UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Aprendizaje automático aplicado al dominio de las redes ópticas

Informe de Proyecto de Grado presentado por

Santiago Olmedo Guillama

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
Eduardo Grampín

Montevideo, 25 de diciembre de 2024

Agradecimientos

Ante todo, quiero agradecer a mis padres Claudio y Carmen, a mis hermanos Juan Andrés y Victoria y a toda mi familia por el apoyo incondicional a lo largo de este proceso académico.

A Natasha, por el constante aliento y apoyo desde que arranqué el proyecto de grado, por estar siempre dispuesta a escuchar, acompañar y a brindar palabras de ánimo cuando más se necesitaron.

A mis amigos y mis compañeros de trabajo por entenderme y respaldarme en todo este proceso.

A Alberto Castro, guía fundamental para la realización de este proyecto de grado. Su apoyo, motivación y orientación en cada paso resultaron en un pilar fundamental para el éxito de este proyecto. A Eduardo Grampín por haber supervisado, guiado y ayudado en todo lo que pedí para la realización del proyecto.

A Martín Giachino por haber suministrado las herramientas necesarias para realizar el proyecto.

Este proyecto de grado es el resultado de un esfuerzo colectivo y les debo un profundo agradecimiento a todos los que de una forma u otra me ayudaron a recorrer esta etapa tan importante en mi vida. Gracias.

Resumen

En el marco de nuestro proyecto de grado, nos propusimos abordar y resolver distintos desafíos asociados a las redes ópticas, específicamente en el ámbito de las telecomunicaciones, utilizando técnicas de ciencia de datos y aprendizaje automático. Nos enfocamos en dos casos de estudio: uno en colaboración con Antel, para desarrollar un flujo de datos completo para el manejo de la información de su red, y otro utilizando un conjunto de datos proporcionado por Fraunhofer HHI para estimar la calidad de transmisión (QoT) en redes ópticas.

En la primera etapa, nuestro objetivo principal fue el desarrollo de un modelo de datos y la creación de un pipeline de datos para Antel, que nos permitiera consumir información de diversas fuentes, transformarla y almacenarla de manera eficiente. Diseñamos un sistema para tratar tanto con datos estáticos provenientes de fuentes preexistentes (como archivos CSV y bases de datos de Antel) como con datos dinámicos que planeábamos obtener a través de interfaces modernas como APIs y sistemas de transmisión de datos en tiempo real (por ejemplo, Kafka). Logramos establecer un marco inicial para el procesamiento y análisis de datos, incluyendo herramientas para la visualización que facilitaron comprender la distribución de la red óptica de Antel y su operativa.

La segunda etapa se centró en aplicar y evaluar modelos de aprendizaje automático que pudieran predecir la calidad de transmisión en redes ópticas, usando un amplio conjunto de datos proporcionado por Fraunhofer HHI. Inicialmente, realizamos un análisis exploratorio para comprender la estructura del conjunto de datos y las interrelaciones entre las diversas características. Luego, implementamos modelos variados, desde métodos de boosting hasta redes neuronales más complejas, explorando su rendimiento tanto en problemas de clasificación como de regresión.

Observamos que con una adecuada selección de características, era posible lograr modelos con alta precisión, incluso superando, en ciertos aspectos, los resultados de estudios previos realizados por Fraunhofer HHI. El trabajo también nos proporcionó oportunidades para explorar técnicas avanzadas de optimización de hiperparámetros utilizando herramientas especializadas y para explorar herramientas para estudiar la interpretación de los modelos, lo que nos permitió identificar de manera clara las características más relevantes para la predicción de la QoT.

En conclusión, este proyecto ha dado lugar a una comprensión más profunda y una sólida implementación práctica en el manejo de redes ópticas y el

aprovechamiento del aprendizaje automático para la estimación de parámetros críticos como la QoT. Sin embargo, queda espacio para mejorar, particularmente en la integración de datos en tiempo real y la aplicación de métodos estadísticos que puedan sacar provecho de todos los aspectos presentes en una red de estas características. A futuro, estos enfoques pueden extenderse para enriquecer la capacidad de predicción y respuesta adaptativa a condiciones cambiantes en tiempo real, lo que beneficiaría ampliamente a la toma de decisiones en la gestión de redes de fibra óptica.

Palabras clave: Redes ópticas, Ciencia de datos, Aprendizaje automático, Calidad de transmisión (QoT), Modelos de datos, *Pipeline* de datos, Análisis exploratorio, Modelos de clasificación, Modelos de regresión, Optimización de hiperparámetros, Interpretabilidad de modelos, Visualización de datos, Antel, Inteligencia artificial

Índice general

1. Introducción **1**
- 1.1. Motivación . . . 1
- 1.2. Objetivos . . . 2
- 1.3. Estructura . . . 2

2. Marco teórico **5**
- 2.1. Dominio de trabajo: redes ópticas . . . 5
  - 2.1.1. Evolución de las redes ópticas . . . 5
  - 2.1.2. Funcionamiento de la tecnología *DWDM* . . . 7
  - 2.1.3. *Wavelength Selective Switches* (*WSS*) y arquitectura *flex-grid* . . . 9
  - 2.1.4. Tecnología Coherente en Comunicaciones Ópticas . . . 9
  - 2.1.5. *ROADMs* . . . 11
  - 2.1.6. Arquitectura Metro/Core . . . 12
  - 2.1.7. Redes ópticas elásticas (*EONs*) . . . 12
  - 2.1.8. Optical Transport Networking (OTN) . . . 16
  - 2.1.9. Control y gestión de redes ópticas . . . 18
- 2.2. Aprendizaje automático . . . 23
  - 2.2.1. Aprendizaje supervisado . . . 23
  - 2.2.2. Aprendizaje no supervisado . . . 25
  - 2.2.3. Aprendizaje por refuerzo . . . 26
  - 2.2.4. Sobreajuste, subajuste y regularización . . . 26
- 2.3. Exploración y preparación de datos . . . 27
  - 2.3.1. Análisis exploratorio de datos (EDA) . . . 27
  - 2.3.2. Técnicas de preprocesamiento de datos . . . 29
- 2.4. Modelos de aprendizaje automático utilizados . . . 30
  - 2.4.1. Aprendizaje basado en árboles de decisión . . . 30
  - 2.4.2. Modelos de *Bagging* y *Boosting* . . . 31
  - 2.4.3. Redes neuronales . . . 34
- 2.5. Interpretación y evaluación de modelos de aprendizaje automático 36
  - 2.5.1. Interpretación de modelos . . . 36
- 2.6. Estado del arte: ciencia de datos sobre redes ópticas . . . 40

2.6.1. Artículo 1 - *Machine-learning method for quality of transmission prediction of unestablished lightpaths* (Rottondi, Barletta, Giusti, y Musumeci, 2018) . . . 41
2.6.2. Artículo 2 - *Quality of transmission estimation and short-term performance forecast of lightpaths* (Aladin, Tran, Allogba, y Tremblay, 2020) . . . 44
2.6.3. Artículo 3 - *Machine-learning-based lightpath QoT estimation and forecasting* (Allogba, Aladin, y Tremblay, 2022) 45
2.7. Otras herramientas y tecnologías utilizadas . . . 47
2.7.1. Alembic y SQLAlchemy para el almacenamiento de datos 47
2.7.2. Streamlit para la creación de interfaces de usuario . . . 48
2.7.3. Optuna para la optimización de hiperparámetros . . . 48
2.7.4. Apache Kafka para la gestión de flujos de datos . . . 48

3. Caso de estudio: Antel **51**
3.1. Dominio de Antel . . . 52
3.1.1. Componentes clave en la red OTN de Antel . . . 53
3.2. Descripción de entidades del dominio . . . 53
3.3. Características Operativas de Antel . . . 54
3.4. OTN en la red de Antel . . . 54
3.5. Materiales . . . 55
3.6. *Pipeline* de datos . . . 56
3.6.1. Consumo y transformación de datos . . . 56
3.6.2. Creación de un modelo de datos . . . 59
3.6.3. Visualización y análisis de datos . . . 61
3.7. Herramientas utilizadas . . . 67
3.8. Conclusiones . . . 67

4. Caso de estudio: HHI - Estimación de calidad de transmisión **69**
4.1. Materiales . . . 70
4.1.1. Descripción de los datos . . . 71
4.1.2. Variables objetivo . . . 75
4.1.3. Conjunto de datos basado en *lightpaths* . . . 76
4.1.4. Conjunto de datos basado en el estado de la red . . . 84
4.2. Metodologías aplicadas sobre el conjunto de datos basado en *lightpaths* . . . 86
4.2.1. Trabajo realizado por HHI . . . 87
4.2.2. Modelos de aprendizaje automático . . . 88
4.3. Resultados obtenidos sobre el conjunto de datos basado en *lightpaths* . . . 90
4.3.1. Primer modelo - HistGradientBoostingRegressor sin ajustes en los hiperparámetros . . . 90
4.3.2. Segundo modelo - HistGradientBoostingRegressor con cinco *features* seleccionadas . . . 94
4.3.3. Tercer modelo - HistGradientBoostingRegressor con tres *features* seleccionadas . . . 94

