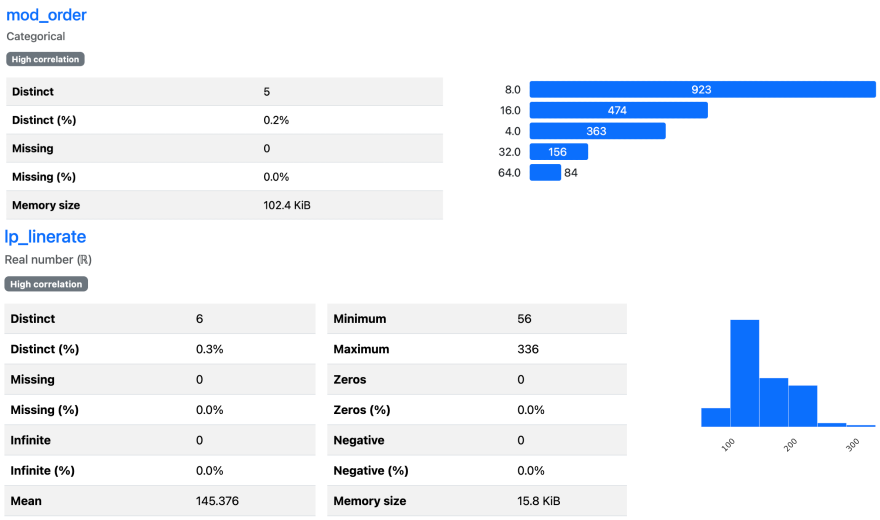


Figura 3.9: Distribuciones de variables de configuración: `mod_order` y `lp_linerate`.

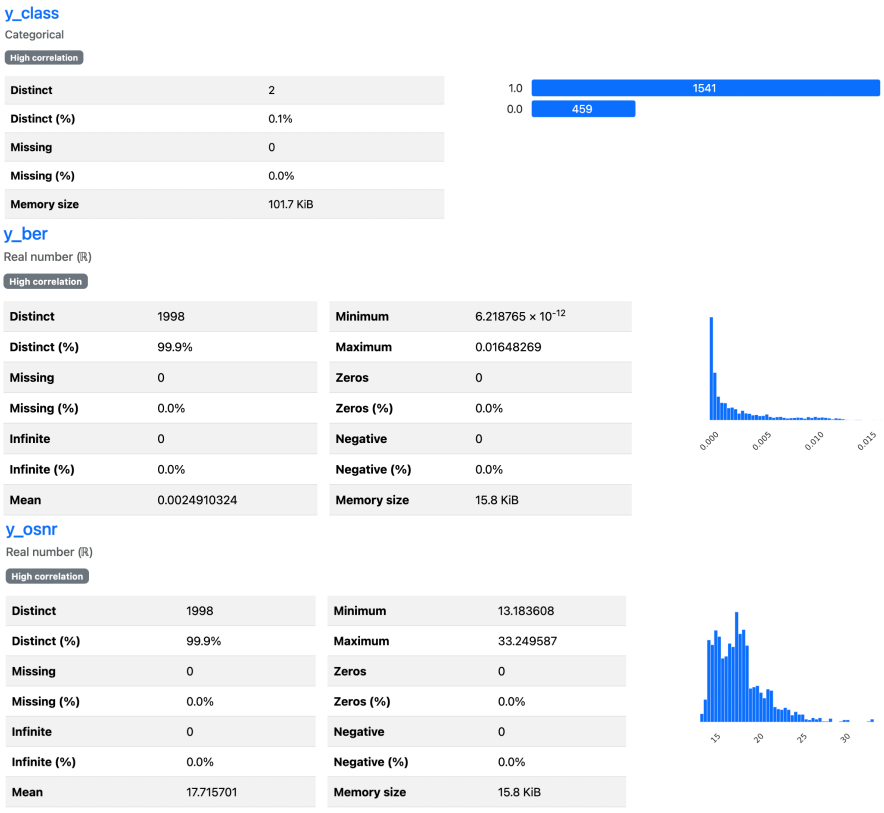


Figura 3.10: Variables objetivo en *Lightpath*: distribuciones de `y_osnr`/`y_ber` y desbalance observado en `y_class`.

Correlaciones y redundancias

Además de las distribuciones marginales, se inspeccionaron correlaciones para identificar redundancias estructurales; la Figura 3.11 (mapa de calor) sintetiza este análisis y permite detectar dependencias fuertes entre familias de variables. En *Lightpath* es esperable observar alta correlación entre variables derivadas de la ruta (p.ej. longitudes y conteos) y, en general, entre variables que capturan el mismo fenómeno con diferentes agregaciones (mínimo/máximo/promedio). En el *preview*, este patrón aparece de forma clara: las variables asociadas a topología/-ruta tienden a agruparse, y se observan relaciones fuertes entre agregaciones de un mismo atributo.

Este diagnóstico es particularmente útil para el *pipeline* hacia vistas *Gold*: aporta evidencia para (i) evitar sobre-representar señales duplicadas, (ii) simplificar el espacio de *features* sin perder información relevante y (iii) reducir complejidad en etapas posteriores de entrenamiento, especialmente cuando se exploran múltiples variantes de entrada. En términos prácticos, las correlaciones sugieren que, para ciertas familias, puede ser suficiente retener un subconjunto representativo (por ejemplo, una agregación por atributo o una variable proxy de complejidad de ruta), dejando explícita la decisión para mantener comparabilidad entre corridas.

En síntesis, el EDA de *Lightpath* cumple un rol "de calibración": valida rápidamente consistencias de codificación y rangos, caracteriza (aunque sea de forma preliminar) el desbalance en la variable objetivo discreta y aporta evidencia para decisiones de selección de variables. Esto permite trasladar el foco a *Network Status* con supuestos ya verificados en la representación más simple, donde las mismas preguntas deberán resolverse a escala y con un costo computacional sustancialmente mayor.

3.6.3. EDA para la representación *Network Status*

Como se detalló en la Sección 3.2 (estructura del dataset), la representación *Network Status* se distingue por capturar el **estado global de la red** en el instante de provisión, en contraste con vistas más compactas centradas en una conexión. En la práctica, esto la convierte en la vista más exigente en términos de volumen y estructura: la unidad de análisis deja de ser "un vector por conexión" y pasa a requerir una organización consistente del contexto de red para cada muestra. Esta diferencia no es meramente conceptual: condiciona qué entendemos por *observación*, qué transformaciones son necesarias para obtener un dataset consumible por modelos, y qué costos emergen al escalar el análisis.

Estructura, granularidad y volumen

En su forma nativa dentro del pipeline, *network status* se preserva inicialmente como una **vista larga** (*long format*) en la capa Bronze: cada fila corresponde a una combinación `(sample, link, freq, feat, metric)` con los valores crudos asociados. En este esquema, `sample` identifica la muestra (snapshot del estado global), `link` referencia un enlace de la topología y `freq` la grilla espectral;

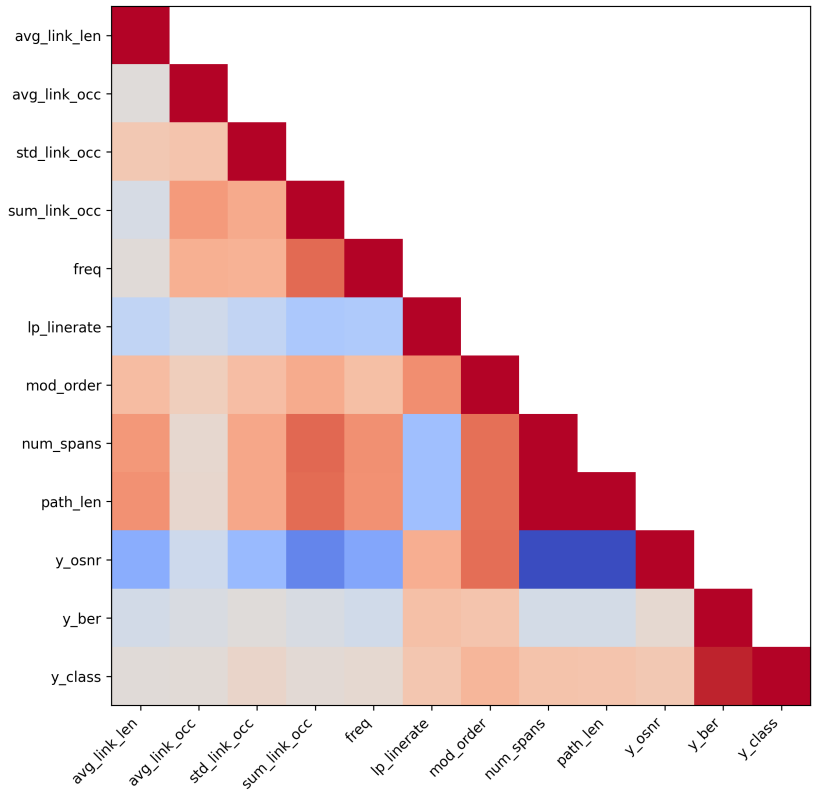


Figura 3.11: Mapa de correlaciones (reporte `ydata-profiling`) para *Lightpath*. Se observan relaciones fuertes entre variables asociadas a ruta/topología y entre distintas agregaciones de un mismo atributo.

mientras que `feat` puede corresponder a atributos del *lightpath* o descriptores topológicos/de red (familias `lp_*` y `nt_*`), y `metric` indica la métrica física u objetivo (p.ej., `osnr`, `snr`, `ber`, `class`). Esta decisión de diseño evita forzar prematuramente una estructura tabular "ancha", preservando el contenido del origen y manteniendo trazabilidad sobre cómo se construyen luego las variables de entrada.

Para poder aplicar EDA de manera operativa, el análisis se apoya en la materialización Silver, donde se realiza la transformación estructural **long → wide** cuando aplica, con una granularidad consistente, **una fila por** `(sample, link, freq)`. En esta vista, las columnas numéricas agrupan: (i) las métricas físicas y la etiqueta (por ejemplo `m_osnr`, `m_snr`, `m_ber`, `m_class`), y (ii) el contexto de red/topología y atributos del *lightpath* (familias `nt_*` y `lp_*`, respectivamente). Esta "tabla base" es la que habilita estadísticos descriptivos, chequeos de dominio y correlaciones sin reintroducir complejidad estructural en cada consulta.

Como evidencia empírica mínima de estructura y escala *a nivel de tabla*, se generó un reporte automático de `ydata-profiling` sobre una **muestra estratificada** de *network status*. En dicha muestra se observan **21 variables** y **190080 observaciones**, sin celdas faltantes, lo que permite ejecutar exploraciones interactivas (notebook + profiling) sin perder representatividad básica para validaciones iniciales. En contraste, el dataset completo de *network status* impone costos sustancialmente mayores: tanto por su tamaño como por la necesidad de consolidar información (reordenamientos y materializaciones) antes de obtener vistas "consumibles". Por ello, el EDA se plantea explícitamente como un proceso que combina (i) lecturas selectivas sobre materializaciones columnar/Parquet y (ii) muestras controladas para inspección rápida, reservando el procesamiento exhaustivo para ejecución por lotes con recursos dimensionados.

Distribución de métricas y atributos; proporción de clases

Como primer chequeo de consistencia, se realizó un conteo de ocurrencias por tipo de métrica (`metric`) y por atributo topológico (`nt_feat`) sobre el *preview*. La motivación es simple: en datasets masivos, un desbalance inesperado en estos conteos suele ser un primer síntoma de ingesta incompleta, particionado defectuoso o filtros aplicados de forma inadvertida. La Tabla 3.4 resume este control básico, mostrando conteos uniformes tanto por métrica como por atributo topológico en el *preview*.

La uniformidad en los conteos es, en buena medida, un resultado **esperable** dado el proceso de simulación y exportación descrito en la Sección 3.3: para cada muestra se materializa el mismo conjunto de métricas principales y de atributos topológicos, de modo que el *preview* debería reflejar ocurrencias comparables por `metric` y `nt_feat`. Más que "probar" cómo se construyó el dataset, este chequeo funciona como *sanity check* de la ingesta: en volúmenes masivos, un desbalance inesperado en estos conteos suele ser la primera señal de particionado incompleto, lecturas truncadas o filtros aplicados de forma inadvertida, antes de avanzar hacia análisis más costosos (p. ej., agregaciones globales o transformaciones estructurales).

Tabla 3.4: Distribución de registros por tipo de métrica y atributo topológico en el *preview* de *network status*.

Métrica (`metric`)	**Cantidad de registros**
`snr`	228.096.000
`osnr`	228.096.000
`ber`	228.096.000
`class`	228.096.000

Atributo de red (`nt_feat`)	**Cantidad de registros**
`link_len`	228.096.000
`num_spans`	228.096.000
`src_degree`	228.096.000
`dst_degree`	228.096.000

Tabla 3.5: Distribución de clases QoT en el *preview* de *network status*.

Clase	**Cantidad**	**Porcentaje**
1 (QoT válida)	523.480.320	74.88 %
0 (QoT no válida)	175.747.968	25.12 %

Finalmente, se inspeccionó la distribución de `class`, que funciona como etiqueta binaria para indicar si la QoT de una conexión es válida (1) o no válida (0). Los resultados se resumen en la Tabla 3.5, que permite cuantificar tempranamente el nivel de desbalance en la etiqueta.

La distribución indica un **desbalance moderado** hacia la clase positiva. Esto no invalida el dataset, pero sí condiciona la etapa de modelado: (i) conviene preservar el desbalance real mediante particiones y muestreos **estratificados**, y (ii) es importante no depender únicamente de *accuracy* como referencia, sino complementar con métricas más informativas ante desbalance (p. ej., *precision/recall*, F1 y curvas/áreas tipo AUC-PR), o bien incorporar ajuste de pesos si el algoritmo lo permite. En conjunto, estos chequeos iniciales dejan planteado desde el EDA qué aspectos deben controlarse para evitar conclusiones sesgadas en la evaluación posterior.

Calidad del dato: rangos no físicos y valores de relleno

Un punto relevante del control de calidad es que los "problemas" no aparecen necesariamente como *missing values*. En el perfil automático generado con `ydata-profiling` sobre una muestra estratificada, el reporte indica **0 % de celdas faltantes** pero al mismo tiempo levanta múltiples alertas. La Figura 3.12 resume este diagnóstico y motiva chequeos explícitos de dominios y de valores especiales antes de aplicar transformaciones globales.

Para inspeccionar este fenómeno de forma más controlada, se calcularon estadísticas descriptivas básicas agrupando por `metric` y `nt_feat`. Allí se observa-

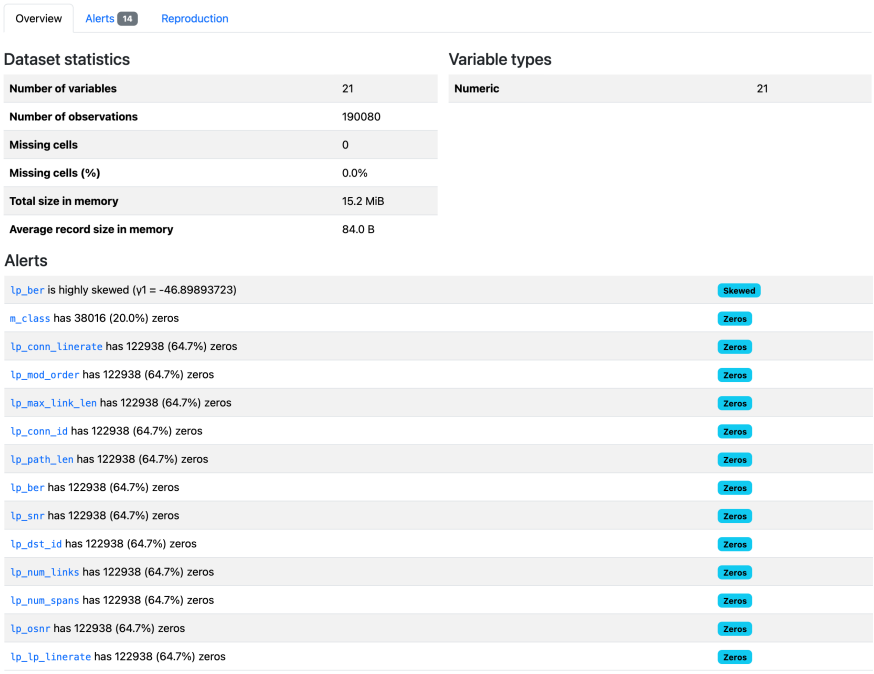


Figura 3.12: Vista general del perfil `ydata-profiling` para una muestra estratificada de *Network Status*. Aunque no se observan celdas faltantes, el reporte indica alertas que motivan chequeos de dominios y valores especiales.

ron extremos no consistentes con rangos físicos esperables, con valores recurrentes como `-1.0` y `99984.0`. En la práctica, estos extremos aparecen concentrados y no se comportan como *outliers* naturales, sino como **valores de relleno** (o codificaciones de exportación) que deben tratarse como una regla explícita de calidad.

Al bajar al detalle por variable, el reporte deja ver patrones que difícilmente correspondan a variación "natural" del fenómeno: en métricas de QoT del *lightpath* (p. ej., `lp_ber`, `lp_snr`, `lp_osnr`) aparecen valores extremos repetidos como `-1` e incluso algunos valores negativos. Más que interpretarlos como outliers, estos casos se toman como indicios de codificación especial que debe tratarse explícitamente. En la misma línea, varias variables presentan una acumulación marcada en cero; según el campo, esto puede ser perfectamente válido, pero también puede reflejar valores de relleno o defaults del generador, por lo que conviene auditarlo antes de entrenar modelos o calcular estadísticas globales. La Figura 3.13 ilustra este comportamiento para un subconjunto representativo de métricas.

En términos de ingeniería, este hallazgo se traduce en **reglas de limpieza y validación** que deben ejecutarse temprano en el *pipeline* (en la capa Silver), antes de operaciones que mezclan o redistribuyen masivamente el dato. En

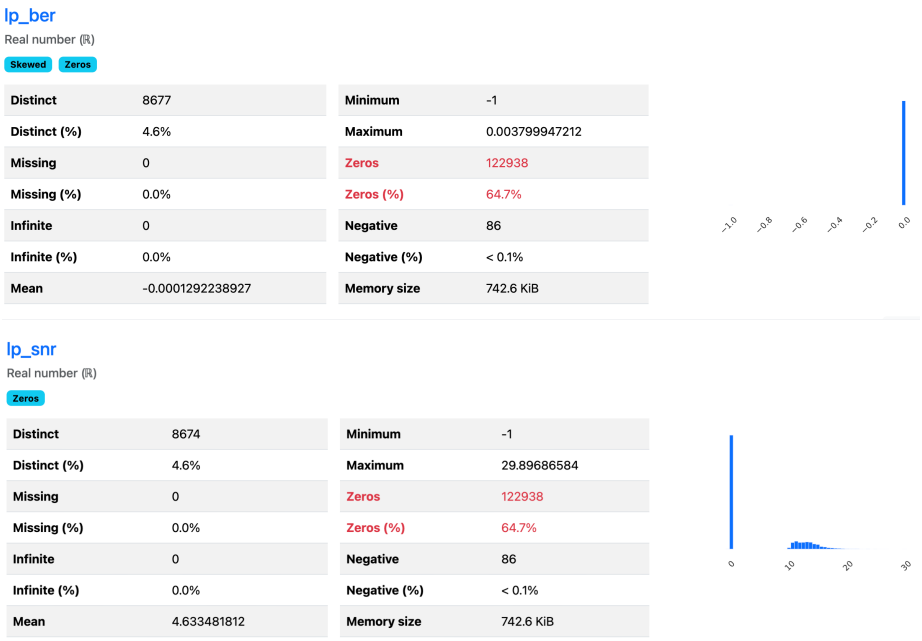


Figura 3.13: Ejemplo de *valores de relleno* / codificación especial en métricas de QoT (muestra estratificada). En `lp_ber` y `lp_snr` se observan mínimos en `-1`, presencia minoritaria de valores negativos y una acumulación marcada en cero.

particular:

- **Normalización de valores de relleno**: mapear valores como `-1` y `99984` a `null`/`NaN` (o filtrar registros según corresponda), de modo de no contaminar promedios, correlaciones y agregaciones.
- **Chequeos de dominio**: imponer restricciones simples (p. ej., no-negatividad en métricas de QoT cuando aplique) y auditar cualquier violación como parte del control de calidad.
- **Separación de etapas**: aplicar la normalización de valores de relleno **antes** de cualquier transformación de *reordenamiento* o *consolidación* a gran escala, ya que luego se vuelven difíciles de rastrear y pueden sesgar tanto estadísticos como entrenamiento.