4.3.4. Cuarto modelo - HistGradientBoostingRegressor con dos *features* seleccionadas . . . 95
4.3.5. Herramientas utilizadas . . . 97
4.3.6. Comparación con el Trabajo de HHI . . . 97
4.4. Metodologías aplicadas sobre el conjunto de datos basado en el estado de la red . . . 98
4.4.1. Trabajo realizado por HHI . . . 99
4.4.2. Transformación inicial de los datos crudos . . . 101
4.4.3. Preprocesamiento para el modelo de aprendizaje automático - Primera versión . . . 103
4.4.4. Preprocesamiento para el modelo de aprendizaje automático - Segunda versión . . . 104
4.4.5. Preprocesamiento para el modelo de aprendizaje automático - Tercera versión . . . 105
4.4.6. Modelo de aprendizaje automático - MLP . . . 106
4.4.7. Optimización de hiperparámetros . . . 107
4.4.8. Ingeniería de *features* aplicada al conjunto de datos basado en el estado de la red . . . 109
4.4.9. Metodologías aplicadas sobre el conjunto de datos luego de la ingeniería de *features* . . . 111
4.5. Resultados sobre el conjunto de datos basado en el estado de la red 112
4.5.1. Evaluación sobre la primera versión de los datos . . . 112
4.5.2. Evaluación sobre la segunda versión de los datos . . . 115
4.5.3. Evaluación sobre la tercera versión de los datos . . . 115
4.5.4. Resultados del modelo de *boosting* sobre el conjunto de datos al que se le aplicó ingeniería de *features* . . . 117
4.5.5. Resultados del modelo de redes neuronales sobre el conjunto de datos al que se le aplicó ingeniería de *features* . . . 122
4.5.6. Herramientas utilizadas . . . 124
4.6. Conclusiones . . . 124

5. Conclusiones sobre el proyecto **125**
5.1. Trabajo futuro . . . 125

Referencias **127**

Capítulo 1

Introducción

1.1. Motivación

Las redes ópticas se han convertido en un pilar fundamental de la infraestructura de telecomunicaciones, debido a su capacidad para manejar el creciente volumen de datos que circula a través de las redes globales. Este aumento exponencial en la demanda de ancho de banda, impulsado por dos tendencias clave (López y Velasco, 2016), está motivando a los proveedores de servicios a continuar invirtiendo en la expansión y mejora de sus redes ópticas:

- El consumo de aplicaciones que demandan gran cantidad de ancho de banda, como el *streaming* de video y las aplicaciones en la nube, está incrementando significativamente la demanda de capacidad en las redes.
- El despliegue de fibra óptica y otros medios de acceso de alta velocidad para los consumidores finales está facilitando el acceso a estos servicios de alta demanda.

Estas tendencias no solo garantizan que los consumidores tendrán acceso a una amplia gama de servicios de alta velocidad, sino que también presionan a los proveedores de servicios para que amplíen y mejoren sus redes ópticas, asegurando así su competitividad en un mercado en rápida evolución.

Las redes ópticas destacan como una de las tecnologías de transmisión más dominantes, gracias a su eficiencia en términos de costos, capacidad y calidad de comunicación, necesarias para satisfacer la creciente demanda. En estas redes, la información se transmite como trenes de fotones a través de caminos de luz (*lightpaths*) que conectan los nodos origen y destino. Entre sus principales ventajas frente a los sistemas de comunicación eléctricos se encuentran:

- Mayor ancho de banda, con frecuencias de transporte en el rango de 10^{13} a 10^{15} Hz.
- Pérdidas de transmisión extremadamente bajas, del orden de 0.002 dB/km.

- Bajo peso y tamaño del medio de transmisión, lo que las hace ideales para conexiones de baja latencia en largas distancias (Norouzi, Zaim, y Ustundag, 2011; Ramaswami, Sivarajan, y Sasaki, 2009).

Sin embargo, la gestión de redes ópticas presenta desafíos significativos. En términos de configuración, es necesario considerar una amplia variedad de factores que afectan la calidad y velocidad del servicio (Patri, Autenrieth, Rafique, Elbers, y Machuca, 2020). Además, el despliegue de estas redes requiere resolver problemas de compatibilidad entre dispositivos de diferentes proveedores y tomar decisiones estratégicas relacionadas con la adquisición de infraestructura para soportar el crecimiento de la red (Ramaswami y cols., 2009). La operación diaria implica tareas como monitorizar la calidad de transmisión, detectar fallos y asignar recursos, entre otras, lo que subraya la complejidad inherente de gestionar estas redes avanzadas.

1.2. Objetivos

El proyecto tiene por objetivo construir un flujo entero de trabajo para la aplicación de métodos de aprendizaje automático en el dominio de las redes ópticas. Para ello, se propone abordar los siguientes objetivos específicos:

- Estudiar los fundamentos teóricos y técnicos de las redes ópticas, incluyendo su implementación y funcionamiento.
- Realizar una revisión bibliográfica de los métodos de aprendizaje automático aplicados a redes ópticas, identificando las técnicas más utilizadas y sus aplicaciones.
- Desarrollo e investigación de herramientas para obtener, procesar, almacenar y analizar datos de redes ópticas a partir de distintas fuentes.
- Desarrollo de modelos de aprendizaje automático para la predicción de la calidad de transmisión en redes ópticas.

1.3. Estructura

El contenido de este informe se organiza de la siguiente manera. En el Capítulo 2 se presenta el marco teórico, que incluye una descripción del dominio de trabajo (redes ópticas) y los conceptos fundamentales relacionados con las tecnologías inherentes a este campo. A su vez, se abordan los conceptos del área de aprendizaje automático y ciencia de datos y su aplicación en el dominio de las redes ópticas. Sobre el final del capítulo, se hace una revisión de la literatura existente en la aplicación de métodos de aprendizaje automático en redes ópticas.

En el Capítulo 3 se describe la metodologías propuestas para obtener, procesar y analizar los datos de la red óptica de Antel. Se detallan las herramientas

utilizadas para la creación de un flujo de trabajo que tiene como salida la construcción de un conjunto de datos que será utilizado para entrenar y evaluar modelos de aprendizaje automático.

En el Capítulo 4 se presenta un estudio exhaustivo de técnicas de análisis exploratorio de datos, preprocesamiento y selección de características, implementación de modelos de aprendizaje automático y evaluación de los mismos para estimar la calidad de transmisión en redes ópticas a partir de un conjunto de datos provisto por Fraunhofer Heinrich Hertz Institute. En este capítulo se detallan los resultados obtenidos y se discuten los hallazgos más relevantes sobre un conjunto de datos establecido.

En el Capítulo 5 se presentan las conclusiones del trabajo realizado, se discuten los resultados obtenidos y se proponen posibles líneas de trabajo futuro.

Capítulo 2

Marco teórico

2.1. Dominio de trabajo: redes ópticas

La tecnología de fibra óptica es ampliamente utilizada en las redes de telecomunicación de hoy en día, ya que permiten la transmisión sobre distancias más largas y a mayores anchos de banda que los cables de cobre. Estas propiedades se deben a las menores pérdidas y al mayor número de canales que pueden ser transportados simultáneamente sobre su amplio espectro.

2.1.1. Evolución de las redes ópticas

En el mundo de las redes ópticas, los datos se transmiten sobre fibras ópticas utilizando luz. La luz utilizada para la comunicación se encuentra dentro de un rango específico de longitudes de onda (o colores) en el espectro. Las fibras ópticas están diseñadas para funcionar mejor en ciertas partes de este espectro, particularmente en el rango de longitudes de onda entre 1.3 y 1.6 micrómetros (μm).

Una parte específica de este rango, conocida como la banda C, es particularmente importante. Esta banda corresponde a longitudes de onda alrededor de 1550 nanómetros (nm), que es aproximadamente lo mismo que 193.10 terahertz (THz) en términos de frecuencia. La banda C es favorecida porque tiene la menor pérdida de señal, lo que significa que los datos pueden ser transmitidos sobre distancias más largas sin necesidad de amplificar la señal tan a menudo.

En las redes ópticas, una técnica llamada Multiplexación por División de Longitud de Onda Densa (*DWDM*) se utiliza para enviar múltiples señales de datos sobre una sola fibra óptica simultáneamente. Esto es similar a cómo múltiples estaciones de radio transmiten sobre diferentes frecuencias en el mismo espectro de radio. En *DWDM*, a cada señal de datos se le asigna una longitud de onda diferente dentro de la banda C.

Inicialmente, estas longitudes de onda estaban separadas por 100 GHz (aproximadamente 0.8 nm en longitud de onda). Este espaciado era estándar porque

permitía que cada señal se transmitiera sin interferencia de las demás. Sin embargo, a medida que el tráfico de datos aumentaba, la industria necesitaba transmitir más señales (canales) sobre la misma fibra.

A medida que pasó el tiempo, la tecnología mejoró, permitiendo que cada canal llevara más datos. Esto se logró aumentando la tasa de datos de cada señal, pasando de 2.5 Gb/s a 100 Gb/s por longitud de onda. El resultado es una mayor eficiencia espectral: se puede enviar más datos sobre la misma cantidad de espectro. Cuando se pasó de 100 Gb/s por longitud de onda, se comenzaron a utilizar formatos de modulación más complejos como 16-QAM (Modulación de Amplitud en Cuadratura), que pueden duplicar la tasa de datos en comparación con métodos más simples como QPSK (Modulación de Desplazamiento de Fase en Cuadratura). Sin embargo, el inconveniente es que estas tasas de datos más altas generalmente solo funcionan sobre distancias más cortas porque la calidad de la señal se degrada más rápido.