3.6.4. Implicancias para el pipeline y próximos pasos

En conjunto, el EDA cumplió un rol doble. Por un lado, permitió entender la unidad real de análisis y el "costo estructural" de trabajar con *Network Status* (formato largo y contexto global), en contraste con la representación *Lightpath*

(tabular y por conexión). Por otro, funcionó como una etapa de ingeniería de calidad y viabilidad: varios hallazgos no se interpretan como "ruido estadístico", sino como reglas operativas que deben quedar incorporadas al procesamiento para que los resultados posteriores sean consistentes y comparables.

Dado el proceso de generación descrito en la Sección 3.3, la presencia de valores especiales (p. ej., codificaciones fuera de rango o concentraciones en valores particulares) debe tratarse de forma explícita: si estos patrones se arrastran a etapas posteriores, pasan a mezclarse con transformaciones masivas y luego se vuelven difíciles de rastrear, afectando tanto estadísticos globales como el entrenamiento. En ese sentido, el EDA no sólo "describe" el dato, sino que también define un contrato mínimo de limpieza, validación y consistencia que el pipeline debe garantizar.

Reglas que nacen del EDA (calidad y dominios). A partir de los chequeos realizados, se consolidan criterios prácticos para la capa Silver: (i) validación de dominios por variable (rangos físicos plausibles y exclusión de codificaciones no físicas), (ii) tratamiento uniforme de valores especiales (normalización, filtrado o recodificación según corresponda), (iii) controles de consistencia por representación para evitar que el mismo fenómeno se codifique de forma distinta entre vistas. Estas reglas se ejecutan *antes* de cualquier reestructuración o agregación global, para evitar propagar artefactos a vistas "listas para modelar".

Justificación de materializaciones (Bronze–Silver–Gold). Los resultados del EDA confirman que el costo dominante en *Network Status* no proviene de "calcular una estadística", sino del **movimiento y reorganización de datos** necesario para construir entradas consumibles por modelos (vistas tabulares y/o estructuradas). Por ello, la estrategia *Bronze–Silver–Gold* (Sección 3.5) se vuelve una decisión metodológica central: desacoplar el costo de transformación (una vez, en *Silver/Gold*) del ciclo iterativo de entrenamiento y análisis (múltiples corridas). En la práctica, la capa *Silver* estabiliza esquema y reglas de limpieza; la capa *Gold* materializa vistas orientadas a modelado, distintas según representación y familia de modelos.

Ejecución a escala: de exploración interactiva a modo batch distribuido. Otra implicancia directa es operativa: la fase exploratoria sirve para validar hipótesis y reglas en subconjuntos, pero los resultados deben replicarse en el dataset completo bajo un modo de ejecución controlado (por lotes) y con recursos dimensionados. Esto conecta con la separación de responsabilidades entre entornos discutida en la Sección 3.4: la VM facilita iteración y depuración; la infraestructura distribuida se reserva para etapas donde predominan redistribuciones y reestructuraciones a gran escala.

Puente hacia la experimentación. Finalmente, este cierre del EDA deja preparado el terreno para el Capítulo 4: al fijar reglas de calidad, dominios y vistas reutilizables, se garantiza que la experimentación compare modelos bajo condiciones equivalentes. En particular, las representaciones no sólo condicionan qué modelos son viables, sino también qué tipo de artefacto debe entregar la capa *Gold*: para *Lightpath*, una vista tabular directa; para *Network Status*, vistas

tabulares y/o estructuradas que preserven contexto, manteniendo un protocolo de evaluación comparable.

En síntesis, el EDA guía decisiones de modelado, pero sobre todo reduce incertidumbre de ingeniería: explicita qué limpiar, qué materializar y dónde ejecutar cada etapa, de forma que el resto del trabajo se apoye en salidas estables, trazables y repetibles.

Capítulo 4

Diseño y metodología experimental

Este capítulo presenta el diseño de la etapa de experimentación sobre los conjuntos de datos ya procesados, con foco en: (i) definir un protocolo de entrenamiento y evaluación consistente, (ii) establecer un marco de comparación reproducible entre modelos y representaciones, y (iii) especificar los criterios y artefactos necesarios para interpretar los resultados de forma operativa.

En línea con el énfasis metodológico del trabajo, las decisiones de ingeniería de datos (ingesta, normalización y generación de vistas reutilizables) se documentan en el Capítulo 3.5. A partir de esa base, aquí se asume la disponibilidad de artefactos *Gold* y se describe cómo se consumen para entrenar, validar y analizar modelos, desacoplando el costo de transformación del ciclo iterativo de modelado.

La estructura del capítulo es la siguiente: primero se resumen las tareas y la matriz de experimentos (problemas, modelos y comparaciones), luego se justifica el alcance del dataset utilizado, se explicita la metodología experimental del pipeline de aprendizaje automático, y se discuten consideraciones de infraestructura y costo computacional. Los resultados y su discusión, organizados por representación, se presentan en el capítulo siguiente.

4.1. Problemas abordados y matriz de experimentos

4.1.1. Tareas supervisadas y objetivos

El objetivo general de esta etapa es evaluar la viabilidad de utilizar aprendizaje supervisado para predecir indicadores de desempeño de una red óptica elástica a partir de datos generados por simulación.

Tareas supervisadas. En este informe se abordan las siguientes tareas sobre ambas representaciones del dataset:

- **Clasificación de QoT**: predecir si una conexión (*lightpath*) cumple (o no) un umbral operativo de calidad de transmisión. Esta tarea refleja un escenario de *admisión* o *aceptación* de conexiones, donde los errores tienen implicancias operativas (p. ej., falsos positivos que aceptarían conexiones inviables versus falsos negativos que rechazarían conexiones potencialmente válidas).

- **Regresión de métricas físicas**: estimar `OSNR`/`SNR` y `BER` como variables continuas. Estas métricas aportan una señal cuantitativa para diagnóstico y validación, ya que permiten analizar no sólo si una conexión es "viable" sino *cuán lejos* está del umbral y cómo se distribuyen los errores bajo distintas condiciones de ruta y carga.

Objetivos específicos. A partir de estas tareas, los objetivos concretos de la experimentación son:

1. **Establecer líneas base reproducibles** para clasificación (QoT) y regresión (`OSNR`/`SNR`, `BER`) en ambas representaciones, utilizando modelos de complejidad creciente (tabulares y neuronales) como punto de comparación con trabajos previos (Olmedo Guillama, 2024).

2. **Cuantificar desempeño y generalización** mediante un protocolo de particionado y métricas estándar, controlando la fuga de información (*leakage*) y preservando consistencia experimental entre modelos.

3. **Analizar resultados con foco práctico** (no sólo reportar métricas): identificar qué escenarios/casos límite degradan el rendimiento, y qué atributos o regiones de la entrada explican las predicciones (interpretabilidad/sensibilidad), con el fin de evaluar factibilidad de uso posterior.

4. **Consolidar el pipeline como entregable**: entrenar y evaluar consumiendo representaciones precomputadas (*Gold*) para evitar recomputar transformaciones costosas en cada corrida. Este objetivo conecta directamente con el énfasis del proyecto en ingeniería de datos y costo computacional.

Relación con el alcance del proyecto. Dado que el costo de cómputo es un factor limitante y el foco del proyecto es el pipeline de datos, se prioriza consolidar una metodología sólida sobre el dataset principal (dataset 1) y dejar la evaluación en datasets adicionales como extensión de robustez. Esta decisión permite maximizar trazabilidad y reproducibilidad sin comprometer la validez del análisis experimental.

4.2. Modelos considerados y representaciones de entrada

Para abordar la predicción de QoT se seleccionaron modelos de distinta complejidad que cubren dos formas de presentar la información al algoritmo: (i) una representación *tabular*, donde cada instancia se describe con un vector de variables, y (ii) una representación *estructurada* (grilla/tensor), donde se preserva explícitamente una noción de vecindad (por ejemplo, canales contiguos en frecuencia o contexto local del espectro). Esta separación permite evaluar, en condiciones comparables, cuánto del rendimiento proviene de la capacidad del modelo y cuánto de la representación empleada.

En todos los casos el objetivo es estimar si una instancia (una solicitud o un *lightpath* propuesto, según el *dataset*) cumple QoT. En formulación de clasificación, la salida se interpreta como una probabilidad de aceptación/rechazo; en formulación de regresión, como una estimación continua de una métrica (p. ej., OSNR o Q-factor) que luego puede compararse contra un umbral operacional. La elección concreta depende del experimento y de las variables disponibles.

4.2.1. Modelos sobre representación tabular

En la representación tabular, cada instancia se resume en un vector de dimensión fija con variables que capturan (según disponibilidad del *dataset*) información de ruta/topología, configuración de transmisión, asignación espectral y/o estado agregado de la red. Esta vista habilita baselines sólidos con bajo costo computacional y permite entrenar de forma eficiente sobre grandes volúmenes de datos (Ayassi y cols., 2022).

Random Forest (ensamble de árboles de decisión)

Random Forest combina muchos árboles de decisión (Breiman, 2001). Un árbol de decisión organiza la predicción como una secuencia de particiones del espacio de variables: en cada nodo se evalúa una condición simple (por ejemplo, comparar una variable contra un umbral) y se continúa por una rama hasta llegar a una hoja que emite la clase o el valor estimado. Esto permite construir reglas no lineales a partir de comparaciones sencillas.

El inconveniente de un único árbol es su sensibilidad a variaciones en los datos de entrenamiento. Random Forest reduce este efecto entrenando múltiples árboles sobre distintas muestras (y con subconjuntos aleatorios de variables) y agregando sus salidas mediante votación o promedio. Además de su robustez frente a heterogeneidad de atributos y su bajo requerimiento de preprocesamiento, provee una primera aproximación de interpretabilidad mediante importancias de variables, útil para analizar qué grupos de *features* aportan señal predictiva (Breiman, 2001). En este trabajo se emplea como baseline principal en entrada tabular, tanto para clasificación de QoT como, cuando corresponde, para regresión de métricas físicas; en escenarios desbalanceados es posible

incorporar ponderación de clases para no favorecer sistemáticamente la clase mayoritaria.

MLP (Multilayer Perceptron)

El perceptrón multicapa (MLP) es una red neuronal alimentada hacia adelante (*feed-forward*) que transforma un vector tabular mediante capas densas y funciones no lineales (Rumelhart, Hinton, y Williams, 1986). En cada capa, la red combina las variables de entrada con pesos aprendidos y aplica una activación (por ejemplo, ReLU), generando una nueva representación interna de la instancia. La salida final puede ser una probabilidad (clasificación) o un valor continuo (regresión).

La diferencia clave frente a un modelo lineal es que el MLP permite que el efecto de una variable dependa del valor de otras. Por ejemplo, puede aprender que el alcance tolerable cambia según la modulación y que esa relación además se modifica bajo determinadas condiciones del entorno espectral. El entrenamiento se realiza mediante retropropagación de errores y optimización diferenciable, como se formaliza clásicamente en (Rumelhart y cols., 1986). En la comparación, el MLP cumple el rol de contraparte neuronal directa de los baselines tabulares: utiliza esencialmente la misma información de entrada que RF, pero con una familia de modelo distinta y mayor sensibilidad al preprocesamiento (p. ej., escalado/normalización) y a hiperparámetros. Esto permite evaluar si una arquitectura diferenciable aporta mejoras sobre el baseline tabular bajo igual contenido de información.

RNN (LSTM/GRU) para explotar estructura secuencial

Las redes recurrentes (RNN) están diseñadas para procesar entradas con estructura secuencial. A diferencia de un MLP, incorporan un estado interno que se actualiza al recorrer la secuencia, permitiendo que la predicción refleje información acumulada a lo largo de toda la serie. En la práctica se emplean variantes como Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU) para mejorar la estabilidad del entrenamiento y capturar dependencias de mayor alcance temporal/posicional (Hochreiter y Schmidhuber, 1997; Cho y cols., 2014).

En este trabajo, la motivación de incorporar una RNN es introducir un sesgo inductivo secuencial cuando la entrada puede organizarse naturalmente como serie ordenada: por ejemplo, un perfil por salto/enlace a lo largo de una ruta, o un contexto en frecuencia alrededor de una asignación. En tales casos, el modelo puede aprender patrones de acumulación progresiva o transiciones locales sin colapsar toda la información en estadísticas agregadas. En la comparación experimental, su rol es contrastar explícitamente contra enfoques que procesan atributos como un vector sin estructura: se evalúa si capturar orden y dependencia local aporta mejoras respecto a MLP/RF, y a qué costo computacional.

4.2.2. Modelos sobre representación estructurada

Además de la vista tabular, se considera una representación estructurada que preserva vecindades relevantes. La motivación es que ciertos efectos que degradan QoT se manifiestan como patrones locales: por ejemplo, en el dominio espectral, los canales adyacentes pueden influir más que los lejanos; o ciertas configuraciones pueden ser viables salvo cuando el entorno inmediato está altamente cargado. Al transformar el estado en una grilla/tensor, se habilita el uso de modelos que aprovechan explícitamente esa estructura.

CNN (modelo convolucional tipo *DCNN*)

Las redes convolucionales (CNN) detectan patrones locales reutilizando los mismos filtros en distintas posiciones de la entrada (Safari, Shariati, Bergk, y Fischer, 2021b). En vez de conectar cada variable con todas las demás, una CNN aplica filtros pequeños (kernels) que recorren la grilla o tensor y producen mapas de características, donde cada activación resume información de una vecindad. Al apilar varias capas, el modelo combina patrones simples en representaciones más abstractas.

Para sintetizar el propósito de cada familia de modelos dentro del diseño experimental, la Tabla 4.1 resume la entrada utilizada y el rol que cumple cada uno en la comparación.

Tabla 4.1: Rol de cada modelo en la comparación experimental.

Modelo	**Entrada**	**Rol en la comparación**
Random Forest	Tabular	Baseline robusto y escalable; referencia principal para evaluar señal predictiva, con soporte de no linealidades e importancias de variables (Breiman, 2001).
MLP	Tabular	Comparación neuronal directa sobre el mismo X; permite observar si una arquitectura diferenciable mejora al baseline tabular bajo igual información de entrada (Rumelhart y cols., 1986).
RNN (LSTM/GRU)	Secuencial	Modelo con sesgo inductivo secuencial; evalúa si capturar orden y dependencias locales aporta sobre el enfoque puramente tabular (Hochreiter y Schmidhuber, 1997; Cho y cols., 2014).
CNN (*DCNN*)	Grilla/Tensor	Modelo orientado a estructura local; evalúa el beneficio de preservar cercanía y patrones locales en la entrada frente a baselines tabulares (Safari y cols., 2021b).

4.2.3. Matriz de experimentos

Esta subsección resume de forma operacional *qué se corrió* durante la etapa experimental. La Tabla 4.2 consolida la matriz de experimentos y permite: (i) verificar cobertura de tareas por representación, (ii) documentar comparaciones directas entre familias de modelos y (iii) servir como índice para las secciones de resultados posteriores.

Tabla 4.2: Matriz de experimentos: representación, unidad de análisis, formato de entrada, tarea/objetivo, modelos y artefactos generados.

Representación	**Unidad**	**Entrada**	**Tarea / objetivo**	**Modelos**	**Artefactos**
Lightpath data	Conexión (*lightpath*)	Tabular (Gold-QoT)	Clasificación (QoT)	RF, MLP	modelo + métricas + curvas ROC/PR + predicciones
Lightpath data	Conexión (*lightpath*)	Tabular (Gold-Reg)	Regresión (OSNR/SNR)	RF Regressor	modelo + métricas + pred vs real + residuales
Lightpath data	Conexión (*lightpath*)	Tabular (Gold-Reg)	Regresión (BER)	RF Regressor	modelo + métricas + cuantiles de error + residuales
Network status	Snapshot de red	Tabular (Gold-NS)	Clasificación (QoT)	RF, MLP	modelo + métricas + curvas ROC/PR + predicciones
Network status	Snapshot de red	Secuencial (Gold-NS secuencia)	Clasificación (QoT)	RNN (LSTM/GRU)	modelo + métricas + learning curves + predicciones
Network status	Snapshot de red	Tensor (DCNN)	Clasificación (QoT)	CNN (DCNN)	modelo + métricas + learning curves + checkpoint

Tamaño de los conjuntos y particionado (placeholder). A modo de referencia, la Tabla 4.3 documenta el tamaño de los conjuntos utilizados y su particionado.

Tabla 4.3: Tamaños de dataset por experimento (QoT Dataset 01).

Experimento	**Train**	**Val**	**Test**
Lightpath QoT (tabular)	1,219,804	152,476	152,475
Lightpath OSNR/SNR (tabular)	1,219,804	152,476	152,475
Lightpath BER (tabular)	1,219,804	152,476	152,475
Network status QoT (tabular)	1,219,804	152,476	152,475
Network status QoT (secuencial)	1,219,804	152,476	152,475

Observaciones. (i) En regresión se entrenan modelos separados por objetivo (`OSNR/SNR` y `BER`) para facilitar interpretación y análisis de error. (ii) Para Network Status, se reportan resultados sobre una entrada tabular (baselines RF/MLP) y una entrada secuencial derivada (RNN) con el fin de evaluar si incorporar estructura ordenada aporta sobre el enfoque tabular, manteniendo comparable el protocolo de evaluación.

4.3. Alcance del dataset y criterios de selección

4.3.1. Dataset utilizado y justificación del alcance

La experimentación se realizó principalmente sobre el **dataset 1**, dado que es el conjunto más completo disponible y el que mejor equilibra cobertura de escenarios con costo computacional. En el contexto de este proyecto, el foco está puesto en consolidar un **pipeline de datos reproducible** (con trazabilidad extremo a extremo y artefactos persistidos) y, sobre esa base, evaluar modelos de aprendizaje automático como prueba de aplicabilidad. Esta priorización responde a las restricciones prácticas de hardware/tiempo y a la necesidad de dedicar esfuerzo a la metodología y su documentación, que constituyen el núcleo del proyecto.

Razonamiento detrás del alcance. El dataset se provee en múltiples variantes (datasets 1–6) que difieren en volumen y/o condiciones experimentales (p. ej., cantidad de escenarios, niveles de carga y combinaciones de configuración), y que fueron definidas por los autores como subconjuntos/variantes para analizar el problema bajo distintos presupuestos de cómputo. El origen del dataset, su proceso de generación y el contexto en el que se definen estas variantes se describen en el Capítulo 3.1. En principio, la metodología propuesta es *agnóstica* al dataset específico: el mismo flujo Bronze→Silver→Gold y el mismo protocolo de entrenamiento/evaluación pueden aplicarse a cualquiera de las variantes sin cambios conceptuales. Sin embargo, ejecutar de forma sistemática sobre todas las variantes implica un costo significativo de cómputo, especialmente en las etapas de generación de artefactos (Silver/Gold) y en las corridas repetidas de entrenamiento y evaluación.

Decisión de alcance. Por los motivos anteriores, se adopta el siguiente criterio:

- **Foco principal: dataset 1.** Se utiliza como base para reportar resultados y análisis, ya que ofrece la mayor representatividad para validar el pipeline y establecer líneas base.
- **Extensión / robustez (datasets 2–6):** se plantean como trabajo futuro o evaluación complementaria, orientada a medir robustez y transferibilidad del enfoque bajo otras condiciones, una vez estabilizadas las decisiones de ingeniería y el protocolo experimental.