Para gestionar estos desafíos, la industria se ha movido hacia un enfoque más flexible llamado tecnología *flex-grid*. *Flex-grid* permite ajustar dinámicamente el espaciado entre canales en función de las necesidades de los datos que se transmiten. En lugar de estar limitado a una grilla fija de 50 GHz, los canales ahora pueden asignar cantidades variables de espectro, como 37.5 GHz para un canal de 100 Gb/s o 75 GHz para un canal de 400 Gb/s. En la Figura 2.1 se muestra un ejemplo de cómo se puede asignar el espectro en una red *flex-grid*. En la figura se ve que ahora se puede tener 8 canales de 200 Gb/s en lugar de 6 pero más importante estos se pueden agrupar para generar súper canales que son tratados como una sola entidad.

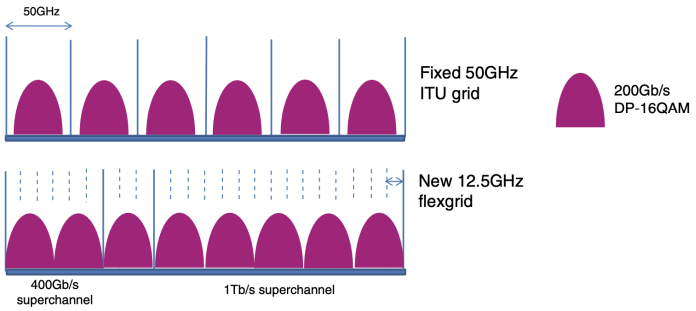

Figura 2.1: Ejemplo de asignación de espectro en una red *flex-grid*. Fuente: (López y Velasco, 2016)

Usando técnicas avanzadas como *DWDM*, formatos de modulación más altos y tecnología de *flex-grid*, la industria ha aumentado drásticamente la capacidad de las fibras ópticas, permitiéndoles transportar mucha más información de la que era posible en el pasado.

2.1.2. Funcionamiento de la tecnología *DWDM*

La Multiplexación por División de Longitud de Onda Densa (*DWDM*) es una tecnología utilizada en redes ópticas para aumentar significativamente la cantidad de datos que se pueden transmitir sobre una sola fibra óptica. Lo hace utilizando múltiples longitudes de onda de luz (o colores) para transportar diferentes flujos de datos simultáneamente a través de la misma fibra. En la Figura 2.2 se muestra un esquema de un sistema *DWDM* punto a punto. En este sistema, los datos se envían de un punto a otro de la siguiente manera:

- Transmisores ópticos: Estos dispositivos generan múltiples señales de luz, cada una a una longitud de onda diferente. Cada longitud de onda transporta una señal de datos diferente. Por ejemplo, una longitud de onda podría transportar datos de video, otra podría transportar texto, y así sucesivamente.

- Multiplexor (*MUX*): Todas estas longitudes de onda diferentes se combinan en una sola señal óptica utilizando un dispositivo llamado multiplexor. Esta señal combinada ahora puede viajar por la fibra como una sola entidad, aunque contiene múltiples canales de datos.

- Enlace de fibra óptica: La señal *DWDM* combinada viaja a través de la fibra óptica. En el camino, la señal podría debilitarse debido a la distancia. Para evitar la pérdida de datos, se utilizan amplificadores ópticos para aumentar la potencia de señal. Estos amplificadores aseguran que los datos puedan viajar largas distancias sin degradación.

- Compensación de la distorsión de la señal: A medida que la luz viaja a través de la fibra, sus señales podrían dispersarse debido a un fenómeno llamado dispersión cromática. Tradicionalmente, se usaría un Módulo de Compensación de Dispersión (*DCM*) para contrarrestar este efecto. Sin embargo, los sistemas modernos que utilizan *coherent technlogy* pueden manejar esta compensación en los receptores, por lo que estos sistemas ya no necesitan *DCM* (López y Velasco, 2016).

- De-multiplexor (*DEMUX*): En el extremo receptor, la señal combinada se divide nuevamente en sus longitudes de onda individuales utilizando un de-multiplexor. Cada flujo de datos se envía entonces al receptor apropiado para ser decodificado y procesado.

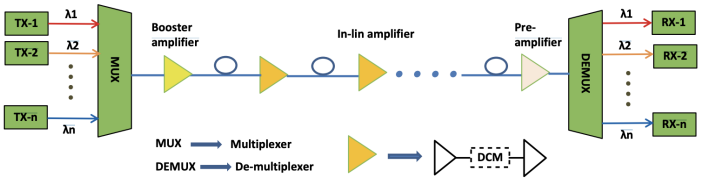

Figura 2.2: Estructura de un sistema *DWDM* punto a punto. Fuente: (López y Velasco, 2016)

La tecnología *DWDM* es una forma muy eficiente de aumentar la capacidad de una red. Al transmitir múltiples señales sobre una sola fibra, reduce el costo total porque la infraestructura costosa (como cables de fibra y amplificadores) se comparte entre todos los canales de datos. A medida que la demanda de datos crece, se pueden agregar más canales simplemente instalando transmisores y receptores adicionales (transpondedores) en los extremos del tráfico (López y Velasco, 2016).

Cuando la tecnología *DWDM* apareció por primera vez a mediados de la década de 1990, la tasa de datos típica por canal era de 2.5 Gb/s. A medida que la tecnología avanzaba, las tasas de datos aumentaron a 10 Gb/s, luego a 40 Gb/s y más recientemente a 100 Gb/s, gracias a las mejoras en las técnicas de modulación y la *coherent technlogy*. Estos avances aumentaron drásticamente la capacidad total de una fibra, lo que significa que se puede enviar más datos sobre la misma fibra, al mismo tiempo que se mejora la eficiencia espectral, que es una medida de cuán eficientemente se utiliza el espectro.

Al principio, los sistemas *DWDM* eran solo conexiones punto a punto, pero evolucionaron. En las redes modernas, es necesario enrutar señales dinámicamente entre diferentes ubicaciones, no solo enviarlas de un punto A a un punto B. Acá es donde entran en juego los *Optical Add-Drop Multiplexers* (*OADMs*) y los más avanzados *Reconfigurable Optical Add-Drop Multiplexers* (*ROADMs*). Los *OADM* permiten agregar o eliminar ciertas longitudes de onda (canales) en puntos intermedios de la red sin perturbar otros canales. Los *ROADM* son una versión más avanzada que puede ser controlada remotamente, permitiendo a los operadores de red reconfigurar la red según sea necesario, agregando, eliminando o redirigiendo canales sin necesidad de ajustar físicamente el equipo. Las redes basadas en *ROADMs* (2.1.5) proporcionan una flexibilidad significativa, especialmente en el manejo de demandas de datos crecientes y dinámicas. Permiten ajustes en la red, ya sea agregando nuevos canales, redirigiendo el tráfico en caso de fallas o equilibrando la potencia entre canales para mantener la calidad de la señal. Todas estas operaciones pueden ser gestionadas de forma remota a través de software, reduciendo los costos operativos y haciendo que la red sea más resistente y adaptable.

2.1.3. *Wavelength Selective Switches* (*WSS*) y arquitectura *flex-grid*

Un componente crucial en las redes ópticas, particularmente en sistemas que utilizan *DWDM*, es el *Wavelength Selective Switches*. Su función principal es tomar una mezcla de señales de luz (cada una a diferentes longitudes de onda) de una sola fibra óptica y dirigir estas señales individuales a varias fibras de salida según sea necesario. Esto se hace sin restricciones, lo que significa que cualquier longitud de onda puede ser enviada a cualquier puerto de salida. Esto se puede ver en la Figura 2.3.

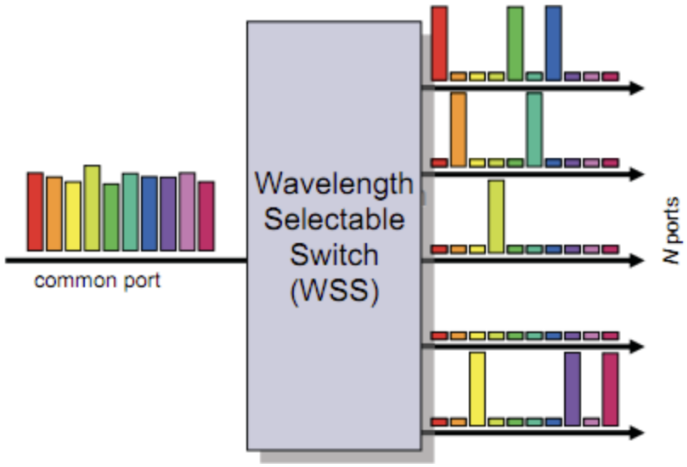

Figura 2.3: Funcionamiento de un *WSS*. Fuente: (FiberOptics4Sale Blog, 2024)

En sistemas tradicionales *DWDM*, las longitudes de onda están espaciadas a intervalos fijos, como 50 GHz o 100 GHz, establecidos durante la fabricación del equipo. A medida que crecen las demandas de datos, las redes necesitan más flexibilidad. Acá es donde entra en juego la tecnología *flex-grid*. A diferencia de los sistemas de grilla fija, *flex-grid* permite que la red ajuste dinámicamente el espaciado entre longitudes de onda. Esta flexibilidad es crucial para optimizar el uso del espectro disponible, especialmente a medida que transmitimos más datos a velocidades más altas. *WSS* habilita este enfoque de *flex-grid* haciendo más que simplemente enrutar longitudes de onda. También debe permitir anchos de banda variables, un control más fino sobre cómo se utilizan estas frecuencias y una rápida reconfiguración para adaptarse a las demandas cambiantes de la red.

2.1.4. Tecnología Coherente en Comunicaciones Ópticas

La tecnología coherente trajo cambios en las comunicaciones ópticas al asociar la detección coherente con el procesamiento digital de señales (DSP), pro-

porcionando soluciones innovadoras a las limitaciones de los sistemas tradicionales. En sus inicios, las comunicaciones ópticas dependían de la modulación de intensidad en formatos básicos como *On-Off Keying* (OOK), donde la información binaria se representaba mediante niveles altos y bajos de intensidad óptica. Sin embargo, con el aumento de las demandas de ancho de banda, técnicas como la Multiplexación por División en el Tiempo (TDM) y la Multiplexación por División de Longitud de Onda (WDM) llevaron las velocidades a 40 Gb/s y 100 Gb/s. Este progreso expuso desafíos significativos, como la dispersión cromática, el ruido y los efectos no lineales, que restringieron las distancias de transmisión (Charlet, 2008).

La detección coherente se basa en el uso de un oscilador local que actúa como una señal de referencia pura, permitiendo la recuperación de la amplitud y la fase del campo eléctrico óptico. Esto la diferencia de la detección directa, donde solo se captura la intensidad óptica. Una de las principales ventajas de la detección coherente es su capacidad para mejorar significativamente la sensibilidad al ruido, acercándose al límite teórico previsto por la teoría de comunicaciones digitales. Además, ofrece una solución digital para compensar efectos lineales como la dispersión cromática y la dispersión por modo de polarización (PMD), sin la necesidad de módulos de compensación óptica.

El DSP juega un papel esencial en los receptores coherentes. Inicialmente, realiza la compensación de la dispersión cromática utilizando filtros de respuesta finita (FIR) o transformadas rápidas de Fourier (FFT). Posteriormente, aplica algoritmos como el *Constant Modulus Algorithm* (CMA) para ecualizar las señales y separar las polarizaciones multiplexadas, contrarrestando distorsiones introducidas durante la propagación. Este procesamiento digital permite manejar incluso altas cantidades de PMD, como se ha demostrado en sistemas que toleran valores que habrían sido inviables con tecnologías tradicionales (Charlet, 2008).

La combinación de detección coherente con formatos de modulación avanzados, como Quadrature Phase Shift Keying (QPSK) y Polarization Division Multiplexing QPSK (PDM-QPSK), ha permitido alcanzar densidades espectrales de información de hasta 100 Gb/s en una cuadrícula de canales de 50 GHz, lo que representa un hito en términos de eficiencia espectral. No obstante, esta tecnología enfrenta desafíos asociados a los efectos no lineales (Charlet, 2008), como la modulación de fase cruzada (XPM), especialmente cuando coexisten canales de 10 Gb/s modulados en intensidad junto con señales coherentes.

El uso de formatos de modulación multivariable, como QPSK, no solo optimiza la ocupación espectral, sino que también mejora la tolerancia al ruido. A diferencia de la modulación OOK, que utiliza la intensidad óptica, QPSK codifica información en la fase de la señal, lo que duplica la duración de los símbolos para un mismo ancho de banda. Este enfoque reduce la sensibilidad a la dispersión y mejora la calidad de la señal transmitida. Además, la multiplexación por polarización, implementada junto con la detección coherente, ha demostrado ser una técnica eficaz para duplicar la capacidad de transmisión en sistemas de alta velocidad.

2.1.5. *ROADMs*

Un *Reconfigurable Optical Add-Drop Multiplexer* es un componente muy importante en las redes ópticas modernas. Su función principal es gestionar las señales de datos transmitidas sobre fibras ópticas permitiendo que longitudes de onda específicas (o canales) sean agregadas, eliminadas o pasadas a través de un nodo de red de forma dinámica. Cuando los datos fluyen a través de un nodo de red equipado con un *ROADM*, pueden seguir uno de los dos caminos principales (este funcionamiento se puede apreciar en la Figura 2.4):

- *Express Path*: Datos que pasan por el nodo sin detenerse. Esta es la ruta más rápida para los datos que no necesitan ser procesados o redirigidos en ese nodo.
- *Add/Drop Path*: Datos que comienzan o terminan en este nodo, o que necesitan ser redirigidos. Este se encarga de agregar nuevas conexiones a la red óptica y de quitar las que deben salir en ese nodo y "subir" a la capa de enlace.

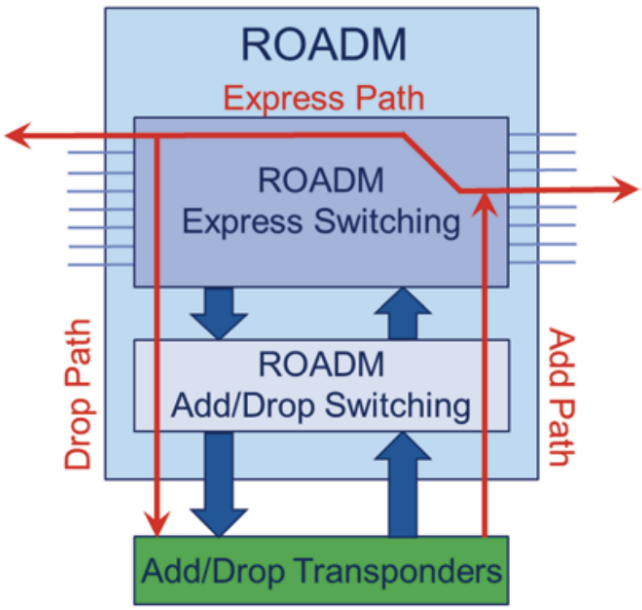

Figura 2.4: Funcionamiento de un *Reconfigurable Optical Add-Drop Multiplexer (ROADM)*. Fuente: (López y Velasco, 2016)

Como se explicó anteriormente, a medida que aumentaron las demandas de la red, se necesitó un enfoque más adaptable que los *ROADMs* iniciales que

operaban en una grilla fija, donde las longitudes de onda estaban espaciadas a intervalos fijos (como 50 GHz o 100 GHz) lo que llevó al desarrollo de los *ROADMs* de *flex-grid*. Los *ROADMs* basados *flex-grid* tienen la ventaja de poder manejar tanto anchos de banda de canal fijos como variables, lo que significa que pueden ajustarse dinámicamente a diferentes tasas de datos y espaciados de canal. Esto es crucial para optimizar el uso del espectro disponible, especialmente a medida que las redes se mueven hacia tasas de datos más altas y configuraciones más complejas. Algunos componentes importantes de un *ROADM* son los *Wavelength Selective Switches* que permiten que longitudes de onda específicas sean enrutadas a diferentes salidas o eliminadas según sea necesario. Los dispositivos *WSS* están disponibles en versiones de grilla fija y *flex-grid* y los *Optical Amplifiers* (*OA*) que amplifican la señal de luz para compensar la pérdida a medida que viajan a través de largas distancias de fibra.

2.1.6. Arquitectura Metro/Core

Las redes desplegadas suelen segmentarse en una arquitectura de tres grandes componentes: la red óptica troncal que es la que vamos a enfocarnos en este trabajo, la red metro o de agregación que consiste en muchas redes IP conectadas al core y la red de acceso que conecta a los usuarios finales con la redes metro. Este tipo de arquitectura se puede ver en la figura 2.5.

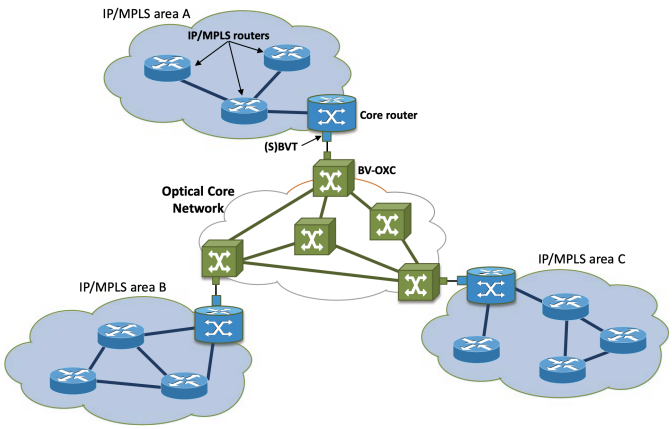

Figura 2.5: Arquitectura metro/core de una red óptica. Fuente: (López y Velasco, 2016)

2.1.7. Redes ópticas elásticas (*EONs*)

Las *Elastic Optical Networks* (*EONs*) son una evolución de las redes ópticas tradicionales que ofrecen una mayor flexibilidad y eficiencia en el uso de los recursos. A diferencia de las redes ópticas tradicionales, que dependen de una grilla de canal fija, las *EONs* introducen un enfoque más flexible, permitiendo que la

red ajuste dinámicamente el uso de sus recursos en función de las demandas actuales. En lugar de estar atadas a una grilla fija de canales, estas permiten que el ancho de banda de cada canal se ajuste para coincidir con los datos que transporta, y que la frecuencia central de cada canal se desplace según sea necesario. Esta flexibilidad se ilustra en la Figura 2.6. Aunque las *EONs* ofrecen mucha más flexibilidad, no son completamente "sin grilla". Todavía hay un "ancho de banda espectral" mínimo o una unidad base que determina cuán finamente se puede dividir el espectro. Este ancho de banda espectral es generalmente una fracción del espaciado tradicional de 100 GHz, como 6.25 GHz o 12.5 GHz. El estándar ITU-T G.694.1 define términos clave para sistemas *flex-grid* ((ITU-T, 2020)):

- *Frequency Grid*: Un conjunto de frecuencias de referencia que define dónde se pueden colocar los canales. En las *EONs*, esta grilla es mucho más fina que en los sistemas tradicionales, lo que permite un control más preciso.
- *Frequency Slot*: El rango de frecuencias asignado a un canal en particular. Este slot está definido por su frecuencia central y ancho.
- *Slot Width*: El ancho total de un slot de frecuencia. En un sistema *flex-grid*, los anchos de slot suelen ser múltiplos de 12.5 GHz.