Implicancias para la interpretación de resultados. Este alcance implica que los resultados presentados deben entenderse como una validación sólida de factibilidad y de metodología sobre el dataset principal. La evaluación cruzada sobre datasets adicionales permitiría profundizar en generalización y sensibilidad a condiciones de simulación, pero excede el presupuesto computacional razonable para el alcance de este trabajo.

4.3.2. Representaciones consideradas y su implicancia en el modelado

El dataset provee dos representaciones complementarias del mismo fenómeno, que inducen elecciones distintas de modelado. Mientras *Lightpath* describe cada conexión de forma individual (en formato tabular), *Network Status* describe el estado global de la red bajo una configuración de tráfico (en formato estructurado). Esta dualidad es central para el diseño experimental, ya que permite evaluar tanto enfoques *per-connection* como enfoques condicionados por el contexto global.

Lightpath data (per-connection, tabular). En esta representación, cada ejemplo corresponde a una conexión óptica individual y se codifica como una fila tabular con atributos derivados de ruta/configuración y acumulados físicos. Su formato es directamente compatible con modelos clásicos para datos tabulares (p. ej., Random Forest y MLP), facilitando establecer líneas base robustas con bajo costo de ingeniería. Además, la granularidad por conexión habilita tareas tanto de clasificación (QoT) como de regresión de métricas físicas (`OSNR`/`SNR`, `BER`).

Network status (estado global, snapshot/bucket). En esta representación, cada ejemplo corresponde a una "foto" del estado global de la red bajo una asignación de espectro y carga determinada. Esta vista agrega información contextual (ocupación/utilización/propiedades por enlace y por slot) y permite estudiar predicción de QoT condicionada al estado global. Desde el punto de vista del modelado, este formato puede consumirse de múltiples maneras: (i) como entrada tabular mediante agregaciones/estadísticos globales (baselines RF/MLP) o (ii) preservando una organización ordenada (p. ej., secuencial) para modelos con sesgo inductivo estructurado (RNN).

Implicancia principal en el pipeline. Las representaciones definen no sólo el tipo de modelo posible sino también el tipo de artefacto que debe producirse en la capa *Gold*: para *Lightpath*, una vista tabular lista para entrenamiento; para *Network Status*, vistas orientadas a consumo tabular y/o estructurado (secuencial), manteniendo un protocolo de evaluación comparable. Esta separación refuerza el objetivo del proyecto: desacoplar el costo de transformación de datos del ciclo iterativo de entrenamiento y análisis.

4.4. Metodología experimental: pipeline de aprendizaje automático

El pipeline de datos (*Bronze–Silver–Gold*) y la visión end-to-end de la solución se presentaron en el Capítulo 3.5. Para facilitar la lectura, recuperamos aquí la Figura 3.5, que resume el flujo completo y ubica al lector en qué punto del proceso se inserta la etapa experimental. En esta sección nos concentramos en el **pipeline de aprendizaje automático**: cómo se consumen las vistas *Gold*, qué protocolo de particionado y evaluación se aplica, y qué artefactos se persisten para garantizar comparabilidad y reproducibilidad entre corridas.

Para complementar la vista end-to-end, la Figura 4.1 muestra un *zoom-in* del pipeline de Machine Learning, enfatizando las tres etapas que estructuran la experimentación: selección de modelos, entrenamiento y evaluación (medición de performance, análisis de errores y comparación entre enfoques).

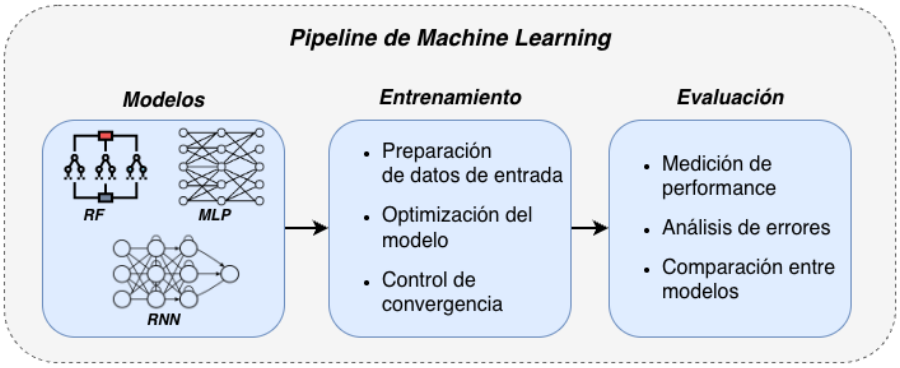


Figura 4.1: Zoom-in del pipeline de Machine Learning utilizado en la etapa experimental: modelos considerados, entrenamiento y evaluación.

4.4.1. Pipeline de aprendizaje automático

A partir de las vistas *Gold* generadas por el pipeline de datos se define el **pipeline de aprendizaje automático**, que organiza de forma uniforme cómo se entrenan y evalúan los modelos. La idea central es simple: mantener constante la entrada (la vista Gold de cada experimento) y variar únicamente lo que se quiere comparar (modelo, hiperparámetros y formato de entrada cuando aplica). De esta forma, las diferencias que se observan en desempeño pueden interpretarse con mayor confianza como un efecto del enfoque de modelado, y no como consecuencia de cambios en el preprocesamiento o en el particionado.

El flujo comienza con la **carga de los datasets Gold** correspondientes a cada experimento (por representación y por tarea). Luego se aplica el particionado train/validation/test definido para el estudio, cuidando dos aspectos prácticos:

(i) mantener el mismo criterio en todas las corridas para que los resultados sean comparables, y (ii) evitar fuga de información (*leakage*) entre conjuntos, especialmente en variables que podrían estar demasiado cerca de la definición operacional del objetivo. En cada corrida también se fijan parámetros básicos de reproducibilidad para que el experimento pueda repetirse.

A continuación se realiza el preprocesamiento mínimo requerido por cada familia de modelos. En baselines tabulares como Random Forest, el preprocesamiento es reducido; en modelos neuronales (MLP/RNN) se incorpora normalización/escalado y se construye el formato de entrada correspondiente cuando la representación se organiza como secuencia. En todos los casos se busca que estas transformaciones sean las estrictamente necesarias y estén bien delimitadas, para no introducir variaciones adicionales entre experimentos. Con los datos preparados se entrena el modelo seleccionado (RF, MLP o RNN, según corresponda) y se registran los parámetros usados.

La evaluación se estructura según el tipo de tarea. Para **clasificación de QoT** se reportan métricas y visualizaciones que permiten leer el comportamiento operativo del clasificador: matriz de confusión, *Accuracy*, *F1* y curvas ROC/PR. Para **regresión** de métricas físicas (`OSNR`/`SNR`, `BER`) se reportan *MAE*, *RMSE* y R^2, complementadas con diagnósticos simples (predicción vs. real y residuales) que ayudan a detectar sesgos o rangos donde el error se concentra. Finalmente, se incluye una etapa de análisis orientada a interpretar resultados: comparación directa entre modelos bajo las mismas condiciones y revisión de casos límite o patrones de error.

Un punto clave para que este proceso sea útil en la práctica es que cada corrida produce **artefactos persistidos**: configuración y semillas, métricas, predicciones sobre test y, cuando aplica, curvas de aprendizaje o checkpoints. Esto permite revisar, comparar y discutir resultados sin reentrenar todo desde cero. El capítulo siguiente utiliza precisamente estos artefactos para presentar los **Resultados de la experimentación**: primero se reporta el desempeño por representación y por familia de modelos, y luego se discuten los errores e interpretaciones más relevantes bajo un mismo protocolo experimental.

Capítulo 5

Resultados de la experimentación

Este capítulo presenta los resultados obtenidos al aplicar el protocolo experimental definido en el Capítulo 4 sobre los artefactos *Gold*. El foco no es enumerar métricas de forma aislada, sino caracterizar el desempeño de generalización de los modelos, identificar qué señales están siendo explotadas y complementar los números con una lectura crítica a partir de errores, patrones por clase e importancia de variables.

La discusión se organiza por representación. Comenzamos por *Lightpath*, que constituye el caso más directo y funciona como *baseline* tabular para contextualizar el rendimiento y el tipo de señales disponibles. Luego pasamos a *Network Status*, donde la estructura del input y su costo de procesamiento vuelven más determinantes las decisiones de infraestructura y el compromiso costo–beneficio. El capítulo cierra con una síntesis comparativa y conclusiones parciales de la etapa experimental.

5.1. Representación: *Lightpath data*

La representación *lightpath* es el caso más directo del dataset: condensa cada conexión en un vector tabular de variables agregadas y, por tanto, permite entrenar y evaluar baselines fuertes con un costo de cómputo moderado. En esta sección se reportan resultados para dos familias de tareas: (i) **clasificación binaria de QoT** mediante `y_class`, y (ii) **regresión** de objetivos continuos (`y_ber` y `y_osnr`). A lo largo del capítulo se adopta la convención `y_class` $= 1$ como **conexión válida** (cumple umbral de QoT) y `y_class` $= 0$ como **no válida**. Bajo esta semántica, un *falso positivo* (FP) implica aceptar una conexión que en realidad no cumple QoT (riesgo operativo), mientras que un *falso negativo* (FN) implica rechazar una conexión viable (pérdida de oportunidad/-capacidad). Este bloque establece así una referencia clara para contrastar luego con *network status*, donde la representación más rica introduce mayores costos

y decisiones adicionales de infraestructura.

5.1.1. Setup experimental y resumen del dataset

La representación *lightpath* se modela como un problema tabular: cada fila corresponde a un *lightpath* y las variables de entrada describen, de forma agregada, su configuración y características del trayecto (por ejemplo, orden de modulación, número de spans/enlaces, longitudes y ocupación). Para asegurar reproducibilidad, los experimentos leen directamente particiones `train` y `test` generadas en la capa GOLD (archivos Parquet), de manera que el entrenamiento y la evaluación no dependen de transformaciones en cada ejecución. El detalle operativo del pipeline utilizado para generar estos artefactos se resume en el Anexo B.

En todos los experimentos reportados en esta sección se utilizaron 34 *features* numéricas, seleccionadas desde `train` excluyendo columnas auxiliares (`sample`), pesos (`weight`) y objetivos (`y_*`). La evaluación se realizó sobre un conjunto *hold-out* de 304 644 filas (TEST), que coincide con el total de filas usadas por el pipeline al momento de calcular métricas. El resumen de datos y configuración general se presenta en la Tabla 5.1.

Tabla 5.1: Resumen de datos y configuración general para la representación *Lightpath*. Se reporta el tipo de representación, la fuente de particiones utilizadas (capa GOLD), la cantidad de variables de entrada empleadas en los modelos y el tamaño del conjunto de evaluación *hold-out* (TEST).

Ítem	**Valor / descripción**
Representación	Tabular (una fila por *lightpath*; variables agregadas de configuración y trayecto).
Fuente de datos	Particiones `train` y `test` de la capa **GOLD** (Parquet).
Cantidad de *features*	34 variables numéricas (selección automática desde `train`; excluye `sample`, `weight` y `y_*`).
Conjunto de evaluación	Conjunto *hold-out* TEST: 304 644 filas.

Desde el punto de vista de ejecución, los modelos se entrenaron utilizando Spark en modo local. Para evitar inestabilidades por consumo de memoria y mantener comparabilidad entre corridas, se fijó una configuración estable: se controló el paralelismo local y se asignó memoria suficiente al proceso principal. En conjunto, estas decisiones buscan que las diferencias observadas entre modelos se deban principalmente al enfoque de aprendizaje y no a variaciones del entorno de ejecución.

5.1.2. Clasificación de QoT con Random Forest

El primer objetivo es la clasificación binaria de QoT a partir de `y_class` (con la semántica 1 = válida, 0 = no válida definida al inicio de la sección). Para

este caso se entrenó el modelo `RandomForestClassifier` de Spark ML con 200 árboles y profundidad máxima 12. En el conjunto de entrenamiento se incorporó una columna `weight` para ponderar clases y mitigar desbalance, mientras que el conjunto de prueba se mantuvo como referencia sin ponderaciones. Para el ensamblado de *features* se utilizó `VectorAssembler`.

La configuración del modelo y las decisiones relevantes de preprocesamiento se resumen en la Tabla 5.2. En particular, se explicitan los hiperparámetros principales (cantidad de árboles y profundidad), la estrategia adoptada para tratar el desbalance (ponderación solo en `train`) y el criterio aplicado ante valores inválidos en las variables de entrada.

Tabla 5.2: Configuración del modelo para clasificación de QoT con Random Forest (Spark ML). Se reportan los hiperparámetros principales del clasificador, la estrategia de ponderación para mitigar el desbalance de clases (aplicada únicamente en `train`) y el método de ensamblado de variables de entrada, incluyendo el tratamiento de filas con valores inválidos.

Ítem	**Valor / detalle**
Modelo	`RandomForestClassifier` (Spark ML)
Objetivo	Clasificación QoT (`y_class`)
Cantidad de árboles	200
Profundidad máxima	12
Desbalance	Columna `weight` solo en `train`; `test` sin ponderación.
Features	`VectorAssembler` con `handleInvalid=skip`.

En TEST se obtuvieron los resultados que se reportan en la Tabla 5.3. Se incluyen métricas complementarias: AUC-ROC y AUC-PR para evaluar capacidad de separación en presencia de desbalance, junto con Accuracy y F1 ponderado como medidas globales de desempeño sobre el conjunto de evaluación.

Tabla 5.3: Métricas en TEST para clasificación de QoT con Random Forest. Se reportan AUC-ROC y AUC-PR como métricas robustas a desbalance de clases, junto con Accuracy y F1 ponderado como indicadores globales de desempeño en el conjunto *hold-out*.

Métrica	**Valor**
AUC-ROC	0.9959
AUC-PR	0.9987
Accuracy	0.9647
F1 (weighted)	0.9653

Para interpretar el desempeño en términos de errores, la Tabla 5.4 presenta la matriz de confusión sobre TEST (TN, FP, FN, TP). Esta descomposición permite vincular el rendimiento del clasificador con decisiones operativas: con 1 = válida y 0 = no válida, los FP representan aceptación riesgosa, mientras que

los FN representan rechazos conservadores de conexiones que sí eran viables.

Tabla 5.4: Matriz de confusión en TEST para clasificación de QoT con Random Forest. Se reportan conteos de verdaderos negativos (TN), falsos positivos (FP), falsos negativos (FN) y verdaderos positivos (TP), permitiendo analizar el patrón de error del clasificador más allá de métricas agregadas.

Componente	**Conteo**
TN	72,466
FP	1,203
FN	9,560
TP	221,415

La matriz de confusión de la Tabla 5.4 refleja un conjunto con prevalencia de la clase 1 cercana al 75,8 %. El patrón de error es informativo: los falsos negativos son aproximadamente 8× más frecuentes que los falsos positivos. En términos operativos, esto indica que el modelo tiende a **rechazar** con mayor frecuencia casos que en realidad eran válidos (mayor conservadurismo), mientras que comete relativamente pocos casos de aceptación indebida (FP), que son los que implican mayor riesgo de degradación de QoT.

Respecto a interpretabilidad, la Tabla 5.5 reporta las importancias normalizadas del modelo para las *features* más influyentes. Se observa una concentración marcada: `mod_order` y `num_spans` explican en conjunto el 63.9 % de la importancia, y al incorporar `lp_linerate` el *top-3* alcanza aproximadamente 77.1 %. El resto del aporte se distribuye entre variables asociadas a capacidad/-configuración (*linerate*), complejidad del trayecto (`num_links`) y características geométricas y de carga de la ruta (longitudes y ocupación). En términos físicos, este patrón es coherente: el orden de modulación, la cantidad de spans y el *linerate* condicionan directamente los márgenes de transmisión y, por ende, la viabilidad de QoT.

Tabla 5.5: Importancias de variables (*feature importance*) para el modelo Random Forest de clasificación de QoT en la representación *lightpath*. Se reportan las 8 variables más relevantes según la importancia normalizada del modelo (Spark ML); el remanente se distribuye entre el resto de *features*.

Feature	**Importancia**	**%**
`mod_order`	0.378	37.8
`num_spans`	0.261	26.1
`lp_linerate`	0.132	13.2
`conn_linerate`	0.045	4.5
`num_links`	0.045	4.5
`max_link_len`	0.040	4.0
`avg_link_len`	0.026	2.6
`sum_link_occ`	0.023	2.3

5.1.3. Regresión de BER y OSNR/SNR con Random Forest Regressor

Además de QoT como clasificación, se evaluó la capacidad predictiva de la representación *lightpath* para objetivos continuos. Se entrenó el modelo **RandomForestRegressor** (Spark ML) con una configuración consistente con la utilizada en clasificación, y el pipeline guardó, además de métricas globales, las predicciones en TEST para habilitar análisis posteriores de residuales y calibración. La Tabla 5.6 resume la configuración del modelo y las salidas almacenadas.

Tabla 5.6: Configuración del modelo para regresión con Random Forest (Spark ML). Se reportan los hiperparámetros principales utilizados en todas las tareas continuas y la decisión de persistir predicciones en TEST para habilitar análisis de residuales y calibración sin reejecutar el pipeline.

Hiperparámetro	**Valor**
Modelo	`RandomForestRegressor` (Spark ML)
Cantidad de árboles	200
Profundidad máxima	12
`subsamplingRate`	0.8
`featureSubsetStrategy`	`sqrt`

Los resultados sobre TEST se presentan en la Tabla 5.7. Para BER (`y_ber`) se observa un ajuste alto ($R^2 = 0{,}8824$), aunque con mayor error residual que en OSNR. Para OSNR (`y_osnr`) el desempeño es muy alto ($R^2 = 0{,}9967$), lo que sugiere que, en el escenario simulado de este dataset, las variables agregadas disponibles en *lightpath* capturan casi toda la variación del objetivo.

Tabla 5.7: Métricas en TEST para regresión con Random Forest sobre la representación *lightpath*. Se reportan RMSE, MAE y R^2 para cada objetivo continuo evaluado, permitiendo comparar desempeño relativo entre objetivos bajo el mismo conjunto de variables de entrada y configuración de modelo.

Objetivo	**RMSE**	**MAE**	R^2
BER (`y_ber`)	0.001081	0.000660	0.8824
OSNR (`y_osnr`)	0.1671	0.1215	0.9967

La comparación de la Tabla 5.7 es coherente con el rol de cada objetivo: en este conjunto, OSNR está fuertemente determinado por factores estructurales del trayecto (por ejemplo, cantidad de spans/enlaces y longitudes), mientras que BER resulta más sensible a interacciones que no siempre quedan completamente capturadas por variables agregadas. En consecuencia, el error residual relativo es mayor para BER aun cuando el ajuste global sea bueno.

En línea con lo observado en clasificación, las importancias de variables muestran consistencia entre tareas. La Tabla 5.8 resume, de forma cualitativa, las variables dominantes por objetivo: para BER destacan `mod_order` y `num_spans`,

mientras que para OSNR domina `num_spans` acompañado por `mod_order` y `num_links`. Esta lectura refuerza la interpretación física de acumulación de degradación a lo largo del trayecto, y su impacto en los indicadores de calidad.

Tabla 5.8: Resumen cualitativo de importancias por tarea en Random Forest Regressor. Se listan variables dominantes y variables relevantes adicionales para cada objetivo continuo, con el fin de contrastar qué factores explican principalmente la variación de BER y de OSNR bajo la representación *lightpath*.

Objetivo	**Variables dominantes**	**Variables relevantes adicionales**
BER	`mod_order`, `num_spans`	—
OSNR	`num_spans`	`mod_order`, `num_links`

5.1.4. Discusión sobre los resultados

En conjunto, *lightpath* se comporta como una representación tabular altamente informativa: logra una capacidad discriminativa muy alta para QoT y un ajuste excelente para OSNR (Tabla 5.7), utilizando un conjunto reducido de variables con interpretación clara (Tablas 5.5 y 5.8). En particular, Random Forest emerge como un *baseline* competitivo y explicable, donde las variables dominantes se alinean con factores físicos esperables (modulación, cantidad de spans/enlaces, tasas y longitudes).