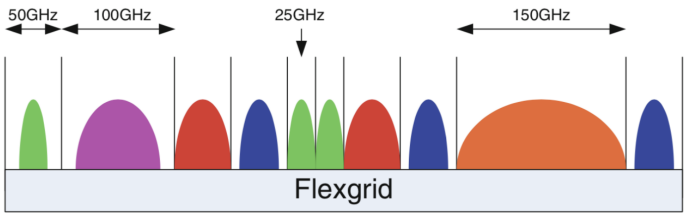

Figura 2.6: Ejemplo de flexibilidad en la asignación de canales en una red óptica elástica. Fuente: (López y Velasco, 2016)

Vemos como las *EONs* nos benefician en términos de eficiencia de recursos con un ejemplo que se puede ver en la Figura 2.7. Hay una red con varios routers (R1, R2, R3) que envían datos a otro router (R4). Estos routers están conectados directamente por enlaces ópticos, lo que significa que pueden enviar datos a R4 utilizando algo llamado *lightpath*. Un *lightpath* es esencialmente un canal dedicado dentro de una fibra óptica que lleva datos de un punto a otro.

En el escenario A, los routers R1 y R2 necesitan enviar datos utilizando un *lightpath* cada uno. Sin embargo, el router R3 tiene más tráfico para manejar y requiere tres *lightpaths* para enviar todos sus datos a R4. Por lo tanto, R4 necesita estar listo para manejar un total de cinco *lightpaths*. En el escenario B, el patrón de tráfico cambia. Ahora, R1 y R2 necesitan dos *lightpaths* cada uno, mientras que R3 solo necesita un *lightpath*. Esto lleva el número total de

lightpaths que R4 necesita manejar nuevamente a cinco, pero la distribución entre los routers cambió.

En una configuración de red estática, los recursos se asignan en función del peor escenario posible. Esto significa que R4 necesitaría estar equipado con suficientes *transponders* (dispositivos que gestionan *lightpaths*) para manejar la carga de tráfico máxima posible de cada router en cualquier momento. En este caso R4 necesitaría 7 *transponders*: 1 para R1, 1 para R2 y 3 para R3 del escenario A, más 2 adicionales para R1 y R2 del escenario B. Aunque el escenario B reduce la carga en R3, la configuración estática debe prepararse para la carga máxima en todos los escenarios, lo que lleva a una ineficiencia de recursos.

En contraste, una configuración de red dinámica se adapta a las condiciones de tráfico cambiantes en tiempo real. En lugar de configurar todos los *lightpaths* posibles por adelantado, la red crea y elimina *lightpaths* según sea necesario en función de la demanda de tráfico actual. En esta configuración dinámica, dado que el número total de *lightpaths* requeridos en cualquier momento es de 5 (independientemente de qué router los necesite), R4 solo necesita 5 *transponders*. Esta flexibilidad permite que R4 maneje los patrones de tráfico cambiantes de manera eficiente sin sobregestionar los recursos.

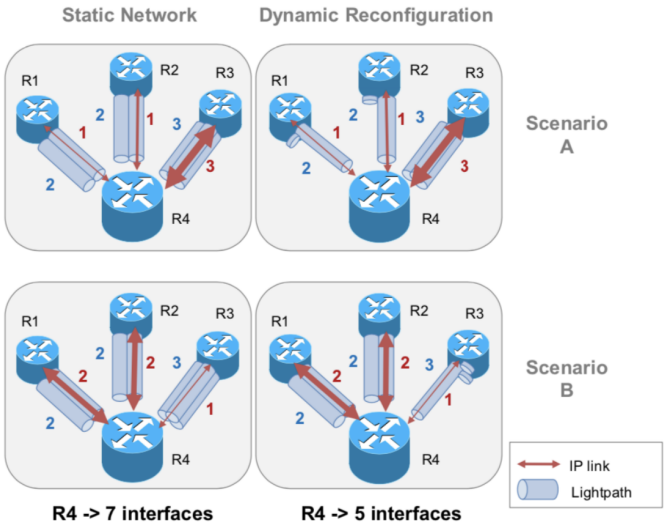

Figura 2.7: Ejemplo de eficiencia de recursos en una red óptica elástica. Fuente: (López y Velasco, 2016)

Otros conceptos importantes en las *EONs*:

- *Baud Rate*: Se refiere a la velocidad a la que se generan los símbolos en la fuente de una transmisión. Cada símbolo puede llevar múltiples bits de información, dependiendo del formato de modulación utilizado.

- *Modulation Format*: El formato de modulación determina cuántos bits puede llevar cada símbolo. Los formatos de modulación comunes incluyen QPSK (Modulación de Desplazamiento de Fase en Cuadratura), 16-QAM (Modulación de Amplitud en Cuadratura) y 64-QAM.

- *Line Rate*: Es la tasa de información real utilizada para transportar datos entre dos nodos en la red, ya sean adyacentes o distantes.

La elección de estos parámetros genera los compromisos entre conectividad, capacidad y alcance óptico. Por ejemplo, en un escenario donde un nodo (digamos nodo A) necesita conectarse a varios otros nodos (B, C, D, etc.), los *carriers* (es decir, las señales ópticas que transportan información en distintas longitudes de onda o frecuencias dentro del espectro disponible) pueden distribuirse entre estas conexiones. Sin embargo, cada conexión requiere *guard-bands* que son bandas de frecuencia sin uso entre *carriers* adyacentes para evitar interferencias. Esta configuración ofrece alta conectividad pero puede reducir la *line rate* general para cada conexión. Alternativamente, todos los *carriers* se pueden combinar en un súper canal para conectar el nodo A a un solo otro nodo (por ejemplo, el nodo C), maximizando la *line rate* (por ejemplo, 1 Tb/s) pero reduciendo la conectividad ya que el súper canal está dedicado a una sola conexión. La elección del formato de modulación también afecta el alcance óptico: la distancia que una señal puede viajar manteniendo una calidad suficiente para ser reconocida por el receptor. Los formatos de modulación más complejos como 64-QAM aumentan la *line rate* pero reducen el alcance óptico, haciendo que el sistema sea más adecuado para distancias cortas. Por otro lado, los formatos de modulación más simples como QPSK pueden extender el alcance óptico pero requieren más ancho de banda y potencialmente más transpondedores. Este ejemplo se puede ver en la Figura 2.8.

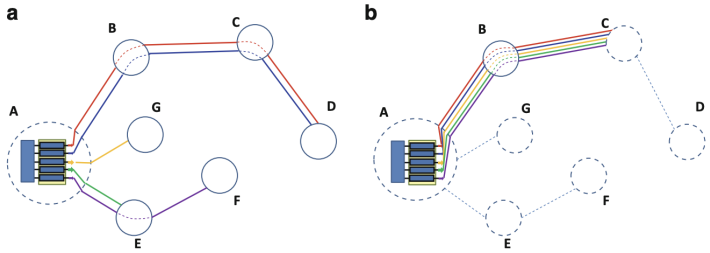

Figura 2.8: Compromisos entre conectividad, capacidad y alcance óptico en una red óptica elástica. Fuente: (López y Velasco, 2016)

El advenimiento de las *EONs* trajo consigo otros avances tecnológicos como:

- *Direct-Detection (DD) Systems*: Los sistemas tempranos utilizaban detección directa de intensidad modulada, donde la intensidad de la señal era la principal forma de codificar datos.

- *Quadrature Amplitude Modulation (QAM)*: Los sistemas modernos utilizan formatos de modulación más sofisticados como QAM, que codifican datos tanto en la amplitud como en la fase de la señal de luz, permitiendo transmitir más información en el mismo ancho de banda.

- *Coherent Technology*: A diferencia de la detección directa, la *coherent technology* preserva tanto la amplitud como la fase de la señal, lo que permite el uso de técnicas avanzadas de Procesamiento Digital de Señales (DSP) para corregir distorsiones de la señal y optimizar la transmisión.

El rol de las técnicas de Procesamiento Digital de Señales (DSP) es fundamental en los sistemas ópticos modernos y permite: compensar las distorsiones de la señal que ocurren durante la transmisión, mejorar los formatos de modulación que permite utilizar formatos más complejos como 16-QAM o 64-QAM y utilizar electrónica de alta velocidad que permite procesar datos a velocidades de hasta 400 Gb/s. A pesar de estos avances, no todos los impedimentos de transmisión pueden ser completamente compensados por DSP: los *nonlinear effects* como la Modulación de Fase Autónoma (SPM) y la Modulación de Fase Cruzada (XPM) pueden distorsionar la señal y son más difíciles de mitigar, lo que sigue siendo un obstáculo significativo para aumentar aún más la capacidad del sistema.

Otros avances en las *EONs* (López y Velasco, 2016) incluyen el uso de *Bandwidth-Variable transponders (BVTs)* y *Sliceable-BVTs (S-BVTs)*. Los *BVTs* pueden ajustar dinámicamente su ancho de banda variando el número de *subcarriers*, tasa de símbolos y formatos de modulación. Esta adaptabilidad es clave para optimizar tanto el alcance como la capacidad. Los *S-BVTs* son aún más avanzados, permitiendo la transmisión *multiflow*, donde los datos se envían desde un punto a múltiples destinos con tasas de tráfico ajustables.

2.1.8. Optical Transport Networking (OTN)

Optical Transport Networking (OTN)(ITU-T, 2016) es una arquitectura de red definida por la ITU(Union, s.f.), en varias recomendaciones, como G.709(ITU-T, 2016) y G.798(ITU-T, 2015), y proporciona una forma eficiente de transportar, conmutar y multiplexar diferentes servicios en longitudes de onda de alta capacidad a través de la red óptica. Hoy en día los proveedores de red utilizan la tecnología OTN en sus redes ópticas para obtener beneficios que incluyen mayor resiliencia, operaciones simplificadas, acuerdos de nivel de servicio mejorados, alcance extendido con corrección de errores hacia adelante (FEC) y la capacidad de maximizar eficientemente la ocupación de longitudes de onda, así como la calidad de servicios de extremo a extremo garantizada.

La arquitectura OTN es un *wrapper* digital que envuelve cada cliente/servicio de forma transparente en un contenedor para transportarlo a través de las redes ópticas, preservando la estructura nativa del cliente, la información de temporización y la información de gestión. La capacidad de multiplexación mejorada de OTN permite que se transporten diferentes tipos de tráfico, incluidos IP,

Ethernet, almacenamiento, video digital y SONET/SDH, sobre una estructura de enmarcado OTN.

Desde su creación en 2001, OTN ha evolucionado y hoy en día soporta Ethernet desde 1GE hasta 400GE.

En la Figura 2.9 se muestra un diagrama de la estructura de información OTN. A su vez, la Figura 2.10 describe como diferentes servicios que utilizan la red óptica (Ethernet, Storage Area Network (SAN), Synchronous optical networking (SONET)) se traducen a Optical Data Units (ODU), son agregados en una o varias Optical Transport Unit (OTU) y esta se termina enviado por el canal óptico.

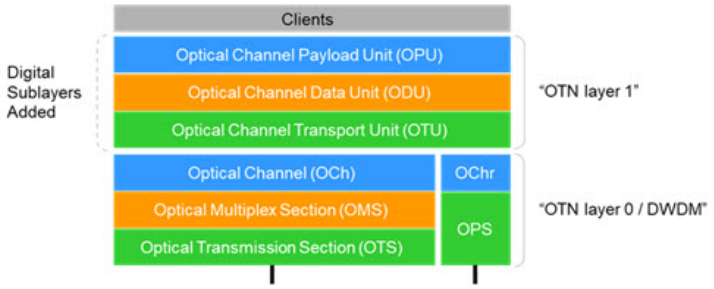

Figura 2.9: Diagrama de la estructura de información OTN. Fuente: (Corporation, 2024)

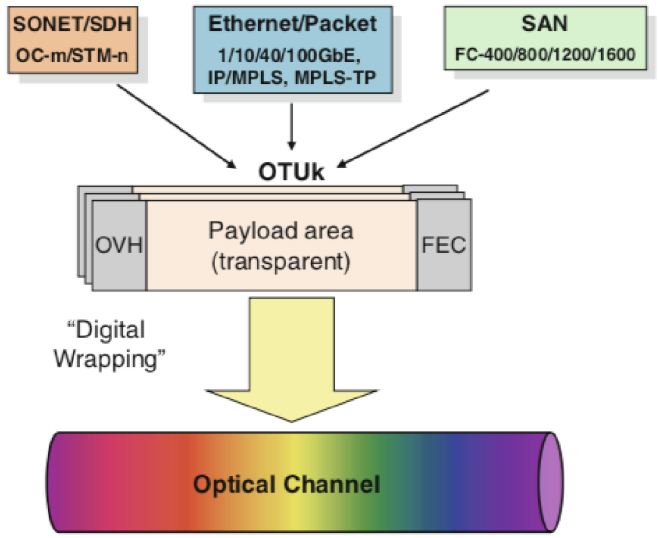

Figura 2.10: Arquitectura de Optical Transport Networking (OTN). Fuente: (Bianchi y Donnangelo, 2022)

Las OTU tienen un equivalente en Gb/s, estos se muestran en la Tabla 2.1. Si un servicio requiere demasiado ancho de banda, por ejemplo, si se quiere transportar 100GbE en una OTU1, entonces se podrá distribuir en varias OTU. En las siguientes secciones se describe la implementación de OTN en la red óptica de Antel.

OTU tipo	**OTUk tasa de bits**	**OTUk tasa de *payload***	**Clientes típicos**
OTU1	3 Gb/s	2 Gb/s	STM-16, GbE, FC-100/200
OTU2	11 Gb/s	10 Gb/s	STM-64, 10GbE WAN, 10GbE LAN FC-400/800
OTU3	43 Gb/s	40 Gb/s	40GbE LAN, STM256
OTU4	112 Gb/s	104 Gb/s	100GbE LAN

Tabla 2.1: Descripción de cada OTU, con equivalente en Gb/s y ejemplos de uso

Dentro de la arquitectura OTN, los *lightpaths* son implementados como Optical Transport Units (OTU), que encapsulan los datos de los clientes en Optical Data Units (ODU) y los envían a través de la infraestructura óptica. Este proceso garantiza que la información sea transportada de forma eficiente y transparente. Por lo tanto, un canal óptico (*optical channel*) puede verse como la representación física de un *lightpath*, mientras que OTN proporciona la estructura lógica y el protocolo necesario para su operación eficiente.

2.1.9. Control y gestión de redes ópticas

En las redes ópticas, el plano de control es esencialmente el software utilizado para controlar todos los dispositivos de la red que gestiona la operación de la red. Es una capa de software que interactúa con los componentes de hardware (como switches, moduladores y transpondedores) para controlar cómo fluyen los datos a través de la red. Las funciones clave del plano de control incluyen:

- *Provisioning* (Configuración y eliminación de conexiones): El plano de control configura automáticamente todos los dispositivos necesarios en la red para establecer una conexión entre dos o más puntos. Este proceso, conocido como señalización, garantiza que los datos puedan fluir de un punto a otro sin intervención manual.

- Restauración: Si hay una falla en la red (como un corte de fibra), el plano de control puede redirigir automáticamente los datos para mantener la calidad del servicio. Este proceso puede implicar cambiar la ruta física que los datos toman a través de la red.

- Descubrimiento automático de elementos de red: El plano de control monitorea constantemente la red para descubrir qué elementos (como switches y routers) están presentes y operativos. Actualiza automáticamente su conocimiento de la topología de la red (el diseño de cómo están conectados todos los dispositivos).

- Enrutamiento: Basado en la topología de red descubierta, el plano de control construye un mapa de la red que incluye nodos (dispositivos como routers o switches) y aristas (las conexiones entre ellos). Utiliza este mapa para determinar las mejores rutas para que los datos viajen a través de la red. Ingeniería de tráfico (TE) es parte de este proceso, donde el plano de control optimiza el uso de los recursos de la red (como el ancho de banda) considerando restricciones como el espectro disponible y los posibles puntos de falla.

- *Path Computation*: El plano de control calcula las rutas más eficientes para que los datos viajen por la red, teniendo en cuenta factores como el costo, la disponibilidad de ancho de banda y los riesgos potenciales (como enlaces compartidos que podrían fallar simultáneamente).

GMPLS es una de las arquitecturas de plano de control más utilizadas en redes ópticas (Mannie, 2004) (Figura 2.11). *GMPLS* es una extensión de lo que se utiliza en redes IP tradicionales (*MPLS*) (Rosen, Viswanathan, y Callon, 2001), diseñado para manejar no solo rutas conmutadas por paquetes, sino también rutas en redes ópticas y de transporte. Los componentes de *GMPLS* incluyen:

- *Signaling (RSVP-TE)*: En *GMPLS*, el protocolo de señalización utilizado es el Protocolo de Reserva de Recursos-Ingeniería de Tráfico (*RSVP-TE*). Este protocolo es responsable de establecer y desmontar rutas en la red, asegurando que los recursos correctos estén reservados para cada conexión.

- *Routing* (*OSPF-TE*): *GMPLS* utiliza *Open Shortest Path First-Traffic Engineering* (*OSPF-TE*) para enrutamiento. Este protocolo construye el mapa topológico de la red y disemina información sobre los recursos disponibles y los riesgos potenciales.

- *Path Computation Element* (PCE): A veces, las redes *GMPLS* dependen de un controlador centralizado conocido como el Elemento de Cálculo de Ruta (*PCE*). El *PCE* ayuda a calcular las mejores rutas para que los datos tomen, especialmente en redes complejas o de múltiples dominios.

Para aprovechar al máximo las capacidades de las *EONs*, los protocolos *GMPLS* necesitan algunas extensiones:

- *Bandwidth Variable transponders (BVTs)*: En las *EONs*, los *BVTs* se utilizan para ajustar dinámicamente el ancho de banda y el formato de modulación de las señales ópticas. El plano de control tiene que gestionar estos *transponders* de manera efectiva, asegurando que estén configurados para coincidir con las necesidades actuales de la red.

- *Flexible Spectrum Allocation*: Las *EONs* permiten un espaciado de canal flexible, lo que significa que el plano de control debe poder asignar slots de frecuencia con diferentes anchos dependiendo de la demanda. *GMPLS* necesita soportar esto gestionando estos slots y asegurando que no haya superposición entre las conexiones.