Al mismo tiempo, estos resultados deben leerse bajo el alcance del escenario evaluado: la capa GOLD de *lightpath* puede incluir variables cercanas a los indicadores de calidad, lo que facilita predicciones altamente precisas. Por esta razón, al reportar resultados se explicita qué información se permite en la representación y, cuando se requiera un enfoque estrictamente *a priori* (sin métricas de calidad observadas o derivadas), se complementa con corridas que excluyan variables directamente relacionadas con QoT.

Este bloque de resultados para *lightpath* establece una referencia directa para comparar con *network status*, donde se espera que la representación más rica aumente el costo de procesamiento y requiera decisiones adicionales de infraestructura y optimización del pipeline.

5.2. Representación: *Network status*

La representación *network status* complementa a *lightpath* al describir el estado global de la red en un instante. Mientras *lightpath* modela una conexión específica (una fila por *lightpath*), *network status* captura un *snapshot* del sistema (métricas por enlace activo y descriptores asociados), habilitando experimentos de QoT a nivel de red y permitiendo estudiar explícitamente el rol del contexto (carga, ocupación y vecindad espectral). En toda esta sección se mantiene la misma convención que en *lightpath*: `y_class` $= 1$ indica una conexión

viable (cumple el umbral de QoT) y `y_class` = 0 una conexión **no viable**.

Por volumen, el dataset se procesa por *buckets*: cada *bucket* identifica una partición del conjunto GOLD (Parquet) que puede leerse y procesarse de forma independiente. Esta organización permite escalar la experimentación (lectura paralela, control de particiones y re-ejecución selectiva) y se alinea con la ejecución distribuida del pipeline.

Sobre esta representación se evaluaron cuatro enfoques de modelado:

- **Baseline tabular (RF):** modelos tradicionales sobre una vectorización numérica por muestra.
- **Tabular no lineal (MLP):** perceptrón multicapa para capturar interacciones más complejas.
- **Secuencial (RNN):** entrada estructurada por enlace para explotar organización interna del *snapshot*.
- **Convolucional (DCNN) (Safari y cols., 2021b):** consumo de un tensor enlace–frecuencia para capturar patrones locales en el dominio espacial/espectral.

La Tabla 5.9 resume el mapa de experimentos: tipo de entrada, cobertura (buckets) y stack de ejecución, de modo de orientar la lectura antes del detalle de cada modelo.

Tabla 5.9: Mapa de experimentos para *Network Status*. Se resume, por modelo, el tipo de entrada, la cobertura por-*bucket* utilizada y el stack de ejecución.

Modelo	**Entrada**	**Buckets**	**Tamaño (aprox.)**	**Ejecución**
RF	Tabular (GOLD Parquet)	000–152	1.52M muestras	Spark (Slurm)
MLP	Tabular (GOLD `mlp_v2`)	000–152	1.52M muestras	Spark / PyTorch (Slurm)
RNN	Secuencial (NPZ $N \times T \times F$)	0–9	100k muestras	PyTorch (Slurm/CPU)
DCNN	Tensor (2 canales, enlace–freq.)	70 / 152	700k / 1.52M	PyTorch (Slurm/CPU)

Dado el volumen del dataset completo y el procesamiento por *bucket*, la experimentación se ejecutó en infraestructura distribuida mediante Slurm (`sbatch`). Se utilizó Spark para lectura y transformación desde la capa GOLD y para el entrenamiento/evaluación de los modelos tabulares. En cambio, el enfoque convolucional opera sobre tensores materializados por el pipeline y se entrena en un flujo separado, manteniendo el mismo protocolo de particionado y evaluación para asegurar comparabilidad.

5.2.1. Setup experimental, ejecución distribuida y construcción de entradas

Capa GOLD y particionado por *bucket*. La capa GOLD para *network status* se almacenó en formato Parquet y se particionó por `bucket=XYZ`. Este diseño permite procesar el dataset completo de forma escalable (lectura paralela, control del tamaño de particiones y re-ejecución selectiva) y se alinea con el flujo de ejecución distribuida del pipeline. El detalle operativo de los scripts utilizados para esta representación se resume en el Anexo C.

Ejecución en clúster (Slurm + Spark). La etapa de entrenamiento y evaluación se orquestó con `sbatch`, habilitando la ejecución controlada de *jobs* sobre el clúster. En particular, los scripts leen el conjunto de directorios `bucket=*` de manera explícita y ejecutan el flujo *read* → *split* → *fit* → *evaluate* sobre Spark para los modelos tabulares.

Tres familias de entrada (tabular, secuencial y tensorial). Se definieron tres tipos de entradas para cubrir los modelos evaluados:

- **Entrada tabular:** una vectorización numérica por muestra (utilizada por RF y MLP).
- **Entrada estructurada/secuencial:** conserva una organización ordenada del *snapshot* (por ejemplo, por enlaces activos), utilizada por la RNN.
- **Entrada tensorial (grilla enlace–frecuencia):** preserva explícitamente vecindades locales y se utiliza por la CNN tipo DCNN, cuyo objetivo es aprender patrones locales en el dominio enlace–frec sin colapsar la estructura en un vector.

Criterios metodológicos comunes. Para todas las variantes se aplicaron criterios consistentes:

- Split estratificado (train/test) respecto de la clase objetivo `y_class`, para preservar proporciones bajo desbalance.
- Mitigación del desbalance: entrenamiento con pesos por clase cuando corresponde.
- Prevención de fuga de información (*leakage*): exclusión sistemática de columnas objetivo (`y_*`), variables derivadas directas del *label* y señales excesivamente cercanas (p. ej., `m_ber` en clasificación QoT, según configuración del experimento).
- Métricas reportadas: se priorizan métricas robustas a desbalance (AUC-PR) y métricas globales (accuracy, F1 ponderado), complementadas con matriz de confusión para interpretar tipos de error.

5.2.2. Modelo 1: Random Forest (baseline tabular) para clasificación QoT

Como primer baseline tabular se entrenó el modelo **RandomForestClassifier** (Spark ML) sobre la salida GOLD particionada por *bucket*. El modelo utiliza únicamente *features* numéricas y se configuró con 200 árboles y profundidad máxima 12, según se resume en la Tabla 5.10.

Un punto metodológico relevante es la exclusión de `m_ber` como *feature*. Esta variable corresponde a una medición (o estimación) directamente asociada a desempeño de transmisión y suele estar muy cercana a la regla operacional que define la viabilidad de QoT. En particular, si la etiqueta `y_class` se deriva explícitamente de un umbral sobre BER (o de una métrica fuertemente correlacionada con ella), incluir `m_ber` como entrada puede inducir fuga de información: el modelo podría aprender una aproximación de la regla de etiquetado en lugar de aprender relaciones causales a partir de variables de configuración/estado. Por ello, se excluyó `m_ber` para reducir el riesgo de que la evaluación refleje un atajo artificial y para favorecer una lectura más honesta del desempeño cuando se pretende predecir viabilidad a partir de condiciones de red.

Tabla 5.10: Configuración del modelo de clasificación tabular para la representación *Network Status*. Se indica la fuente de entrada (GOLD particionado por *bucket*), el tipo de variables utilizadas (numéricas), los hiperparámetros principales del `RandomForestClassifier` y la decisión metodológica de excluir `m_ber` para mitigar el riesgo de fuga de información.

Ítem	**Valor / detalle**
Modelo	`RandomForestClassifier` (Spark ML)
Entrada	GOLD tabular por-*bucket* (`bucket=000..152`)
Features	Numéricas; `m_ber` excluida (control de fuga de información).
Cantidad de árboles	200
Profundidad máxima	12

Distribución de clases y split. La Tabla 5.11 muestra la distribución global de clases y el particionado utilizado. Se observa un desbalance marcado (mayoría de clase 1), por lo que se aplicó un split estratificado 80/20 respecto de `y_class` para preservar proporciones en entrenamiento y test. Para mitigar el impacto del desbalance durante el ajuste, el entrenamiento incorporó pesos por clase, manteniendo el conjunto de test sin ponderación para una evaluación consistente.

Resultados. Las métricas obtenidas en test se resumen en la Tabla 5.12. En particular, se reporta AUC-ROC = 0,756 y AUC-PR = 0,917, junto con *accuracy* = 0,625 y F1 ponderado = 0,650. En un escenario desbalanceado, AUC-PR

Tabla 5.11: Distribución global de clases y esquema de particionado para el baseline Random Forest en Network Status. Se reportan los conteos por clase en el conjunto global y los tamaños de train/test obtenidos mediante un split estratificado 80/20. También se indica el uso de pesos por clase en entrenamiento para mitigar el desbalance.

Descripción	**Valor**
Clase 0 (global)	369,120
Clase 1 (global)	1,155,635
Split	Estratificado 80/20 por `y_class`
Train	1,219,360
Test	305,395
Mitigación de desbalance	Pesos por clase $\propto N/(K \cdot n_c)$

resulta especialmente útil para interpretar el compromiso precisión–recobrado de la clase positiva, complementando la lectura más global de AUC-ROC.

Tabla 5.12: Métricas en test para clasificación de QoT con Random Forest sobre Network Status. Se reportan AUC-ROC y AUC-PR (robustas para comparar capacidad discriminativa), además de *accuracy* y F1 ponderado como medidas globales del desempeño.

Métrica	**Valor**
AUC-ROC	0.756
AUC-PR	0.917
Accuracy	0.625
F1 ponderado	0.650

La interpretación de tipos de error se apoya en la Tabla 5.13. Se observa que FN (106,073) es muy superior a FP (8,391), lo que indica un sesgo hacia decisiones conservadoras: bajo el umbral implícito de decisión, el modelo tiende a *rechazar* con mayor frecuencia, perdiendo una fracción relevante de conexiones viables. Esta lectura es importante porque FN y FP tienen costos operativos distintos: FN implica pérdida de oportunidad (bloquear una conexión factible), mientras que FP implica riesgo de degradación (aceptar una conexión inviable).

Tabla 5.13: Patrón de error observado en test para el baseline Random Forest sobre Network Status. Se reportan falsos negativos y falsos positivos tomando `y_class` $= 1$ (conexión viable) como clase positiva.

Componente	**Conteo**
Falsos negativos (FN)	106,073
Falsos positivos (FP)	8,391

Señales dominantes y lectura metodológica. La Tabla 5.14 reporta las importancias normalizadas del modelo para las *features* más influyentes. Se observa que `m_osnr` y `m_snr` concentran la mayor parte del poder explicativo (en conjunto, 90.7 % de la importancia), mientras que el resto de variables individuales aporta contribuciones marginales ($\leq 2\,\%$ cada una). Esto es consistente con el dominio: métricas de calidad capturan de forma directa la degradación y tienden a dominar la predicción. Al mismo tiempo, refuerza la lectura metodológica del baseline: cuando la entrada incluye mediciones muy cercanas al resultado, el modelo puede lograr buen desempeño sin necesariamente aprender relaciones más generales basadas en configuración y contexto de red. Finalmente, la presencia (aunque baja) de identificadores como `conn_id` y `dst_id` sugiere una señal residual asociada a estructura/topología o correlaciones espurias, por lo que se plantea como paso siguiente un estudio de ablación que remueva variables "cercanas al outcome" y también *IDs* numéricos, para evaluar robustez y generalización.

Tabla 5.14: Importancias de variables (*feature importance*) para el baseline Random Forest en Network Status. Se reportan las 8 variables más relevantes según la importancia normalizada del modelo (Spark ML); el remanente se distribuye entre el resto de *features*.

Feature	**Importancia**	**%**
`m_osnr`	0.482	48.2
`m_snr`	0.426	42.6
`conn_id`	0.020	2.0
`lp_ber`	0.010	1.0
`dst_id`	0.007	0.7
`lp_num_spans`	0.006	0.6
`conn_linerate`	0.006	0.6
`min_link_len`	0.006	0.6

5.2.3. Modelo 2: MLP (tabular) para clasificación QoT

Este segundo baseline utiliza un perceptrón multicapa (MLP) sobre una representación tabular, con el objetivo de capturar relaciones no lineales entre variables que pueden no ser modeladas de forma directa por enfoques basados en particiones.

Origen de la entrada GOLD. La tabla GOLD se construye imponiendo una fila por muestra (`sample`). Para cada muestra se conserva un conjunto de variables de ruta/configuración (propias del servicio) y se agregan estadísticos de estado de red a partir de mediciones por fila en SILVER. Además, se define la etiqueta binaria (`label_i`) a partir de la variable de clase en SILVER. La Tabla 5.15 resume la estructura conceptual de `mlp_v2` y explicita los grupos de columnas que resultan del proceso de agregación, incluyendo una variable de

control (`rows_per_sample`) que indica cuántas filas de SILVER se consolidaron para formar cada muestra.

Tabla 5.15: Estructura conceptual de `mlp_v2` (capa GOLD). Se agrupan los campos en identificadores/partición, etiqueta, variables de ruta/configuración y resúmenes agregados del estado de red. También se incluye `rows_per_sample` como control del proceso de agregación desde SILVER.

Grupo	**Descripción**
Identificadores y partición	`sample` (id de muestra), `sample_bucket` (bucket de origen).
Etiqueta	`label_i` (clase binaria), `label` (versión numérica).
Ruta / configuración	Variables del servicio y su ruta (distancias, saltos, orden de modulación, tasas, grados nodales e ids).
Estado de red (resúmenes)	Estadísticos agregados a partir de múltiples filas en SILVER (media, desvío, mínimo, máximo y percentiles).
Control de agregación	`rows_per_sample`: cantidad de filas agregadas en SILVER para formar una muestra GOLD.

Las variables preservadas de ruta/configuración se listan en la Tabla 5.16. Estas columnas describen el servicio y el trayecto, e incluyen parámetros del *lightpath* (por ejemplo, longitudes, número de spans y enlaces) junto con campos que identifican extremos o describen su conectividad. Por su parte, la Tabla 5.17 resume las variables agregadas del estado de red: en lugar de conservar todas las filas de SILVER, se computan estadísticos (media, dispersión, extremos y percentiles) sobre mediciones asociadas a la muestra, condensando así información del snapshot en una representación de dimensión fija adecuada para modelos tabulares.

Tabla 5.16: Variables de ruta/configuración preservadas en la entrada GOLD. Estas columnas describen el servicio y su trayecto (longitudes, spans/enlaces, parámetros del *lightpath* y atributos de los nodos extremos) y se mantienen sin agregación, ya que son intrínsecas a la muestra.

Variable	**Interpretación (resumen)**
`mod_order`	Orden de modulación asociado al servicio.
`conn_linerate`	Tasa del servicio o conexión (*line rate*).
`lp_linerate`	Tasa del *lightpath* (*line rate*).
`len_km`	Longitud total aproximada de la ruta (km).
`nspans`	Número de tramos (*spans*) en la ruta.
`src_degree`, `dst_degree`	Grado del nodo de origen y del nodo de destino.
`path_len`	Longitud de la ruta (representación interna o medida de recorrido).
`num_links`	Cantidad de enlaces en la ruta.
`min_link_len`, `max_link_len`	Longitudes mínima y máxima de los enlaces en la ruta.
`src_id`, `dst_id`	Identificadores de los nodos de origen y destino.
`lp_snr`, `lp_osnr`	Métricas de calidad del *lightpath* (SNR y OSNR).
`lp_num_spans`	Número de *spans* del *lightpath*.

Tabla 5.17: Resúmenes de estado de red agregados en la entrada GOLD por muestra. Para cada métrica se computan estadísticos (media, desviación estándar, extremos y percentiles), de modo de condensar múltiples filas de SILVER en una representación numérica fija. `rows_per_sample` permite controlar cuántas observaciones contribuyeron a cada agregado.

Variable	**Interpretación (resumen)**
`m_snr_mean`, `m_snr_std`	Media y dispersión (desvío estándar) de SNR agregada.
`m_snr_min`, `m_snr_max`	Extremos (mínimo y máximo) de SNR agregada.
`m_snr_p10`, `m_snr_p50`, `m_snr_p90`	Percentiles 10, 50 y 90 de SNR agregada.
`m_osnr_mean`, `m_osnr_std`	Media y dispersión (desvío estándar) de OSNR agregada.
`m_osnr_min`, `m_osnr_max`	Extremos (mínimo y máximo) de OSNR agregada.
`m_osnr_p10`, `m_osnr_p50`, `m_osnr_p90`	Percentiles 10, 50 y 90 de OSNR agregada.
`rows_per_sample`	Número de filas de SILVER utilizadas para formar la muestra.

Particionado y distribución de clases. El entrenamiento se realizó concatenando los buckets `000..152` y aplicando un particionado hold-out 70/15/15. La Tabla 5.18 reporta tamaños y distribución de clases para `train`, `val` y `test`, evidenciando el desbalance (clase 1 mayoritaria) también en esta vista agregada por muestra. Esta distribución motiva que, además de métricas globales, se analicen criterios sensibles al desempeño en ambas clases (por ejemplo, macro-F1) y que el umbral de decisión se trate como un grado de libertad.

Tabla 5.18: Particionado hold-out (70/15/15) y distribución de clases para la tabla. Se reportan los tamaños de `train`/`val`/`test` y el conteo por clase, evidenciando el desbalance hacia la clase 1.

Conjunto	Tamaño	Clase 0	Clase 1
Train	1,067,304	258,382	808,922
Val	228,634	55,361	173,273
Test	228,817	55,377	173,440

Configuración y selección de variables. Se seleccionaron automáticamente variables numéricas de la tabla GOLD y se excluyeron columnas auxiliares. En pruebas preliminares se observó que ciertas columnas de tipo identificador (por ejemplo, `src_id`, `dst_id`) y columnas de control de agregación (`rows_per_sample`) no necesariamente aportan señal útil directa para clasificación y pueden introducir ruido o sesgos por efectos de codificación. Por ello se evaluaron variantes eliminándolas y se fijó un conjunto de entrada para las corridas principales. La Tabla 5.19 lista las 28 variables numéricas finalmente utilizadas, que combinan descriptores del trayecto (`len_km`, `nspans`, `num_links`, `path_len`) con resúmenes del estado de red y métricas del *lightpath*.

Tabla 5.19: Conjunto final de variables utilizado en las corridas principales del MLP ($n = 28$). Se incluyen descriptores del trayecto y variables agregadas del estado de red, además de métricas disponibles a nivel de *lightpath*.

Variables
`conn_linerate`, `dst_degree`, `len_km`, `lp_linerate`, `lp_num_spans`, `lp_osnr`, `lp_snr`, `m_osnr_max`, `m_osnr_mean`, `m_osnr_min`, `m_osnr_p10`, `m_osnr_p50`, `m_osnr_p90`, `m_osnr_std`, `m_snr_max`, `m_snr_mean`, `m_snr_min`, `m_snr_p10`, `m_snr_p50`, `m_snr_p90`, `m_snr_std`, `max_link_len`, `min_link_len`, `mod_order`, `nspans`, `num_links`, `path_len`, `src_degree`.

Experimentos y resultados. Se realizaron experimentos en dos etapas: (i) baseline en Spark ML y (ii) migración a PyTorch para habilitar un control más fino del entrenamiento, optimización de hiperparámetros y análisis de interpretabilidad.

(i) Baseline Spark ML: sensibilidad a desbalance y umbral. En una primera etapa se exploró el impacto de balancear `train` por muestreo. El muestreo balanceado (submuestreo de la clase mayoritaria) redujo el sesgo hacia la clase dominante, pero degradó el desempeño global en métricas agregadas. En cambio, al mantener el desbalance original, el modelo alcanzó mejor desempeño global. La Tabla 5.20 compara ambas variantes en test y muestra esta diferencia: la variante sin balance alcanza mayor *accuracy* y AUC-PR, mientras que la variante con balance mejora el énfasis relativo sobre la clase minoritaria (reflejado en F1-macro), a costa de una caída en desempeño agregado.

Tabla 5.20: Resultados del MLP entrenado en Spark ML sobre `mlp_v2` (test). Se comparan dos variantes: con balance por submuestreo y sin balance (distribución original). La tabla resume el compromiso entre métricas globales y métricas sensibles al desbalance.

Variante	**Accuracy**	**F1-macro**	**AUC-ROC**	**AUC-PR**
Con balance (submuestreo)	0.576	0.604	0.672	0.882
Sin balance	0.757	0.657	0.729	0.909

(ii) Migración a PyTorch: entrenamiento por épocas y ajuste con Optuna. La migración a PyTorch mantuvo la misma entrada (`mlp_v2`) y el mismo particionado, y permitió entrenar por épocas observando evolución de pérdida y métricas de validación. Luego se aplicó Optuna para optimizar hiperparámetros de arquitectura y regularización maximizando AUC-PR en validación. La Tabla 5.21 resume el desempeño final en test para dos configuraciones: (a) entrenamiento inicial con pocas épocas y (b) configuración seleccionada mediante Optuna y re-entrenada con más épocas. En conjunto, las mejoras son consistentes aunque moderadas en AUC-ROC/AUC-PR, mientras que macro-F1 y balanced accuracy permanecen estables, lo que sugiere que el principal beneficio proviene de ajustes finos más que de un cambio radical en la capacidad discriminativa.

Tabla 5.21: Resultados en test para el MLP implementado en PyTorch. Se compara una configuración base (pocas épocas) con una configuración ajustada mediante Optuna. En ambos casos se utilizó selección de umbral optimizando macro-F1 en validación, para mitigar el sesgo inducido por desbalance.

Configuración	**AUC-ROC**	**AUC-PR**	**Macro-F1**	**Bal. Acc.**
PyTorch (5 épocas, umbral optimizado)	0.733	0.911	0.613	0.643
PyTorch + Optuna (10 épocas, umbral optimizado)	0.738	0.913	0.613	0.665

Selección de umbral y lectura operacional. Con desbalance marcado (clase 1 mayoritaria), un umbral estándar tiende a maximizar el acierto global prediciendo mayormente la clase dominante. Por ello se utilizó un barrido de umbral para optimizar macro-F1 en validación, y el umbral seleccionado se aplicó a test. La Tabla 5.22 muestra la matriz de confusión final (configuración PyTorch+Optuna con umbral optimizado), permitiendo interpretar el compromiso alcanzado: se incrementan las detecciones de la clase minoritaria (reducción de FN) a costa de un aumento de FP, patrón esperable al desplazar el umbral hacia una decisión más equilibrada entre clases.

Tabla 5.22: Matriz de confusión en test para el MLP final (PyTorch+Optuna) con umbral optimizado en validación. La tabla permite analizar el tipo de error dominante (FN vs. FP) y contextualizar métricas agregadas en un escenario desbalanceado.

Componente	**Conteo**
TN	38,188
FP	17,265
FN	62,277
TP	111,087

Interpretabilidad con Captum (Integrated Gradients). Para interpretar el modelo final se aplicó un método de atribución basado en gradientes (Integrated Gradients) sobre un subconjunto de muestras, agregando las atribuciones para obtener un ranking global. La Tabla 5.23 reporta el top-10 de variables por importancia global (promedio del valor absoluto), mostrando dominancia de métricas asociadas al *lightpath* y resúmenes del estado de red, seguidas por un descriptor de ruta (`path_len`). Este resultado es coherente con la intuición física: variables de calidad y su contexto explican gran parte de la variación en viabilidad, mientras que los descriptores geométricos del trayecto actúan como factores estructurales de segundo orden.

Conclusiones parciales. Los resultados permiten extraer tres observaciones. Primero, el desbalance de clases hace que la selección de umbral sea un factor crítico: optimizar macro-F1 en validación mejora el compromiso entre clases sin necesidad de re-entrenar el modelo. Segundo, la migración a PyTorch habilita una optimización sistemática de hiperparámetros (Optuna) y un entrenamiento por épocas más controlado, logrando mejoras consistentes en AUC-PR/AUC-ROC, aunque de magnitud acotada para esta representación. Tercero, el análisis de atribuciones muestra que el modelo apoya sus decisiones en variables con interpretación física esperable (métricas del *lightpath* y resúmenes del estado de red), lo que resulta útil tanto para validar coherencia como para orientar futuras reducciones de dimensionalidad y depuración de variables.

Tabla 5.23: Top-10 de variables según importancia global estimada con Integrated Gradients (promedio del valor absoluto) para el MLP final. Se observa dominancia de métricas del lightpath y resúmenes del estado de red, complementadas por descriptores de ruta.

Feature	Importancia
lp_osnr	0.0997
lp_snr	0.0909
m_snr_max	0.0583
m_snr_min	0.0497
m_snr_p10	0.0479
m_osnr_p50	0.0465
m_snr_p50	0.0451
m_osnr_max	0.0447
m_snr_p90	0.0443
path_len	0.0439

Próximos pasos. Como extensiones directas se plantea: (a) evaluar estabilidad de importancias bajo distintas semillas y subconjuntos, (b) probar variantes de regularización y calibración, y (c) comparar de forma controlada MLP en Spark ML vs PyTorch bajo el mismo particionado, conjunto de variables y criterio de selección de umbral.

5.2.4. Modelo 3: RNN (entrada secuencial) para clasificación QoT

Además de los modelos tabulares (RF/MLP) sobre un vector de variables por snapshot, se evaluó un enfoque secuencial basado en RNN con el objetivo de preservar parte de la estructura interna del estado de red. La representación *network status* describe simultáneamente múltiples enlaces activos bajo una misma configuración global; por tanto, una entrada secuencial permite modelar el snapshot como una colección ordenada de mediciones por enlace, evitando la agregación total en un único vector. En este trabajo la RNN se utiliza como baseline secuencial para estudiar si explotar esta organización aporta señal adicional frente a las variantes tabulares.

Procesamiento adicional: construcción de GOLD secuencial y exportación a NPZ. A diferencia de RF/MLP (que consumen GOLD tabular en Parquet), el modelo secuencial requirió dos pasos extra. La Tabla 5.24 resume las etapas y las decisiones clave: (i) construir un GOLD secuencial a nivel de fila por bucket, preservando la secuencia de enlaces para un `freq_idx` de referencia por muestra, y (ii) exportar esa representación a tensores en formato `.npz` para entrenamiento eficiente en un framework de *deep learning*.

1. **SILVER → GOLD (RNN, row-level por bucket).** Para cada `sample` se selecciona un único `freq_idx` de referencia tomando el mínimo `freq_idx`

observado en `link==0` (mismo criterio base usado en los modelos tabulares). Luego se conservan todas las filas del snapshot correspondientes a ese `freq_idx`, manteniendo todos los enlaces para formar la secuencia. Se define la etiqueta como `y_class = int(m_class)`, se añade `sample_bucket`, y se define explícitamente el orden secuencial como `step = link`. Por diseño, se excluye `m_ber` de la salida para reducir el riesgo de fuga de información: en esta colección, `m_ber` está fuertemente ligado al desempeño físico y puede resultar una señal demasiado cercana a la definición del *label*, inflando artificialmente el desempeño si el objetivo es evaluar capacidad predictiva a partir de variables menos "directas".

2. **GOLD → NPZ (tensor $N \times T \times F$).** A partir del GOLD secuencial se agrupa por `sample` y se construye la secuencia como una lista de `struct(step, f1, f2, ...)`. Luego se ordena con `sort_array` para garantizar que la entrada respete el orden creciente de `step` (equivalente a `link`). Finalmente, se exporta a `.npz` por bucket con:

$$X \in \mathbb{R}^{N \times T \times F}, \quad y \in \{0,1\}^N,$$

donde T es el largo de la secuencia (cantidad de enlaces conservados) y F el número de variables numéricas seleccionadas.

Tabla 5.24: Resumen del procesamiento adicional requerido por la entrada secuencial (RNN). Se describen las etapas de construcción del GOLD secuencial y su exportación a tensores `.npz`, junto con las decisiones que fijan la semántica y el orden de la secuencia.

Etapa	**Entrada → salida**	**Decisiones clave**
1) Construcción GOLD secuencial	SILVER → GOLD (row-level, por *bucket*)	`freq_idx` por muestra = mínimo observado en `link==0`; se conservan todas las filas del snapshot para ese `freq_idx`; `y_class=int(m_class)`; `step=link`; se excluye `m_ber`.
2) Exportación a tensor	GOLD secuencial → NPZ	Agrupar por `sample`; construir secuencia `struct(step, f1, ...)`; ordenar con `sort_array`; exportar $X \in \mathbb{R}^{N \times T \times F}$, $y \in \{0,1\}^N$.

Definición de la secuencia (semántica y orden). Bajo este diseño, cada muestra corresponde a un snapshot y se representa como una secuencia de pasos. La Tabla 5.25 formaliza la semántica de esta decisión: el paso t corresponde a un enlace del snapshot, el orden se fija ascendente por `link`, y se utiliza un único `freq_idx` por muestra (seleccionado de forma determinista) para evitar que la longitud de la secuencia o el contenido dependan de múltiples selecciones espectrales.

Tabla 5.25: Definición de la secuencia utilizada por la RNN. Se fija la unidad de paso, el criterio de ordenamiento y la regla de selección espectral que determina qué filas del snapshot se preservan para cada muestra.

Componente	**Definición**
Unidad de paso (t)	Un enlace del snapshot (campo `link`).
Orden	Ascendente por `link` (`step=link`; ordenado con `sort_array`).
Selección espectral	Un único `freq_idx` por muestra: mínimo observado en `link==0`; se conservan todos los enlaces para ese `freq_idx`.

Selección de variables y control de leakage. Para construir los tensores se seleccionan columnas numéricas excluyendo claves/IDs y variables de partición/objetivo. En particular, se excluyen `sample`, `y_class`, `step`, `link`, `sample_bucket` y otros identificadores típicamente categóricos (`conn_id`, `src_id`, `dst_id`). Adicionalmente, `m_ber` se elimina por defecto en el GOLD secuencial: dado que BER es una métrica directamente asociada a calidad, su presencia como entrada puede acercar el modelo a un esquema de "detección" basado en la propia definición de calidad, en lugar de evaluar el aporte de la estructura del snapshot. La Tabla 5.26 resume estos criterios y agrega una decisión de estandarización: las estadísticas se calculan solo sobre TRAIN y se aplican luego a TRAIN y TEST, evitando contaminación de información desde el conjunto de evaluación.

Tabla 5.26: Criterios de selección de variables y control de fuga de información en la preparación de entrada secuencial. Se excluyen identificadores y columnas no predictivas, se elimina `m_ber` por su cercanía al *label*, y se estandariza usando solo estadísticas de entrenamiento.

Aspecto	**Decisión**
Selección de columnas	Columnas numéricas; exclusión de claves, identificadores y particiones/objetivo.
Exclusiones explícitas	`sample`, `y_class`, `step`, `link`, `sample_bucket`, `conn_id`, `src_id`, `dst_id`.
Leakage	`m_ber` eliminado por defecto en el GOLD secuencial.
Estandarización	Estadísticas calculadas solo en TRAIN; luego aplicadas a TRAIN y TEST.

Entrenamiento y evaluación (configuración, 10 buckets). El entrenamiento se realizó en modo global concatenando 10 archivos `.npz` (buckets 0–9) y entrenando un único modelo. La arquitectura utilizada fue una GRU (1 capa) con estado oculto de tamaño 64, como punto de partida para evaluar el aporte del enfoque secuencial. La Tabla 5.27 resume los datos y los hiperparámetros: el experimento utiliza $N = 100{,}000$ muestras con secuencias de longitud fija $T = 198$ y $F = 19$ variables por paso, bajo un desbalance consistente con el

observado en otras variantes. Para mitigar este desbalance se utilizó una pérdida de cross-entropy con pesos balanceados.

Tabla 5.27: Datos e hiperparámetros del baseline secuencial (GRU) entrenado sobre 10 *buckets*. Se reporta el tamaño del conjunto, la dimensionalidad del tensor de entrada y los hiperparámetros principales del entrenamiento.

Ítem	**Valor**
Modo de entrenamiento	Global (concatena 10 `.npz`, *buckets* 0–9).
Arquitectura	GRU (1 capa), `hidden`=64.
Muestras	$N = 100{,}000$ (TRAIN 80,000 / TEST 20,000).
Dimensión de entrada	$T = 198$ pasos; $F = 19$ variables.
Desbalance global	Clase 0: 24,530 (24.5 %); clase 1: 75,470 (75.5 %).
`epochs`	10
`batch`	64
`layers`	1
`dropout`	0.1
`lr`	0.001
`seed`	42
Pérdida	Cross-Entropy con pesos balanceados.
Dispositivo	CPU

Resultados. En test se obtuvo *accuracy* = 0,610 y F1-macro = 0,594. La Tabla 5.28 resume estas métricas, y se utilizan como referencia directa para comparar contra los baselines tabulares cuando se controlan buckets y particionado.

Tabla 5.28: Métricas en test para el baseline secuencial (RNN/GRU) entrenado sobre 10 buckets. Se reportan *accuracy* y F1-macro, esta última más informativa bajo desbalance.

Métrica	**Valor**
Accuracy	0.610
F1-macro	0.594

La matriz de confusión en test ($N = 20{,}000$) se reporta en la Tabla 5.29. Esta tabla permite caracterizar el tipo de error dominante y vincularlo con decisiones de diseño (pérdida ponderada, umbral por defecto y capacidad del modelo).

Interpretación. El patrón observado en la matriz de confusión sugiere un sesgo hacia minimizar falsas alarmas: los falsos positivos son bajos en comparación con los falsos negativos. Esto se refleja en la Tabla 5.30, donde se resume la lectura del comportamiento: para la clase positiva (clase 1), la precisión es alta (aprox. 0.91) pero el recall es moderado (aprox. 0.54), perdiendo una fracción relevante de positivos reales. En términos operacionales, este compromiso

Tabla 5.29: Matriz de confusión en test para el baseline secuencial (RNN/GRU). Se reportan TN/FP/FN/TP para interpretar el patrón de error más allá de las métricas agregadas.

Componente	**Conteo**
TN	4,096
FP	823
FN	6,978
TP	8,103

puede ser aceptable si el objetivo es reducir alarmas, pero puede ser riesgoso si se prioriza no omitir casos positivos.

Tabla 5.30: Resumen del patrón de error e interpretación del baseline secuencial. Se destaca la asimetría FN$\gg$FP y se reporta, para la clase positiva, el compromiso entre precisión y recall inducido por el criterio de entrenamiento y la decisión por defecto.

Aspecto	**Resultado**
Errores dominantes	FP = 823 (bajos), FN = 6,978 (altos).
Clase 1 (positiva)	Precisión $\approx$ 0,91; recall $\approx$ 0,54.
Lectura	Tendencia a minimizar falsas alarmas, con pérdida de una fracción relevante de positivos reales.

Discusión y próximos pasos. Este experimento constituye un baseline secuencial inicial y deja definidos ejes concretos de mejora. La Tabla 5.31 organiza estos ejes: ampliar cobertura (más buckets) para reducir varianza y mejorar generalización, explorar mayor capacidad o arquitecturas que exploten mejor estructura, y realizar una comparación controlada contra RF/MLP bajo el mismo conjunto de buckets y particionado para obtener evidencia directamente comparable.

Tabla 5.31: Ejes de mejora propuestos a partir del baseline secuencial. Se organizan en tres líneas: cobertura de datos, capacidad/arquitectura del modelo y comparación controlada contra baselines tabulares.

Eje	**Descripción**
Cobertura	Ampliar el entrenamiento global a más *buckets* para reducir la varianza y mejorar la generalización.
Modelo	Explorar mayor capacidad (más capas, mayor `hidden`) y alternativas que exploten mejor la estructura (p. ej., mecanismos de atención).
Comparación	Contrastar RF/MLP frente a RNN bajo el mismo conjunto de *buckets* y el mismo particionado, para concluir con evidencia comparable.

5.2.5. Modelo 4: Clasificación de QoT con CNN

En esta subsección se replica el enfoque de *Deep Convolutional Neural Network* (DCNN) propuesto en (Safari y cols., 2021b) para estimación *network-wide* de QoT. La idea central es evitar una vectorización tabular del snapshot y, en cambio, describir el estado de la red mediante matrices sobre las cuales la CNN puede aprender patrones locales en el dominio enlace–frecuencia.

Representación de entrada (formulación del paper). Siguiendo (Safari y cols., 2021b), cada estado de red i se representa mediante dos matrices $D_{i,1}$ y $D_{i,2}$ de tamaño $L \times (2 + F)$, donde L es el número de enlaces unidireccionales de la topología y F el número de *frequency slots*. La Tabla 5.32 resume esta definición: las primeras dos columnas codifican *features* topológicas por enlace (longitud y número de spans), mientras que el bloque $L \times F$ codifica el estado espectral por enlace y slot. En particular, $D_{i,1}$ almacena valores de BER de *lightpaths* activos y marca el *lightpath under test* (LUT) con un valor especial (en el paper se utiliza -1 para indicar que su QoT es desconocida *a priori*); $D_{i,2}$ almacena la cardinalidad de modulación de los *lightpaths* activos y del LUT, dado que la configuración planificada del LUT es conocida (Safari y cols., 2021b).

Construcción práctica de tensores y variables representadas. En la implementación, las dos matrices se apilan como canales para obtener una entrada de forma $(2, L, 2 + F)$ por muestra (y luego $(N, 2, L, 2 + F)$ a nivel de dataset). La Tabla 5.33 apoya esta lectura al desglosar qué información queda en cada canal: ambos incluyen las dos columnas topológicas, y difieren en el mapa enlace–slot (BER en $D_{i,1}$, modulación en $D_{i,2}$). Esta representación es la que habilita a la CNN a explotar correlaciones locales sobre la grilla enlace–frecuencia, en contraste con un enfoque tabular que colapsa el snapshot a un vector.

Tabla 5.32: Definición de las matrices de entrada del DCNN según (Safari y cols., 2021b). Se reportan dimensiones, significado de L y F, y el contenido de las columnas topológicas y del bloque enlace–slot.

Componente	**Descripción**
Dimensión	$D_{i,1}, D_{i,2} \in \mathbb{R}^{L\times(2+F)}$
L	Número de enlaces unidireccionales de la topología (en el paper: 198 para CON y 112 para TID).
F	Número de *frequency slots* por enlace (en el paper: $F = 96$).
Columna 1	Longitud del enlace (`len`).
Columna 2	Número de spans del enlace (`nspan`).
Bloque $L \times F$ en $D_{i,1}$	BER de lightpaths activos; para slots/enlaces del LUT se utiliza un marcador (en el paper: -1).
Bloque $L \times F$ en $D_{i,2}$	Cardinalidad de modulación de lightpaths activos y del LUT.

Tabla 5.33: Resumen de variables representadas en cada canal del tensor de entrada. Cada canal incluye información topológica por enlace y un mapa enlace–slot; el canal 1 codifica BER (con marcador para el LUT) y el canal 2 codifica modulación.

Canal	**Contenido**
$D_{i,1}$	Dos columnas topológicas por enlace (`len`, `nspan`) + mapa enlace–slot con BER de lightpaths activos y un marcador para el LUT.
$D_{i,2}$	Dos columnas topológicas por enlace (`len`, `nspan`) + mapa enlace–slot con la cardinalidad de modulación de lightpaths activos y del LUT.

Elección de 70 buckets (escala comparable al paper). El trabajo base reporta datasets del orden de $\sim$ 740,000 muestras por topología y utiliza un split 70/20/10 para train/val/test (Safari y cols., 2021b). Para reproducir un escenario de escala comparable bajo el esquema de procesamiento por-bucket del pipeline, se entrenó el DCNN sobre 70 buckets, totalizando 700 000 muestras. La Tabla 5.34 contrasta explícitamente la referencia del paper con la réplica: se mantiene el mismo particionado y el mismo orden de magnitud en cantidad de muestras, permitiendo iteración y control de recursos sin alterar el procedimiento experimental (concatenación global + split 70/20/10).