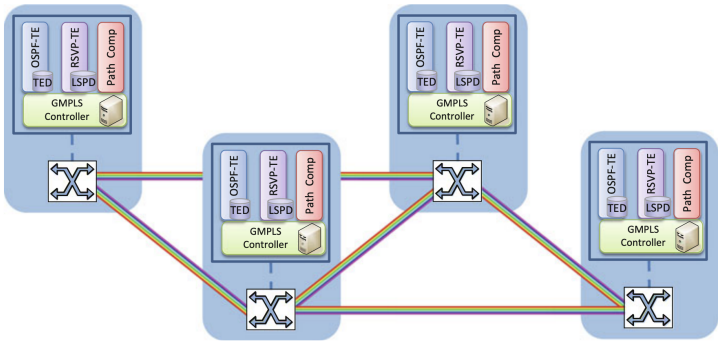

Figura 2.11: Red óptica controlada por GMPLS. Fuente: (López y Velasco, 2016)

Otra tecnología que cambió la forma en que se gestionan las redes es la *Software Defined Networking (SDN)* (López y Velasco, 2016). *SDN* es un enfoque de red moderno que separa la lógica de control de la red de los dispositivos de hardware, permitiendo una gestión más flexible y dinámica de las funciones de red. En lugar de tener la lógica de control integrada en cada dispositivo de red, *SDN* centraliza esta lógica en un controlador de red. La arquitectura *SDN* consta de tres capas principales que se pueden ver en la Figura 2.12:

- Capa de Aplicación: Aquí es donde residen las aplicaciones y servicios de red. Estos podrían incluir cosas como balanceadores de carga, firewalls o herramientas de monitoreo de red.

- Capa de Control: Esta capa actúa como el cerebro de la *SDN*, albergando el controlador (o controladores) *SDN*. El controlador gestiona la red interactuando con el hardware (plano de datos) basado en los requisitos establecidos por las aplicaciones en la capa de aplicación.

- Capa de Infraestructura: También conocida como plano de datos, aquí es donde reside el hardware de red real (por ejemplo, switches, routers, nodos ópticos). Estos dispositivos reenvían paquetes basados en las instrucciones recibidas de la capa de control.

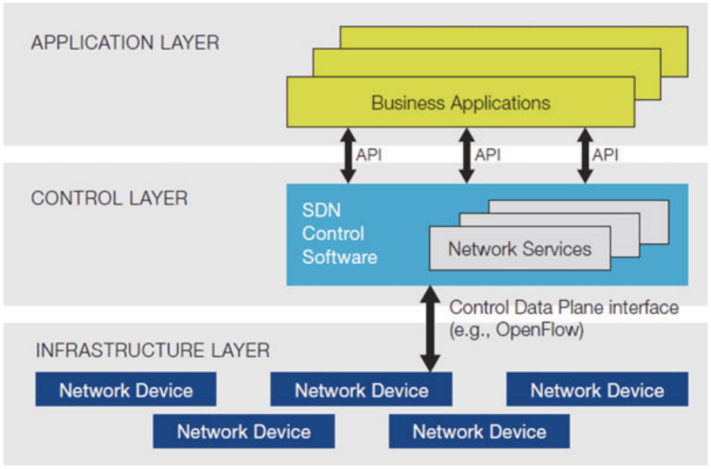

Figura 2.12: Arquitectura de una red definida por software (*SDN*). Fuente: (Open Networking Foundation (ONF), 2012)

A su vez, en la Figura 2.12 se puede ver cómo estas capas interactúan a través de interfaces:

- *Northbound Interface*: Conecta la Capa de Aplicación con la Capa de Control, típicamente a través de APIs.

- *Southbound Interface*: Conecta la Capa de Control con la Capa de Infraestructura.

En las redes tradicionales, los dispositivos de red tienen software y hardware integrados, lo que hace que cualquier cambio sea complejo y dependiente del hardware. *SDN* aborda esto separando la lógica de control del hardware, lo que permite más programabilidad y flexibilidad. Esta separación es similar a cómo los sistemas operativos de computadora están separados del hardware, lo que permite que diferentes aplicaciones se ejecuten en la misma plataforma de hardware. Se puede ver lo mencionado en la Figura 2.13.

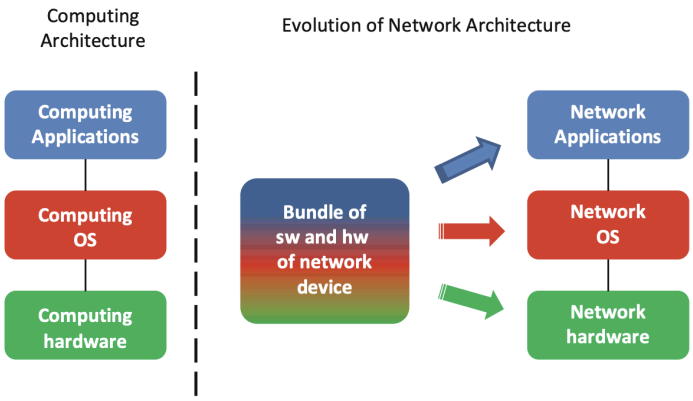

Figura 2.13: Cómo *SDN* cambia la arquitectura de red. Fuente: (López y Velasco, 2016)

El protocolo más utilizado en *SDN*, especialmente para controlar cómo se reenvían los paquetes por los switches y routers en la red, es OpenFlow (McKeown y cols., 2008). OpenFlow es un protocolo de comunicación que permite que el controlador *SDN* programe directamente el comportamiento de los dispositivos de red (McKeown y cols., 2008).

Sin embargo, implementar *SDN* en redes ópticas presenta algunos desafíos y oportunidades como:

- Ingeniería de tráfico: El control centralizado de *SDN* permite una ingeniería de tráfico más sofisticada, optimizando cómo fluyen los datos a través de la red. Esto es particularmente útil en redes ópticas, donde gestionar recursos como longitud de onda y espectro de manera eficiente es crítico.

- Virtualización de la red: *SDN* permite la creación de múltiples redes ópticas virtuales (*slices*) sobre una sola infraestructura física. En el contexto de las redes ópticas, esto significa que se pueden desplegar y operar múltiples servicios ópticos de manera aislada pero concurrente, cada uno con su propia configuración óptima, lo que es importante para manejar diversos tipos de tráfico en la misma red.

- Integración con tecnologías ópticas: *SDN* necesita adaptarse a las características específicas de las redes ópticas, como la naturaleza analógica de las señales ópticas, los formatos de modulación variables y la necesidad de gestionar impedimentos físicos como la dispersión y las no linealidades.

A su vez, presenta desafíos y consideraciones como: abstracción de recursos ópticos ya que las redes ópticas tienen propiedades físicas complejas que deben ser abstraídas de manera que sean manejables por los controladores *SDN* sin perder detalles esenciales, escalabilidad y confiabilidad ya que el control

centralizado en *SDN* puede llevar a problemas de escalabilidad y confiabilidad, especialmente en redes grandes y rendimiento y recuperación ya que si bien *SDN* ofrece un control y flexibilidad mejorados, debe poder responder rápidamente a fallas en la red.

2.2. Aprendizaje automático

Nos referimos a aprendizaje automático como una subdisciplina de la inteligencia artificial que estudia la construcción de sistemas que aprenden a realizar una tarea de forma automática a partir de un conjunto de datos (Mitchell, 1997) (Jurafsky y Martin, 2009). Se van a explicar diferentes tipos de algoritmos que van a permitir al lector un mejor entendimiento de los trabajos que se van a presentar en las siguientes secciones. En las siguientes subsecciones se va a presentar una clasificación de los algoritmos de aprendizaje automático y se va a explicar cada uno de ellos.

2.2.1. Aprendizaje supervisado

El aprendizaje supervisado es un enfoque fundamental en el aprendizaje automático donde el objetivo es aprender una asignación de variables de entrada (*features*) a variables de salida (en esta tesis se va a hacer referencia a estos como *targets*) basada en un conjunto de datos etiquetado (Bishop, 2006). Este enfoque se utiliza ampliamente en aplicaciones como reconocimiento de voz, detección de spam y reconocimiento de objetos.

En el aprendizaje supervisado, tenemos un conjunto de datos que consiste en pares: vectores de entrada x y sus salidas correspondientes y. El objetivo es predecir la salida y para una nueva entrada x no vista. La salida y puede ser continua (un problema de regresión) o discreta (un problema de clasificación).

El conjunto de datos utilizado para el entrenamiento consta de N ejemplos, cada uno conteniendo un vector de entrada x y su salida correspondiente y. El modelo aprende una función $y(x)$ a partir de este conjunto de datos, que luego se puede utilizar para predecir salidas para nuevas entradas.

Los modelos de aprendizaje supervisado se pueden clasificar en dos tipos principales: modelos paramétricos y modelos no paramétricos.

- Modelos paramétricos: Los modelos paramétricos asumen que la función que se está aprendiendo se puede describir mediante un número fijo de parámetros. Una vez que se han aprendido estos parámetros, el modelo descarta los datos de entrenamiento y se basa únicamente en los parámetros aprendidos para hacer predicciones.

  Ejemplo: Modelos lineales

  Un modelo lineal es un tipo básico de modelo paramétrico donde la salida es una combinación lineal de las características de entrada. Por ejemplo, en la regresión lineal, el modelo predice la salida y como:

  $y(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_mx_m$

donde $w = (w_0, w_1, \ldots, w_m)$ son los parámetros aprendidos a partir de los datos de entrenamiento. Estos parámetros definen la relación lineal entre las características de entrada y la salida.

Redes neuronales (NNs) como modelos paramétricos

Las redes neuronales son un tipo más complejo de modelo paramétrico. Una red neuronal consta de capas de unidades (o neuronas), donde cada unidad aplica una función no lineal a una suma ponderada de sus entradas. Los parámetros de la red neuronal son los pesos w que se ajustan durante el proceso de entrenamiento.