Configuración de entrenamiento (réplica). El entrenamiento se ejecutó en CPU en el clúster (Slurm), utilizando Adam con `lr`$= 10^{-4}$ y `batch`$= 256$, en línea con la configuración reportada en (Safari y cols., 2021b). Se entrenó por 10 épocas y se guardó el mejor checkpoint según *val loss*. La Tabla 5.35 deja documentados estos parámetros para reproducibilidad y para interpretar costos

Tabla 5.34: Tamaño de dataset y split: referencia del paper vs. réplica en este trabajo. Se preserva el split 70/20/10 y se mantiene el mismo orden de magnitud de muestras mediante 70 buckets del pipeline.

Ítem	**Valor**
Referencia (paper)	∼740k muestras por topología (p. ej., CON: 738 318; TID: 743 540).
Split (paper)	70/20/10 (train/val/test).
Réplica (este trabajo)	70 buckets ⇒ 700 000 muestras; split 70/20/10 sobre el conjunto concatenado.

de ejecución al escalar el volumen de datos.

Tabla 5.35: Configuración del entrenamiento DCNN utilizada en la réplica. Se reportan épocas, tamaño de batch, optimizador, tasa de aprendizaje y criterio de checkpoint.

Parámetro	**Valor**
Épocas	10
Batch size	256
Optimizador	Adam
Learning rate	1×10^{-4}
Checkpoint	Mejor por *val loss*
Dispositivo	CPU
Semilla	42

Resultados (70 buckets). La evolución por época se resume en la Tabla 5.36. Se observa convergencia rápida: desde la primera época se alcanzan valores de *accuracy* cercanos a 0.999 tanto en entrenamiento como en validación, y la pérdida disminuye de forma sostenida con oscilaciones moderadas. El desempeño final en test se reporta en la Tabla 5.37: el mejor checkpoint (seleccionado por *val loss*) alcanzó *accuracy* = 0,9997 con *loss* = 0,0008.

Extensión a 152 buckets (entrenamiento a escala completa). Una vez validada la réplica sobre 70 buckets, se ejecutó la misma configuración sobre el conjunto completo disponible de tensores: 152 buckets (1,520,000 muestras). Se mantuvo el split 70/20/10, obteniéndose 1,064,000 muestras de entrenamiento, 304,000 de validación y 152,000 de test. La Tabla 5.38 resume de forma compacta ambas corridas (70 vs. 152 buckets), incluyendo tamaños de particiones y métricas finales. En esta escala, el modelo alcanzó nuevamente *accuracy* = 0,9997 con *loss* = 0,0005, sugiriendo ganancia marginal principalmente en la pérdida (asociable a calibración/confianza) más que en acierto.

Tabla 5.36: Evolución por época del DCNN entrenado con 70 buckets. Se reportan *loss* y *accuracy* en entrenamiento y validación para evidenciar convergencia y estabilidad del ajuste.

Epoch	Train loss	Train acc	Val loss	Val acc
01	0.0068	0.9978	0.0020	0.9992
02	0.0032	0.9986	0.0038	0.9981
03	0.0023	0.9991	0.0080	0.9972
04	0.0020	0.9992	0.0015	0.9993
05	0.0017	0.9993	0.0068	0.9974
06	0.0016	0.9995	0.0013	0.9994
07	0.0014	0.9995	0.0012	0.9995
08	0.0013	0.9995	0.0015	0.9993
09	0.0010	0.9996	0.0009	0.9995
10	0.0009	0.9997	0.0010	0.9995

Tabla 5.37: Resultado final en test para el DCNN (70 buckets), usando el mejor checkpoint por *val loss*. Se reportan pérdida y *accuracy* como métricas agregadas del desempeño final.

Métrica	Valor
Test loss	0.0008
Test accuracy	0.9997

Costo de ejecución a escala completa. Además del desempeño, es relevante documentar el costo computacional del entrenamiento a escala. La Tabla 5.39 reporta el tiempo total *wall-clock* para 152 buckets bajo la misma configuración de entrenamiento, lo que permite dimensionar el costo de entrenar un modelo convolucional de este tipo dentro del pipeline.

Conclusión al escalar (70 vs. 152 buckets). Al aumentar el volumen de entrenamiento desde 700 000 a 1 520 000 muestras, el desempeño en test se mantuvo prácticamente invariante (accuracy = 0,9997 en ambos casos), con una leve mejora en la pérdida (0.0008 $\rightarrow$ 0.0005), como se evidencia en la Tabla 5.38.

Tabla 5.38: Resumen de ejecución y desempeño del DCNN al escalar de 70 a 152 buckets. Se comparan tamaños de dataset, particionado train/val/test y métricas finales para cuantificar la ganancia marginal al incorporar más datos.

Corrida	#B	Total	Train/Val/Test	Loss	Acc
Réplica (escala paper)	70	700,000	489,999 / 140,000 / 70,001	0.0008	0.9997
Réplica (escala completa)	152	1,520,000	1,064,000 / 304,000 / 152,000	0.0005	0.9997

Tabla 5.39: Costo de ejecución reportado para el entrenamiento DCNN a escala completa (152 *buckets*). Se reporta el tiempo total *wall-clock* bajo CPU y el esquema de entrenamiento utilizado.

Ítem	**Valor**
Dispositivo	CPU
Épocas	10
Batch size	256
Tiempo total (*wall-clock*)	18:44:43

Esto sugiere que el modelo ya operaba en un régimen de alta separabilidad con 70 buckets (escala comparable a la reportada en (Safari y cols., 2021b)) y que, al incorporar más datos, la ganancia marginal se observa más en términos de calibración/confianza que en métricas de acierto. En términos de costo-beneficio, la corrida completa implica un costo computacional sustancial (Tabla 5.39) para una mejora cuantitativa acotada en métricas agregadas, por lo que el subconjunto de 70 buckets resulta una aproximación útil para iteración y validación del pipeline, mientras que los 152 buckets quedan como referencia de entrenamiento a escala y consolidación del resultado final.

La réplica confirma que, bajo la formulación de (Safari y cols., 2021b), el DCNN puede alcanzar desempeños muy altos al operar sobre una representación que preserva simultáneamente la dimensión enlace (L) y la dimensión frecuencia (F). En esta configuración, las convoluciones procesan patrones locales en la grilla enlace–slot y la red integra dicha información para producir una decisión binaria de QoT.

5.2.6. Conclusiones generales sobre Network Status

En conjunto, los resultados confirman que *network status* constituye un escenario **más desafiante** que *lightpath*: el problema está condicionado por el estado global de red, el desbalance y la presencia de señales de calidad que, si bien son físicamente relevantes, requieren un tratamiento metodológico cuidadoso para evitar conclusiones optimistas por fuga de información.

El baseline con Random Forest provee un primer ancla cuantitativa, destacando dos mensajes: (i) el desempeño en métricas sensibles a desbalance (AUC-PR) resulta informativo para este dominio, y (ii) el patrón FN$\gg$FP evidencia un tipo de error dominante (no detección de positivos) que guía decisiones posteriores (ajuste de umbrales, pesos, o elección de modelo). A partir de esta referencia se estructura la comparación con MLP y RNN, buscando balancear rendimiento predictivo, interpretabilidad y costo computacional dentro del pipeline reproducible propuesto.

5.2.7. Discusión sobre los resultados

En conjunto, la representación *network status* confirma que incorporar el **contexto global de la red** aporta señal útil para predecir QoT, pero también introduce desafíos que no aparecen (o aparecen atenuados) en *lightpath*. A diferencia del caso tabular por conexión, aquí cada ejemplo sintetiza un *snapshot* del sistema bajo una configuración de tráfico, y el desempeño termina estando fuertemente condicionado por (i) el **desbalance** de clases y (ii) la **cercanía** de ciertas variables a la forma en que se define la **variable objetivo** en el dataset.

Un primer baseline tabular (Random Forest) ofrece una referencia reproducible y permite identificar señales dominantes; sin embargo, el patrón de errores sugiere que la performance efectiva depende de cómo se fijan umbrales y de qué costo relativo se asigna a falsos positivos vs. falsos negativos. En este sentido, la lectura no debería apoyarse únicamente en *accuracy*: en un escenario desbalanceado, métricas orientadas a precisión/recall y la matriz de confusión aportan una interpretación más honesta del comportamiento del clasificador.

El segundo baseline tabular (MLP) deja una señal metodológica clara: bajo la configuración actual, el modelo puede **colapsar hacia una predicción casi constante**, alineando el *accuracy* con la prevalencia de la clase mayoritaria sin capturar adecuadamente la clase minoritaria. Esta observación refuerza la necesidad de incorporar una estrategia explícita de balance (en el criterio de entrenamiento o en el muestreo) antes de extraer conclusiones comparativas finas entre arquitecturas.

Finalmente, el baseline estructurado (RNN) se plantea como un primer paso para explotar la organización interna del *snapshot* (tratándolo como secuencia por enlace), a costa de mayor complejidad de ingeniería (artefactos GOLD secuenciales y exportación a tensores). Más allá del resultado puntual, este modelo cumple el rol de **puente** entre un consumo totalmente agregado (tabular) y enfoques que preservan estructura, y ayuda a justificar por qué en *network status* la elección de representación y la estrategia de cómputo pasan a ser parte del problema.

5.3. Conclusiones parciales de la experimentación

Este capítulo estableció líneas base reproducibles para estimación de QoT sobre las dos representaciones del dataset (*lightpath* y *network status*), comparando modelos tabulares y modelos con sesgo inductivo estructurado. En lugar de repetir el análisis por modelo, a continuación se sintetizan las conclusiones parciales más relevantes para interpretar los resultados y orientar el trabajo posterior.

Hallazgos principales

- **Lightpath como baseline sólido e interpretable.** La representación *lightpath* se comporta como un problema tabular altamente informativo: permite desempeño muy alto en clasificación de QoT y un ajuste excelente en regresión de métricas físicas (particularmente OSNR). Además, las importancias de variables son coherentes con la lectura física del problema (configuración y "complejidad" del trayecto), por lo que resulta una base confiable y de bajo costo para comparación.

- **Network Status como escenario más exigente.** En *network status* el desempeño está más condicionado por (i) el desbalance de clases, (ii) la forma de particionado/evaluación y (iii) el control de fuga de información. Los baselines confirman que el resultado agregado (accuracy) puede ser engañoso si no se analiza el patrón de errores y métricas robustas al desbalance.

- **Patrones de error y trade-offs operativos.** Los modelos evaluados exhiben perfiles de error distintos (por ejemplo, sesgo hacia FN o hacia FP). En un uso operativo, estos perfiles se traducen en decisiones sobre umbral, ponderación de clases o elección del modelo según el objetivo: minimizar falsas alarmas versus evitar degradaciones no detectadas.

Limitaciones y próximos pasos

- **Comparación controlada entre modelos en Network Status.** Repetir RF/MLP/RNN bajo el mismo conjunto de buckets y el mismo protocolo de particionado para aislar el efecto del modelo de la variabilidad del muestreo.

- **Robustez ante leakage.** Ejecutar un *ablation study* removiendo señales cercanas al outcome (y/o variables derivadas) para cuantificar cuánto del desempeño se sostiene con información "más lejana" al label.

- **Tratamiento explícito del desbalance.** Evaluar estrategias compatibles con el entorno de entrenamiento (ponderación, muestreo balanceado y/o calibración de umbral) y reportar métricas centradas en precisión/recall además de accuracy.

Finalmente, estas conclusiones parciales se utilizan como base para las conclusiones globales del trabajo (Capítulo 7), donde se integran los hallazgos con las decisiones de ingeniería del pipeline y la infraestructura de ejecución.

Capítulo 6

Ingeniería de software

Este capítulo presenta los principales aspectos de ingeniería de software asociados al desarrollo del proyecto. A diferencia de los capítulos previos, centrados en el dominio del problema, la ingeniería de datos y la experimentación, aquí se abordan las decisiones metodológicas y organizativas que sustentaron el proceso de desarrollo.

En particular, se describen las herramientas y entornos utilizados, la forma en que se planificó y gestionó el proyecto, y las prácticas adoptadas para asegurar la corrección, reproducibilidad y trazabilidad de los resultados. El objetivo no es detallar implementaciones específicas, sino contextualizar el marco de trabajo que permitió estructurar un flujo de desarrollo ordenado y consistente con las características del problema y las restricciones del entorno computacional.

6.1. Herramientas y entorno de desarrollo

Para el desarrollo del trabajo se utilizaron herramientas orientadas a facilitar la iteración rápida sobre el pipeline y los experimentos, así como a mantener un flujo de colaboración ordenado entre los integrantes del equipo y los tutores.

En cuanto a lenguajes, se empleó principalmente *Python*, dada su amplia adopción en proyectos de *machine learning* y su ecosistema de bibliotecas para análisis y procesamiento de datos. Como complemento, se utilizó *Bash* para tareas de *scripting* asociadas a la ejecución y automatización de procesos.

El desarrollo comenzó de forma local en las computadoras de los autores, utilizando *Docker* como base para asegurar consistencia del entorno. Una vez disponible la infraestructura correspondiente, el flujo de trabajo fue migrado a una máquina virtual, manteniendo la misma lógica de ejecución y reduciendo fricciones asociadas a diferencias entre entornos.

Para la coordinación y comunicación, se utilizó *Mattermost* como canal oficial con los tutores y *Zoom* para reuniones de seguimiento semanales. En el trabajo diario entre los integrantes del equipo se utilizó principalmente *WhatsApp*, complementado con *Google Meet* para reuniones de coordinación. En paralelo,

se utilizó *Notion* como espacio de referencia para centralizar documentación del proyecto y organizar el material relevado (trabajos previos, *papers* y *surveys*) a lo largo del proceso.

Finalmente, para control de versiones y seguimiento del desarrollo se utilizó *GitLab* como plataforma central del repositorio del proyecto.

6.2. Gestión del proyecto

La gestión del proyecto se organizó a partir de una planificación por etapas y un seguimiento semanal con los tutores. Como insumo de planificación, se mantuvo un registro cronológico de actividades e hitos (reuniones, decisiones relevantes, entregables y avances técnicos), que permitió reconstruir el recorrido completo del trabajo y, a partir de él, definir un cronograma sintético. Este enfoque facilitó ordenar prioridades, explicitar dependencias entre tareas (por ejemplo, disponibilidad de infraestructura o cierre de una etapa del pipeline antes de escalar experimentos) y sostener un avance medible a lo largo del período de desarrollo.

6.2.1. Planificación y cronograma

La planificación se estructuró en cuatro etapas principales: *investigación teórica*, *herramientas e infraestructura*, *desarrollo/experimentación* y *documentación*. Si bien estas etapas presentan solapamientos naturales (por ejemplo, la documentación acompaña cierres parciales y la investigación teórica continúa ante hallazgos técnicos), el desglose permite sintetizar el trabajo en una línea temporal, identificar hitos y visualizar dependencias. El diagrama de Gantt de la Figura 6.1 resume esta organización, indicando duraciones aproximadas y puntos de control.

A modo de síntesis, las etapas consideradas son:

- **Investigación teórica (*03/2025–10/2025*).** Estudio del marco conceptual del problema de QoT, revisión de literatura y lectura de documentación asociada al dataset y a las técnicas de modelado consideradas. Esta etapa incluye instancias de validación conceptual con tutores y ajustes del alcance.
- **Herramientas e infraestructura (*05/2025–12/2025*).** Preparación y estabilización del entorno de trabajo (desarrollo local y posterior migración a infraestructura disponible), definición del flujo de ejecución y verificación de que las herramientas soportan el volumen y la estructura del dataset.
- **Desarrollo y experimentación (*08/2025–01/2026*).** Implementación del pipeline de datos y generación de artefactos intermedios, junto con la ejecución de experimentos de modelado bajo un protocolo consistente (particionado, métricas y criterios de comparación), iterando sobre mejoras y validaciones.

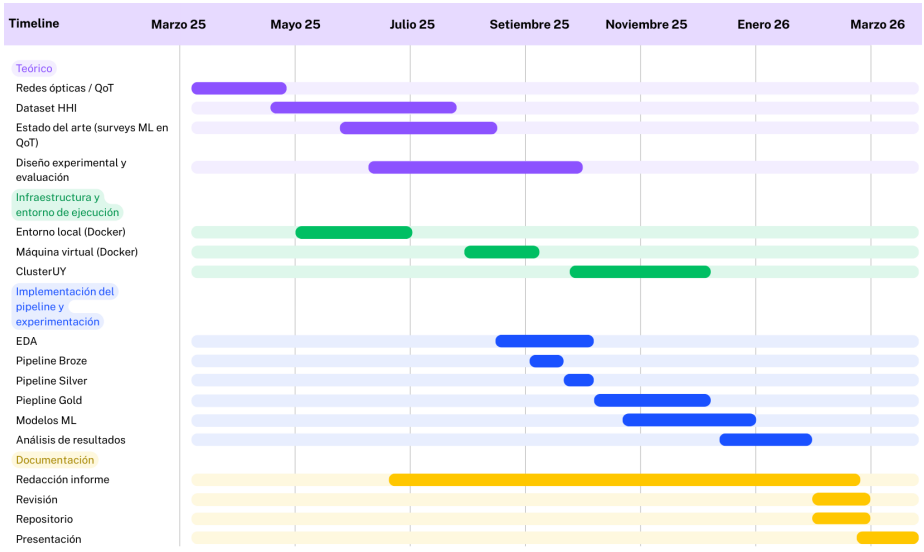


Figura 6.1: Diagrama de Gantt del proyecto, organizado en etapas principales e hitos de seguimiento.

- **Documentación y cierre (*06/2025–02/2026*).** Redacción incremental del informe, integración de resultados y ajustes de estructura en función de devoluciones, hasta consolidar una versión final con trazabilidad entre decisiones, pipeline y evidencia experimental.

El seguimiento del cronograma se realizó mediante reuniones periódicas con los tutores, utilizadas para revisar el avance, validar decisiones relevantes y acordar objetivos de corto plazo. El Gantt se emplea como síntesis de este proceso, más que como una planificación rígida, reflejando un enfoque iterativo con ajustes a medida que se consolidaban resultados y se identificaban nuevas necesidades.

6.2.2. Estrategia de desarrollo

El desarrollo se llevó adelante con un enfoque incremental e iterativo, priorizando validaciones tempranas antes de escalar el procesamiento y la experimentación. En una primera etapa, se trabajó en forma local utilizando Docker para asegurar consistencia del entorno y habilitar iteración rápida. Una vez disponible la infraestructura correspondiente, el flujo se migró a una máquina virtual, manteniendo la misma lógica de ejecución y reduciendo fricciones por diferencias entre entornos.

A nivel de organización del trabajo, se buscó construir un *camino mínimo de punta a punta* (ingesta → transformaciones → dataset listo para modelado →

entrenamiento/evaluación) y luego refinarlo por capas, incorporando controles y mejoras de manera progresiva. Este criterio permitió desacoplar actividades de alto costo computacional del ciclo de exploración y ajuste, y facilitó aislar problemas cuando surgían cambios en datos, transformaciones o configuraciones. El seguimiento semanal con tutores funcionó como mecanismo de control: se revisaban avances, se validaban decisiones relevantes y se re-priorizaban objetivos de corto plazo cuando aparecían nuevos hallazgos o restricciones.

6.3. Calidad y reproducibilidad

Dado que el trabajo combina procesamiento de datos a escala y experimentación con modelos, se adoptaron prácticas orientadas a asegurar la corrección de los artefactos intermedios y la reproducibilidad de las corridas. En primer lugar, el uso de Docker y la posterior migración controlada a la VM contribuyeron a mantener un entorno consistente, minimizando diferencias de dependencias o configuración entre ejecuciones. En segundo lugar, el desarrollo se apoyó en el versionado mediante GitLab, permitiendo trazar cambios en scripts, configuraciones y ajustes del pipeline a lo largo del tiempo.

Adicionalmente, se procuró que cada etapa del flujo produjera salidas verificables (materializaciones intermedias y datasets derivados), lo que facilitó repetir corridas bajo condiciones equivalentes y acotar la fuente de desvíos cuando se modificaban transformaciones o parámetros. En conjunto, estas decisiones permitieron sostener un proceso de trabajo ordenado, con resultados comparables entre iteraciones y con trazabilidad entre implementaciones, ejecuciones y evidencia reportada en el informe.

Capítulo 7

Conclusiones y Trabajo Futuro

Este capítulo cierra el informe con una síntesis de los principales resultados y aprendizajes obtenidos a lo largo del trabajo. En primer lugar, se presentan las conclusiones más relevantes a partir de la evidencia experimental y de las decisiones de ingeniería adoptadas. Luego, se discuten las limitaciones y aspectos que condicionaron el alcance del estudio, junto con una breve reflexión crítica sobre el proceso. Finalmente, se proponen líneas de trabajo futuro que permitirían profundizar y extender lo realizado.

7.1. Conclusiones principales

A partir de la evidencia obtenida, se sintetizan a continuación las conclusiones más relevantes del trabajo:

- **Las dos representaciones estudiadas resultan complementarias y definen un compromiso entre simplicidad y expresividad.** *Lightpath* ofrece una caracterización directa y eficiente que funciona como referencia sólida, mientras que *Network Status* incorpora información de contexto de red con mayor potencial descriptivo, a costa de exigir mayor cuidado metodológico y computacional.

- **El beneficio de modelos más complejos depende de controlar estrictamente el diseño experimental.** Las mejoras no se vuelven sistemáticas si no se mantienen condiciones comparables entre corridas (criterio de particionado, reglas de preprocesamiento, variables elegibles y búsqueda de hiperparámetros). En este sentido, la calidad del diseño experimental determina si una diferencia de desempeño es atribuible al modelo o a cambios en los datos/protocolo de evaluación.

- **La evaluación debe reflejar el objetivo operativo, no solo un promedio global.** En presencia de desbalance, métricas agregadas pueden ocultar comportamientos relevantes por clase. Por lo tanto, resulta clave reportar **precisión/recall** (y métricas balanceadas) y analizar explícitamente el intercambio entre falsos positivos y falsos negativos. El ajuste de umbral aparece como una herramienta simple para alinear el clasificador con prioridades operativas concretas.

- **La variabilidad física del canal óptico se refleja en patrones consistentes en los datos.** Más allá del modelo utilizado, el análisis confirma un aprendizaje de dominio: la calidad de transmisión está fuertemente condicionada por factores ligados a la ruta y la acumulación de degradaciones (distancia efectiva recorrida, cantidad de tramos/equipamiento intermedio y condiciones del espectro/ocupación). Esto refuerza que la QoT no depende de un único "parámetro crítico", sino del efecto acumulativo de múltiples componentes del trayecto y del contexto de red.

- **Las decisiones de ingeniería fueron habilitadoras para la reproducibilidad y la iteración.** La adopción de un flujo por capas (materializaciones intermedias) permitió desacoplar transformaciones costosas del ciclo de entrenamiento/evaluación y sostuvo la repetición de corridas bajo condiciones comparables. El EDA cumplió además un rol instrumental al anticipar costos reales de transformaciones y orientar reglas de limpieza/validación, mientras que la infraestructura de ejecución resultó determinante para viabilizar experimentos a escala.

En conjunto, el trabajo muestra que los resultados dependen del acople entre representación, control metodológico, métricas y decisiones de ingeniería para procesar los datos: una representación más simple puede ser altamente efectiva como base comparativa, y una representación más rica puede aportar información adicional siempre que se gestione con disciplina el costo y los riesgos asociados.

7.2. Aportes y alcance del trabajo

Este trabajo deja como resultado un conjunto de aportes reutilizables y, a la vez, delimita el marco dentro del cual deben interpretarse los resultados reportados.

Aportes. El aporte central es la construcción de un pipeline de datos reproducible, organizado por capas y con materializaciones intermedias, que desacopla transformaciones costosas del ciclo iterativo de experimentación. Sobre esa base se obtienen datasets derivados listos para modelado, acompañados por criterios explícitos de limpieza y validación, así como scripts y configuraciones de ejecución que permiten repetir corridas en los entornos utilizados bajo condiciones

equivalentes; estos artefactos se encuentran versionados en el repositorio del proyecto (Optical Networks ML, 2026). Complementariamente, el trabajo define y aplica un protocolo experimental consistente—incluyendo particionado, métricas y criterios de comparación—que habilita evaluar enfoques bajo un control metodológico uniforme. Finalmente, se aporta evidencia empírica y aprendizajes sobre el compromiso entre representación de los datos, costo computacional y robustez metodológica, útiles para orientar iteraciones futuras del pipeline y del diseño experimental; en particular, las limitaciones identificadas en limpieza, validación y escalado del pipeline constituyen puntos de entrada concretos para mejoras en trabajos subsiguientes.

Alcance. El estudio se restringe a (i) los datos y representaciones considerados en el informe, (ii) un conjunto acotado de modelos y estrategias de entrenamiento y evaluación, y (iii) el presupuesto de cómputo disponible para explorar configuraciones e hiperparámetros. En consecuencia, quedan fuera del alcance una exploración exhaustiva del espacio de modelos y arquitecturas, la aplicación de validación cruzada a gran escala, un análisis sistemático de generalización a escenarios adicionales, y la integración operacional del enfoque en un sistema en línea de provisión o monitoreo.

7.3. Limitaciones y lecciones aprendidas

Si bien los resultados son informativos, su interpretación queda condicionada por el alcance efectivo del estudio. En particular, el presupuesto de cómputo y el tiempo disponible limitaron la exploración sistemática del espacio de hiperparámetros y la repetición de corridas bajo múltiples configuraciones, lo que reduce la posibilidad de afirmar tendencias finas más allá de los patrones dominantes observados. Además, en un problema con desbalance y alta sensibilidad al protocolo, variaciones en el particionado, en el preprocesamiento o en el conjunto de variables consideradas pueden impactar las conclusiones, especialmente en la representación *Network Status*. Finalmente, aunque se tomaron precauciones, el riesgo de fuga de información (por variables que anticipan directa o indirectamente la etiqueta) sugiere la necesidad de análisis adicionales para fortalecer la robustez de las conclusiones.

Como aprendizaje general, el proceso reforzó la importancia de priorizar **comparabilidad y trazabilidad** antes de optimizar modelos: fijar un protocolo estable, documentar reglas de preparación y materializar datasets intermedios reduce ambigüedades y facilita iteraciones confiables. Del mismo modo, el desbalance y la posible fuga de información no deben tratarse como detalles periféricos, sino como aspectos centrales del diseño experimental, ya que condicionan tanto las métricas reportadas como la interpretación práctica de los resultados.

7.4. Trabajo futuro

A partir de lo realizado, se abren múltiples líneas de trabajo futuro para profundizar y extender el alcance del estudio. Estas líneas son complementarias y pueden abordarse en paralelo según el foco del trabajo.

7.4.1. Datos y pipeline

Como continuación natural del trabajo, resulta relevante fortalecer el pipeline construido, tanto en alcance como en eficiencia. Por un lado, el conjunto de HHI incluye **múltiples datasets/escenarios adicionales** que no fueron explorados en este informe; extender el flujo a esos casos permitiría evaluar en qué medida los patrones observados se sostienen, y comparar el comportamiento de modelos y representaciones bajo condiciones de red diferentes, manteniendo un criterio común de preparación y evaluación.

Por otro lado, existe margen para optimizar la generación de las distintas capas (*Bronze–Silver–Gold*). Esto incluye revisar y mejorar transformaciones costosas, ajustar estrategias de particionado y materialización para reducir recomputaciones y *shuffle*, y sistematizar validaciones de calidad que detecten valores anómalos o inconsistencias temprano en el proceso. Estas mejoras no solo reducen el costo de cómputo, sino que también hacen más fluida la iteración experimental al contar con datasets intermedios estables y trazables.

Adicionalmente, una extensión relevante sería validar el pipeline y los modelos entrenados sobre datos reales provenientes de redes en operación, en lugar de datos sintéticos de simulación. Esto permitiría evaluar en qué medida los patrones aprendidos generalizan a condiciones de despliegue real, donde el ruido de medición, la variabilidad operacional y la heterogeneidad de equipos introducen desafíos adicionales no presentes en el dataset de referencia.

7.4.2. Modelado y experimentación

En términos de modelado, queda margen para profundizar la exploración con un *tuning* más sistemático y comparaciones bajo un protocolo estrictamente controlado, de modo de aislar el efecto del algoritmo de variaciones en datos y preprocesamiento. En particular, resulta relevante reforzar el tratamiento del desbalance (tanto en entrenamiento como en evaluación) y estudiar con mayor detalle la calibración de probabilidades y el ajuste de umbrales, dado su impacto directo en el compromiso entre falsos positivos y falsos negativos.

Asimismo, para la representación *Network Status* se abren líneas específicas: explorar alternativas de representación que capturen mejor su estructura (y reduzcan dimensionalidad cuando corresponda), evaluar arquitecturas o enfoques que aprovechen explícitamente esa organización, y estudiar qué combinaciones de variables aportan información útil sin introducir sesgos por variables que anticipen la etiqueta.

7.4.3. Análisis e interpretación

Más allá del desempeño, una extensión relevante es profundizar la **interpretación de qué aprende el modelo** desde el punto de vista del dominio. Esto incluye analizar la contribución relativa de variables asociadas a ruta, ocupación y degradaciones acumuladas, y contrastar esos hallazgos con intuiciones físicas del canal óptico. Este tipo de análisis aporta explicaciones accionables y permite transformar el modelo en una herramienta no solo predictiva, sino también de comprensión del fenómeno.

Referencias

Akiba, T., Sano, S., Yanase, T., Ohta, T., y Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. En *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623–2631). ACM. doi: 10.1145/3292500.3330701

Apache hadoop. (s.f.). https://hadoop.apache.org/. (Accessed: 2026-02-15)

Apache parquet. (s.f.). https://parquet.apache.org/. (Accessed: 2026-02-15)

Apache spark documentation. (s.f.). https://spark.apache.org/docs/latest/. (Accessed: 2026-02-15)

Ayassi, R., Triki, A., Crespi, N., Minerva, R., y Laye, M. (2022). Survey on the use of machine learning for quality of transmission estimation in optical transport networks. *Journal of Lightwave Technology*, *40*(17), 5803–5815. doi: 10.1109/JLT.2022.3184178

Bergk, G., Shariati, B., Safari, P., y Fischer, J. K. (2022a, marzo). ML-assisted QoT estimation: a dataset collection and data visualization for dataset quality evaluation. *Journal of Optical Communications and Networking*, *14*(3), 43–55. doi: 10.1364/JOCN.442733

Bergk, G., Shariati, B., Safari, P., y Fischer, J. K. (2022b, marzo). ML-assisted QoT estimation: a dataset collection and data visualization for dataset quality evaluation. *Journal of Optical Communications and Networking*, *14*(3), 43–55. doi: 10.1364/JOCN.442733

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. doi: 10.1023/A:1010933404324

Captum API: IntegratedGradients. (s.f.). https://captum.ai/api/integrated_gradients.html. (Accessed: 2026-02-07)

Captum: Model interpretability for PyTorch. (s.f.). https://captum.ai/. (Accessed: 2026-02-07)

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., y Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. En *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. doi: 10.3115/v1/D14-1179

Clemente, F., Ribeiro, G. M., Quemy, A., Santos, M. S., Pereira, R. C., y Barros, A. (2023). ydata-profiling: Accelerating data-centric AI with high-quality data. *Neurocomputing*, *554*, 126585. doi: 10.1016/j.neucom.2023.126585

ClusterUY. (s.f.). https://cluster.uy/. (Accessed: 2026-02-01)

ClusterUY. (s.f.). *ClusterUY (sitio oficial)*. https://www.cluster.uy/. (Descripción institucional del servicio de cómputo de alto desempeño. Accessed: 2026-02-06)

Databricks. (2020). *Medallion architecture*. https://www.databricks.com/glossary/medallion-architecture. (Accessed: 2026-02-01)

de Lima, L. A., y Pavani, G. S. (2021). Provisioning and recovery in flexible optical networks using ant colony optimization. En *2021 IFIP/IEEE international symposium on integrated network management (im)* (pp. 677–681). Descargado de https://opendl.ifip-tc6.org/db/conf/im/im2021short/211936.pdf

Di Tommaso, P., Chatzou, M., Floden, E. W., Prieto Barja, P., Palumbo, E., y Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, *35*(4), 316–319. doi: 10.1038/nbt.3820

Docker overview. (s.f.). https://docs.docker.com/get-started/docker-overview/. (Accessed: 2026-02-15)

Fraunhofer Heinrich-Hertz-Institut (HHI). (s.f.). *QoT dataset collection — PLATON (planning tool for optical networks)*. https://www.hhi.fraunhofer.de/en/pn-software/qot-dataset-collection.html. (Accessed: 2026-02-15)

Fraunhofer Heinrich Hertz Institute (HHI). (s.f.). *QoT dataset collection*. https://www.hhi.fraunhofer.de/en/pn-software/qot-dataset-collection.html. (Accessed: 2026-01-31)

Fundación Julio Ricaldoni. (2018). *Supercomputación en uruguay*. https://www2.ricaldoni.org.uy/noticias?start=130. (Noticia institucional sobre el lanzamiento del Centro Nacional de Supercomputación (ClusterUY). Accessed: 2026-02-06)

HDFS architecture guide. (s.f.). https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. (Accessed: 2026-02-15)

HDFS users guide. (s.f.). https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. (Accessed: 2026-02-15)

Hochreiter, S., y Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Huyen, C. (s.f.). *Design a machine learning system*. https://huyenchip.com/machine-learning-systems-design/design-a-machine-learning-system.html. (Accessed: 2026-02-15)

Jupyter documentation. (s.f.). https://docs.jupyter.org/. (Accessed: 2026-02-15)

Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. Sebastopol, CA: O'Reilly Media.

Köster, J., y Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520–2522. doi: 10.1093/bioinformatics/bts480

López, V., y Velasco, L. (Eds.). (2016). *Elastic optical networks: Architectures, technologies, and control*. Cham: Springer International Publishing. doi: 10.1007/978-3-319-30174-7

Mata, J., de Miguel, I., Durán, R. J., Merayo, N., Singh, S. K., Jukan, A., y Chamania, M. (2018). Artificial intelligence (AI) methods in optical networks: A comprehensive survey. *Optical Switching and Networking*, *28*, 43–57. doi: 10.1016/j.osn.2017.12.006

Musumeci, F., Rottondi, C., Nag, A., Macaluso, I., Zibar, D., Ruffini, M., y Tornatore, M. (2019). An overview on application of machine learning techniques in optical networks. *IEEE Communications Surveys & Tutorials*, *21*(2), 1383–1408. doi: 10.1109/COMST.2018.2880039

NCR, DaimlerChrysler, y SPSS. (1999). *CRoss-Industry Standard Process Model for Data Mining (CRISP-DM), discussion paper 05-03-99.* https://keithmccormick.com/wp-content/uploads/CRISP-DM%20No%20Brand.pdf. (Accessed: 2026-02-15)

Olmedo Guillama, S. (2024). *Aprendizaje automático aplicado al dominio de las redes ópticas* (Tesis de grado). Universidad de la República (Uruguay), Facultad de Ingeniería, Instituto de Computación, Montevideo, Uruguay. (Disponible en Colibri (Udelar))

Optical Networks ML. (2026). *optical-networks-ml: Repositorio del proyecto (código, scripts e infraestructura).* GitLab, Facultad de Ingeniería (Universidad de la República), https://gitlab.fing.edu.uy/mobile-optical-networks/optical-networks-ml. (Accessed: 2026-02-21)

Optuna: A hyperparameter optimization framework. (s.f.). https://optuna.org/. (Accessed: 2026-02-07)

Optuna documentation: Pruning unpromising trials. (s.f.). https://optuna.readthedocs.io/en/stable/tutorial/10_key_features/003_efficient_optimization_algorithms.html. (Accessed: 2026-02-07)

Overview — apache parquet. (s.f.). https://parquet.apache.org/docs/overview/. (Accessed: 2026-02-15)

Overview — Slurm workload manager. (s.f.). https://slurm.schedmd.com/overview.html. (Accessed: 2026-02-15)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... others (2019). PyTorch: An imperative style, high-performance deep learning library. En *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc.

Phase-shift keying. (s.f.). https://en.wikipedia.org/wiki/Phase-shift_keying. (Accessed: 2026-02-15)

PyTorch. (s.f.). https://pytorch.org/. (Accessed: 2026-02-07)

Quadrature amplitude modulation. (s.f.). https://en.wikipedia.org/wiki/Quadrature_amplitude_modulation. (Accessed: 2026-02-15)

Rafique, D., y Velasco, L. (2018). Machine learning for network automation: Overview, architecture, and applications. *Journal of Optical Communications and Networking*, *10*(10), D126–D143. doi: 10.1364/JOCN.10.00D126

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning repre-

sentations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi: 10.1038/323533a0

Safari, P., Shariati, B., Bergk, G., y Fischer, J. K. (2021a). Deep convolutional neural network for network-wide QoT estimation. En *2021 optical fiber communications conference and exhibition (ofc)*. San Francisco, CA, USA: IEEE. doi: 10.1364/OFC.2021.W1G.2

Safari, P., Shariati, B., Bergk, G., y Fischer, J. K. (2021b). Deep convolutional neural network for network-wide QoT estimation. En *2021 optical fiber communications conference and exhibition (ofc)*. San Francisco, CA, USA: IEEE. doi: 10.1364/OFC.2021.W1G.2

Safari, P., Shariati, B., Bergk, G., y Fischer, J. K. (2021c). Secure multi-party computation and statistics sharing for ML model training in multi-domain multi-vendor networks. En *Proceedings of the european conference on optical communication (ecoc)*. IEEE.

Slurm workload manager: Documentation. (s.f.). https://slurm.schedmd.com/documentation.html. (Accessed: 2026-02-01)

Spark SQL & DataFrames. (s.f.). https://spark.apache.org/sql/. (Accessed: 2026-02-15)

Villa, G., Tipantuña, C., Guamán, D. S., Arévalo, G. V., y Arguero, B. (2023). Machine learning techniques in optical networks: A systematic mapping study. *IEEE Access*, *11*, 98714–98750. doi: 10.1109/ACCESS.2023.3312387

What is a container? (s.f.). https://docs.docker.com/get-started/docker-concepts/the-basics/what-is-a-container/. (Accessed: 2026-02-15)

Anexo A

Herramientas e infraestructura computacional

Este apéndice describe el entorno experimental y las herramientas utilizadas para implementar el *pipeline* de datos y ejecutar los experimentos de aprendizaje automático. El objetivo es fijar la terminología y dejar documentados los componentes tecnológicos relevantes para la reproducibilidad: contenerización (del inglés containerization), almacenamiento y procesamiento distribuido, formatos de persistencia, entorno interactivo, ejecución en HPC y bibliotecas de modelado.[1]

A.1. Docker: contenerización

Docker es una plataforma para construir, distribuir y ejecutar aplicaciones empaquetadas en *contenedores*. Conceptualmente, un contenedor es una unidad de ejecución aislada a nivel de procesos que agrupa una aplicación con sus dependencias, de manera que el software se ejecute de forma consistente entre distintos entornos.

La documentación oficial enfatiza esta idea al describir que Docker permite "empaquetar y ejecutar una aplicación en un entorno aislado (de manera flexible) llamado contenedor" y que un contenedor puede entenderse como "un proceso aislado con todos los archivos que necesita para ejecutarse" (*Docker overview*, s.f.; *What is a container?*, s.f.). En contraste con la virtualización tradicional, se destaca que múltiples contenedores pueden compartir el mismo *kernel*, reduciendo sobrecarga y facilitando densidad de despliegue.

[1] Las páginas oficiales citadas se encuentran, en su mayoría, publicadas en inglés; las citas textuales entre comillas se presentan en español.

Además, Docker formaliza una distinción útil en términos de ingeniería: una *imagen* como artefacto inmutable (con capas de sistema y dependencias) y un *contenedor* como instancia en ejecución. Este modelo promueve reproducibilidad técnica: a igual imagen, el entorno de ejecución se mantiene estable, lo que simplifica transferencia y validación de resultados entre equipos y máquinas.

En este trabajo se utiliza Docker para empaquetar y desplegar el entorno de prototipado (Spark, HDFS y notebooks) en una VM, asegurando ejecuciones reproducibles del *pipeline* a pequeña escala.

A.2. Apache Hadoop: ecosistema para datos a gran escala

Apache Hadoop es un ecosistema de software orientado al almacenamiento y procesamiento distribuido de grandes volúmenes de datos en clústeres. En términos generales, se concibe como una plataforma que permite escalar el cómputo y el almacenamiento desde servidores individuales hasta conjuntos extensos de nodos, manteniendo una visión unificada del sistema (*Apache Hadoop*, s.f.).

A nivel conceptual, Hadoop se asocia a un principio arquitectónico típico en *Big Data*: aprovechar paralelismo a nivel de clúster y acercar el cómputo al dato. También es relevante notar que Hadoop no es una única "aplicación", sino un conjunto de componentes que se integran: *Hadoop Distributed File System* (HDFS) para almacenamiento distribuido, *Yet Another Resource Negotiator* (YARN) para administración de recursos y ejecución, y MapReduce como modelo clásico de procesamiento por lotes. Esta modularidad ayuda a entender por qué, en arquitecturas actuales, Hadoop suele coexistir con motores de procesamiento alternativos como Spark, manteniendo interoperabilidad sobre datos y servicios del mismo ecosistema.

Aquí se adopta Hadoop principalmente como ecosistema base para almacenamiento distribuido (HDFS) e interoperabilidad, sobre el cual se ejecuta el procesamiento con Spark.

A.3. HDFS: sistema de archivos distribuido

HDFS (*Hadoop Distributed File System*) es el sistema de archivos distribuido asociado al ecosistema Hadoop, diseñado para almacenar archivos grandes y servirlos con alto rendimiento (*throughput*) sobre clústeres. A nivel conceptual, su finalidad es ofrecer una abstracción de sistema de archivos donde los datos se distribuyen en múltiples nodos, habilitando acceso paralelo y tolerancia a fallos (*HDFS Users Guide*, s.f.; *HDFS Architecture Guide*, s.f.).

En una arquitectura típica se distinguen roles bien definidos: el *NameNode* administra el espacio de nombres y los metadatos del sistema de archivos, mientras que los *DataNodes* almacenan los bloques y sirven datos a clientes y motores de cómputo. Dos ideas que resultan centrales: (i) la partición de los archivos en *bloques* distribuidos, que habilita paralelismo en lectura y escritura,

y (ii) la tolerancia a fallos, donde la replicación y la reconfiguración ante fallas forman parte del funcionamiento esperado del sistema.

En el entorno de prototipado, HDFS se utiliza como capa de persistencia para las zonas *bronze/silver/gold*, permitiendo almacenar y acceder en paralelo a particiones (buckets/chunks) durante las conversiones y la validación experimental.

A.4. Apache Spark: motor unificado de procesamiento

Apache Spark es un motor general de procesamiento y analítica para ejecutar flujos de trabajo de datos sobre una máquina o un clúster. A nivel conceptual, combina un modelo de programación de alto nivel (APIs) con un motor de ejecución que planifica y optimiza transformaciones, lo que permite expresar *pipelines* complejos sobre datos estructurados y no estructurados (*Apache Spark Documentation*, s.f.).

Un rasgo importante es su compatibilidad con el ecosistema Hadoop: puede ejecutarse sobre distintos gestores de recursos y procesar datos almacenados en HDFS y otros sistemas. En términos funcionales, Spark cubre tareas de ETL y analítica por lotes, y puede integrar bibliotecas para consultas y preparación de datos, lo que lo vuelve un motor multipropósito en entornos de ingeniería de datos (*Spark SQL & DataFrames*, s.f.).

En este trabajo, Spark se emplea para implementar el *pipeline* de transformación (lectura, limpieza, agregación y generación de *features*) y para materializar datasets a escala mediante ejecuciones batch por-bucket.

A.5. Apache Parquet: formato de archivo columnar

Apache Parquet es un formato de archivo de datos orientado a columnas, diseñado para almacenamiento y recuperación eficientes. Su motivación principal se vincula a cargas analíticas: organizar los datos por columnas favorece compresión efectiva y reduce lectura cuando solo se requieren algunas variables, mejorando eficiencia de E/S en *pipelines* de datos (*Apache Parquet*, s.f.; *Overview — Apache Parquet*, s.f.).

Desde el punto de vista conceptual, Parquet aporta dos beneficios: (i) eficiencia por diseño para operaciones analíticas (compresión y lectura selectiva por columna) y (ii) interoperabilidad entre motores, desacoplando almacenamiento persistente del motor de ejecución.

En este trabajo, Parquet es el formato de persistencia principal para las tablas derivadas (silver/gold), por su eficiencia en lectura selectiva y su buen acople con Spark y bibliotecas de ML.

A.6. Project Jupyter: notebooks para computación interactiva

Project Jupyter provee herramientas para computación interactiva basada en notebooks. Un notebook es un documento ejecutable que combina código, texto y resultados, y funciona como soporte natural para exploración, explicación y comunicación técnica en ciencia de datos y computación científica (*Jupyter Documentation*, s.f.).

A nivel conceptual, Jupyter separa la interfaz del mecanismo de ejecución mediante *kernels*. La interfaz gestiona el documento (celdas, salidas, visualizaciones), mientras el kernel ejecuta el código y devuelve resultados, permitiendo flujos iterativos y experimentación rápida.

En este trabajo, Jupyter se utiliza para EDA, validaciones rápidas de calidad de datos y prototipado de etapas del *pipeline* antes de su ejecución batch en el clúster.

A.7. ClusterUY y SLURM: HPC y planificación de trabajos

ClusterUY es una plataforma de computación de alto desempeño (HPC) en Uruguay orientada a investigación y desarrollo. En infraestructuras HPC, la ejecución se organiza típicamente mediante gestores de recursos y colas, que asignan recursos compartidos y ordenan trabajos en el tiempo. SLURM (*Slurm Workload Manager*) es un gestor y planificador ampliamente utilizado en clústeres Linux (Fundación Julio Ricaldoni, 2018; ClusterUY, s.f.; *Overview — Slurm Workload Manager*, s.f.).

En términos conceptuales, un planificador como SLURM cumple funciones clave: asignar acceso a recursos por un período, proveer un marco para iniciar y monitorear trabajos, y arbitrar contención mediante colas de trabajos pendientes. Estas funciones constituyen la base operativa de un entorno HPC: gestión explícita de recursos, ejecución controlada y planificación.

En este trabajo, ClusterUY/SLURM se emplea para ejecutar el procesamiento y entrenamiento a escala (jobs batch, por-bucket y/o por-experimento), controlando recursos (CPU/RAM/GPU cuando corresponde) y trazabilidad mediante logs.

A.8. PyTorch: framework de *deep learning*

PyTorch es un *framework open-source* ampliamente utilizado para desarrollar y entrenar modelos de *deep learning*. Su enfoque de diseño favorece la experimentación y el control explícito del proceso de entrenamiento, con soporte de aceleración por hardware (por ejemplo, GPU) (*PyTorch*, s.f.; Paszke y cols., 2019). En términos generales, PyTorch se utiliza para definir modelos, optimizar parámetros a partir de datos y evaluar generalización sobre conjuntos separados.

En este trabajo se adopta como base para experimentos que requieren flexibilidad fuera de *pipelines* puramente declarativos, y como ecosistema para integrar optimización de hiperparámetros e interpretabilidad.

Aquí se utiliza PyTorch para implementar y entrenar modelos neuronales (p. ej., MLP y CNN) sobre los datasets materializados por el *pipeline*.

A.9. Optuna: optimización de hiperparámetros

Optuna es un *framework* para optimización automática de hiperparámetros (HPO) en *machine learning*. Su objetivo es sistematizar la búsqueda de configuraciones (por ejemplo, tamaño del modelo, tasa de aprendizaje, regularización) ejecutando múltiples *trials* y registrando resultados para comparar alternativas, incluyendo mecanismos de detención temprana de ejecuciones con desempeño claramente inferior (*pruning*) (*Optuna: A hyperparameter optimization framework*, s.f.; Akiba, Sano, Yanase, Ohta, y Koyama, 2019; *Optuna Documentation: Pruning Unpromising Trials*, s.f.). En un flujo experimental, esto ayuda a separar la etapa de búsqueda basada en validación de la etapa de entrenamiento y reporte final, mejorando trazabilidad y reproducibilidad.

En este trabajo, Optuna se utiliza para explorar de forma sistemática configuraciones de entrenamiento/modelo y comparar alternativas bajo un protocolo controlado de validación.

A.10. Captum: interpretabilidad para modelos PyTorch

Captum es una biblioteca de interpretabilidad para modelos PyTorch que provee métodos de atribución (*feature attribution*) para estimar el aporte de cada variable de entrada en una predicción. Entre sus enfoques se incluyen técnicas basadas en gradientes y variantes que comparan contra una referencia (*baseline*) para cuantificar contribuciones relativas (*Captum: Model Interpretability for PyTorch*, s.f.; *Captum API: IntegratedGradients*, s.f.). Este tipo de herramientas se utiliza para validar coherencia con conocimiento de dominio, identificar variables dominantes o potencialmente redundantes y mejorar la documentación del comportamiento de los modelos.

En este trabajo, Captum se plantea como apoyo para interpretar modelos entrenados (especialmente tabulares), contrastando la importancia de variables con intuición de dominio y detectando dependencias espurias.

Cierre del apéndice

En síntesis, el entorno experimental se apoya en (i) una capa reproducible de ejecución mediante contenerización, (ii) almacenamiento distribuido y formatos

eficientes para persistir datasets intermedios y finales, (iii) un motor de procesamiento para materializar transformaciones a escala, y (iv) un entorno HPC para ejecutar conversiones y entrenamientos en modo batch. Sobre esta base de ingeniería de datos se montan las bibliotecas de modelado y experimentación, que habilitan construir, optimizar e interpretar modelos de ML bajo un esquema trazable.

Anexo B

Scripts del pipeline para la representación *lightpath*

B.1. Propósito y alcance

Este anexo resume los scripts y artefactos de software utilizados para materializar el pipeline *Bronze–Silver–Gold* y los experimentos base con Random Forest sobre la representación *lightpath*. El foco es proveer una referencia operativa compacta (qué produce cada paso y cómo se ejecuta), evitando documentación redundante a nivel de implementación.

B.2. Convenciones comunes de ejecución

En el repositorio del proyecto (Optical Networks ML, 2026), la estructura separa `data_pipeline/` (ETL y materializaciones) de `ml_pipeline/` (entrenamiento/evaluación). Los datasets y artefactos de salida se persisten **fuera** del repositorio, y los scripts reciben rutas por parámetro.

```
# Ejemplo de convención
export DATA_ROOT=~/datasets
export OUT_ROOT=~/outputs

# Patrón general
python <script>.py --input <...> --output_dir <...>
```

B.3. Pipeline y artefactos materializados

El pipeline para *lightpath* se organiza en los siguientes pasos:

1. NetCDF → Parquet (capa **Bronze**).

2. Limpieza/normalización (capa **Silver**).
3. Vistas **Gold** para clasificación QoT y regresión.
4. Entrenamiento y evaluación de Random Forest para cada tarea.

B.3.1. Estructura de artefactos

```
<DATA_ROOT>/
  lightpath_dataset.nc
  lightpath_dataset_bronze.parquet
  lightpath_dataset_silver.parquet/
  lightpath_dataset_gold_qot/
  lightpath_dataset_gold_reg/

<OUT_ROOT>/
  rf_qot_out/
  rf_reg_out/
```

B.4. Índice compacto de scripts

Tabla B.1: Índice compacto de scripts para *lightpath* (ETL y ML).

Script	Etapa	Entrada → Salida	Ejecución mínima
build_bronze.py	Bronze	.nc → bronze.parquet	python build_bronze.py --input ...nc --output ...parquet
build_silver.py	Silver	bronze.parquet → silver.parquet/	python build_silver.py --input ...parquet --output_dir .../
build_gold_qot.py	Gold (QoT)	silver/ → gold_qot/	python build_gold_qot.py --silver_dir ... --output_dir ...
build_gold_reg.py	Gold (Reg)	silver/ → gold_reg/	python build_gold_reg.py --silver_dir ... --output_dir ...
train_eval_rf_qot.py	RF (clf)	gold_qot/ → rf_qot_out/	python train_eval_rf_qot.py --data_dir ... --output_dir ...
train_eval_rf_reg.py	RF (reg)	gold_reg/ → rf_reg_out/	python train_eval_rf_reg.py --data_dir ... --output_dir ...

B.5. Notas metodológicas específicas

- **Gold–QoT.** Define una etiqueta binaria de QoT y materializa una vista lista para clasificación, incluyendo selección de *features* y particionado

cuando aplica.

- **Gold–Reg.** Define un objetivo continuo y materializa una vista para regresión bajo el mismo criterio de consistencia (selección de variables y partición).
- **RF.** Los experimentos guardan métricas y configuración en `rf_*_out/` para asegurar comparabilidad entre ejecuciones.

Anexo C

Scripts del pipeline para la representación *network status*

C.1. Propósito y alcance

Este anexo describe la estructura operativa del pipeline para *network status*, cuya principal diferencia respecto a *lightpath* es el costo de procesamiento y la necesidad de ejecución **batch** en clúster (Slurm), típicamente particionando el *Gold* por `bucket=*`. Las convenciones generales (paths, organización del repo y patrón de CLI) se mantienen según la Sección B.2.

C.2. Pipeline y artefactos materializados

1. Ingesta y normalización hasta **Silver** (granularidad base del dominio).
2. Construcción de **Gold** particionada por `bucket=*` para facilitar ejecución y re-ejecución selectiva.
3. Generación de insumos por **familia de representación** (tabular/secuencial/tensorial, según el modelo).
4. Entrenamiento y evaluación (RF/MLP/RNN/CNN) con ejecución por `sbatch`.

C.2.1. Estructura de artefactos (alto nivel)

```
<DATA_ROOT>/
  network_status_*.nc
  network_status_bronze.parquet
```

```
  network_status_silver.parquet/
  network_status_gold/ # tipicamente con particion bucket=*/

<OUT_ROOT>/
  rf_out/ mlp_out/ rnn_out/ cnn_out/
  logs_slurm/
```

C.3. Índice compacto de scripts

Nota. El detalle fino de parámetros se externaliza a los *jobs* de Slurm, de modo que el anexo mantenga una referencia breve y estable.

Tabla C.1: Índice compacto de scripts para *network status*.

Script	**Bloque**	**Entrada → Salida**
`build_bronze_*.py`	ETL	`.nc` → `bronze.parquet`
`build_silver_*.py`	ETL	`bronze` → `silver/`
`silver_to_gold_*_bucket.py`	Gold	`silver/` → `gold/ bucket=*/`
`train_eval_rf_*.py`	ML (tabular)	`gold/` → `rf_out/`
`train_eval_mlp_*.py`	ML (tabular)	`gold/` → `mlp_out/`
`train_eval_rnn_*.py`	ML (secuencial)	`gold/` → `rnn_out/`
`train_eval_cnn_*.py`	ML (tensorial)	`gold/` → `cnn_out/`

C.4. Ejecución en clúster con Slurm (`sbatch`)

Los *jobs* encapsulan (i) recursos, (ii) paths y (iii) parámetros del script. El patrón es ejecutar por `bucket` para minimizar recomputación, manteniendo logs por corrida.

C.4.1. Template mínimo de job

```
#!/bin/bash
#SBATCH --job-name=ns_gold_bucket
#SBATCH --cpus-per-task=16
#SBATCH --mem=64G
#SBATCH --time=02:00:00
#SBATCH --output=<OUT_ROOT>/logs_slurm/%x_%j.out
#SBATCH --error=<OUT_ROOT>/logs_slurm/%x_%j.err

python silver_to_gold_tabular_bucket.py \
  --silver_dir <DATA_ROOT>/network_status_silver.parquet \
  --output_dir <DATA_ROOT>/network_status_gold \
  --bucket_id 12
```

C.4.2. Ejecución de entrenamiento/evaluación (ejemplo)

El entrenamiento y evaluación se ejecutan mediante *jobs* de Slurm que encapsulan (i) recursos, (ii) paths de entrada/salida, y (iii) parámetros del modelo. En un flujo típico, el job define `DATA_ROOT` y `OUT_ROOT`, referencia el *Gold* (`network_status_gold/`) y persiste resultados en una carpeta de salida por familia (p. ej., `rf_out/`), dejando logs en `logs_slurm/`.

Como ejemplo concreto, el siguiente comando lanza un job que entrena y evalúa Random Forest (tabular) sobre la vista *Gold* particionada:

```
sbatch jobs/rf/job_train_eval_rf_tabular.sbatch
```

Durante la ejecución, el estado puede monitorearse con:

```
squeue -u $USER
```

Al finalizar, se espera encontrar los artefactos en:

```
<OUT_ROOT>/rf_out/
<OUT_ROOT>/logs_slurm/*.out
<OUT_ROOT>/logs_slurm/*.err
```

Anexo D

Conexión a la máquina virtual de experimentación

D.1. Propósito y alcance

Este anexo resume el procedimiento de acceso remoto a la máquina virtual (VM) utilizada para ejecutar el entorno contenerizado de experimentación (HDFS, Spark y Jupyter). El objetivo es habilitar una conexión segura y reproducible mediante túneles SSH, manteniendo el despliegue aislado y permitiendo acceder a las interfaces web desde la máquina local.

D.2. Requisitos previos

- Acceso SSH institucional (usuario en el *jump host* y credenciales habilitadas).
- Cliente `ssh` disponible en la máquina local.
- Script de conexión provisto con el repositorio (ver Sección D.3).

D.3. Acceso remoto con túnel SSH

El acceso se realiza mediante un *jump host* y *port forwarding* hacia los servicios con UI web. Para automatizar la conexión se utiliza el script `connect-sparkvm4t.sh`, versionado en el repositorio del proyecto (Optical Networks ML, 2026), que recibe como parámetro el usuario institucional:

```
chmod +x connect-sparkvm4t.sh
./connect-sparkvm4t.sh <usuario_institucional>
```

Al establecerse la sesión SSH, los servicios quedan disponibles en `localhost` (máquina local) a través de los puertos tunelizados.

D.4. Servicios expuestos y puertos

La Tabla D.1 resume los endpoints típicos accesibles desde la máquina local una vez levantado el túnel.

Tabla D.1: Puertos tunelizados para acceder a servicios del entorno en la VM.

Servicio	**URL local**	**Uso**
Spark Master UI	`http://localhost:8080`	Monitoreo del clúster Spark (master).
Spark Worker UI	`http://localhost:8081`	Monitoreo del/los workers.
HDFS NameNode UI	`http://localhost:9870`	Estado de HDFS y exploración del filesystem.
HDFS DataNode UI	`http://localhost:9864`	Estado del DataNode.
Jupyter	`http://localhost:8888`	Ejecución interactiva (notebooks).
Spark App UI	`http://localhost:4040`	UI por aplicación (puede variar si hay múltiples apps).

D.5. Proxy institucional

Dado que la VM puede no contar con salida directa a Internet, el esquema de conexión contempla habilitar el uso de proxy para tareas que requieren conectividad (por ejemplo, descarga de dependencias). En términos operativos, esto se logra encapsulándolo en la misma sesión SSH que establece el túnel.

D.6. Verificación rápida del entorno

Una vez conectado, en la VM el stack se opera con `docker compose`. Por ejemplo:

```
cd docker_spark
docker compose ps
docker compose logs -f jupyter
```