Las redes neuronales son poderosas porque pueden aproximar funciones complejas y no lineales. Están organizadas en capas, típicamente con una capa de entrada, varias capas ocultas y una capa de salida. Las capas entre la entrada y la salida se conocen como capas ocultas, y cada neurona en estas capas transforma los datos de entrada utilizando una función de activación.

Funciones de activación: Las funciones de activación comúnmente utilizadas incluyen la sigmoide logística, la tangente hiperbólica y ReLU (Unidad Lineal Rectificada). Estas funciones introducen no linealidad en el modelo, permitiéndole capturar patrones complejos en los datos.

Entrenamiento de redes neuronales: Las redes neuronales se entrenan minimizando una función de error (también llamada función de pérdida) que mide la diferencia entre las salidas predichas y las salidas reales en los datos de entrenamiento. El método más común para entrenar redes neuronales es el algoritmo de retropropagación, que calcula el gradiente de la función de error con respecto a los pesos y actualiza los pesos para reducir el error.

Preprocesamiento y normalización de características: Antes del entrenamiento, es común normalizar las características, asegurando que tengan una media de cero y una desviación estándar de uno. Este paso de normalización ayuda al modelo a converger más rápido y a tener un mejor rendimiento porque la mayoría de los algoritmos de aprendizaje automático asumen que las características están en una escala similar.

- Modelos no paramétricos: Los modelos no paramétricos no asumen una forma fija para la función que se está aprendiendo. En cambio, estos modelos retienen los datos de entrenamiento y utilizan estos datos para hacer predicciones.

  Ejemplo: *k-Nearest Neighbors (KNN)*

  KNN es un modelo no paramétrico simple y poderoso utilizado para clasificación y regresión. En KNN, las predicciones se realizan encontrando los k ejemplos de entrenamiento más cercanos al nuevo punto de entrada y utilizando estos ejemplos para hacer una predicción. Para clasificación, la etiqueta de clase más común entre los vecinos más cercanos se asigna

al nuevo punto. Para regresión, la salida es típicamente el promedio de las salidas de los vecinos más cercanos.

Elección de k: La elección de k puede afectar significativamente el rendimiento del modelo. Un k pequeño hace que el modelo sea sensible al ruido en los datos, mientras que un k grande puede suavizar demasiado las predicciones. La validación cruzada se utiliza a menudo para seleccionar un k óptimo.

- *Support Vector Machines (SVMs)*: SVMs son otro modelo no paramétrico popular utilizado principalmente para clasificación. SVMs funcionan encontrando el hiperplano que mejor separa los datos en diferentes clases, con el margen máximo entre los puntos de cada clase (llamados vectores de soporte). El kernel es una característica clave de SVMs que permite operar en un espacio de alta dimensión sin calcular explícitamente las coordenadas en ese espacio.

  Elección del kernel: La elección del kernel puede afectar significativamente el rendimiento del modelo. Los kernels comunes incluyen el kernel lineal, el kernel polinómico y el kernel radial (también conocido como kernel gaussiano). La validación cruzada se utiliza a menudo para seleccionar un kernel óptimo.

2.2.2. Aprendizaje no supervisado

El aprendizaje no supervisado es un enfoque en el aprendizaje automático donde, a diferencia del aprendizaje supervisado, no se proporcionan salidas etiquetadas en los datos de entrenamiento (Bishop, 2006). En cambio, el aprendizaje no supervisado tiene como entrada un conjunto de datos con vectores de entrada x y el objetivo es encontrar patrones o estructuras en los datos.

El aprendizaje no supervisado se utiliza comúnmente en aplicaciones como *clustering*, reducción de dimensionalidad y detección de anomalías. Las medidas son típicamente definidas por una función de distancia, como la distancia euclidiana, pero pueden variar dependiendo de la naturaleza de los datos y la aplicación específica.

Métodos de *clustering* Los métodos de *clustering* se pueden clasificar en diferentes tipos, como métodos de particionamiento, métodos jerárquicos, métodos basados en densidad y métodos basados en modelos. Uno de los algoritmos de *clustering* más conocidos y ampliamente utilizados es *k-means* debido a su simplicidad y eficiencia.

k-means es un método de particionamiento que divide el conjunto de datos en un número preespecificado de grupos, denotado por k. El algoritmo opera de forma iterativa y se puede resumir en los siguientes pasos:

- Inicialización: Comienza seleccionando aleatoriamente k centros de clúster iniciales, a menudo denominados centroides.

- Paso de asignación: Asigna cada punto de datos en el conjunto de datos al centroide más cercano, basado en una métrica de distancia elegida (típicamente la distancia euclidiana). Este paso forma k clústeres.
- Paso de actualización: Recalcula el centroide de cada clúster promediando las posiciones de todos los puntos de datos asignados a ese clúster.
- Iteración: Repite los pasos de asignación y actualización hasta que los centroides dejen de cambiar significativamente o se alcance un número máximo de iteraciones.

El algoritmo tiene como objetivo minimizar la varianza entre los elementos de un clúster, que es la suma de las distancias al cuadrado entre los puntos de datos y sus centroides correspondientes. Sin embargo, debido a que *k-means* minimiza una función objetivo no convexa, puede converger a un mínimo local en lugar del óptimo global, lo que significa que el agrupamiento final puede depender de la elección inicial de los centroides. A su vez, *k-means* es bastante sensible a valores atípicos.

2.2.3. Aprendizaje por refuerzo

El aprendizaje por refuerzo (RL) es un paradigma de aprendizaje automático utilizado en escenarios donde un agente aprende a tomar decisiones interactuando con un entorno, y se utiliza en aplicaciones como robótica, finanzas y gestión de inventarios (Musumeci y cols., 2019). El objetivo en RL es aprender una política, que es una asignación de estados del entorno a acciones que el agente debe tomar para maximizar su recompensa acumulada a lo largo del tiempo. A diferencia de otros paradigmas de aprendizaje, RL se basa en el ensayo y error y en la retroalimentación, donde el agente recibe recompensas basadas en sus acciones también considerando el impacto a largo plazo de sus decisiones.

RL utiliza comúnmente los Procesos de Decisión de Markov (MDPs) para modelar el entorno. En un MDP, el agente observa un estado del entorno y toma una acción, lo que lo lleva a un nuevo estado y recibe una recompensa en el proceso. El agente busca descubrir la política óptima que maximiza la recompensa esperada a lo largo del tiempo. Uno de los algoritmos de RL más comunes es Q-learning, un método sin modelo que estima la función de valor de acción óptima, o función Q, que representa el retorno esperado de tomar una acción particular en un estado dado. El agente utiliza esta función Q para seleccionar la acción que maximiza su recompensa esperada.

2.2.4. Sobreajuste, subajuste y regularización

El sobreajuste y el subajuste son desafíos críticos en el aprendizaje automático, particularmente en la selección de modelos. El sobreajuste ocurre cuando un modelo es demasiado complejo para el conjunto de datos disponible, lo que lleva al modelo a ajustarse estrechamente a los datos de entrenamiento, incluido el ruido y los valores atípicos, pero resulta en una mala generalización, lo que

significa que se desempeña mal en datos nuevos y no vistos. Por otro lado, el subajuste ocurre cuando un modelo es demasiado simple para capturar la estructura subyacente de los datos, lo que lleva a un mal rendimiento tanto en los conjuntos de entrenamiento como de prueba.

Para evaluar el rendimiento de un modelo y prevenir el sobreajuste o el subajuste, el conjunto de datos se divide típicamente en un conjunto de entrenamiento y un conjunto de prueba. Alternativamente, se puede utilizar la validación cruzada, especialmente con datos limitados, donde el conjunto de datos se divide en k subconjuntos, y el modelo se entrena k veces, cada vez utilizando un subconjunto diferente para la validación. El sobreajuste se indica por un error de entrenamiento bajo pero un error de prueba alto, mientras que el subajuste resulta en altos errores en ambos conjuntos.

Para mitigar estos problemas, la regularización es una técnica común. La regularización agrega un término de penalización a la función de error durante el entrenamiento, desalentando modelos excesivamente complejos al empujar los parámetros hacia valores más pequeños. Las formas comunes de introducir regularización son el decaimiento de los pesos (*L2 regularization*) y *lasso* (*L1 regularization*), cada uno controlado por un coeficiente de regularización λ, que equilibra el compromiso entre ajustar los datos de entrenamiento y mantener el modelo simple. Seleccionar el λ óptimo generalmente implica probar un rango de valores y elegir el que minimiza el error de validación. En las redes neuronales, la exclusión (*dropout*) es otra técnica de regularización efectiva, donde las unidades y sus conexiones se eliminan temporalmente durante el entrenamiento para prevenir el sobreajuste al fomentar que la red aprenda características más robustas.

2.3. Exploración y preparación de datos

En esta sección se va a presentar una descripción de las herramientas y técnicas utilizadas para explorar y preparar los datos antes de entrenar los modelos de aprendizaje automático.

2.3.1. Análisis exploratorio de datos (EDA)

El análisis exploratorio de datos (EDA) es un enfoque crítico en el aprendizaje automático que implica explorar y visualizar los datos para comprender mejor sus *features* y relaciones. El EDA ayuda a identificar patrones, tendencias y anomalías en los datos, lo que puede guiar la selección de *features*, la elección del modelo y la evaluación del rendimiento. No nos referimos a EDA como una técnica o serie de pasos específicos, sino como un enfoque general para comprender los datos y extraer información significativa de ellos. En esta tesis se utilizó un proceso iterativo de EDA que incluyó las siguientes etapas: análisis semántico de las *features*, visualización y resumen de la distribución de los datos, identificación de correlaciones y relaciones entre las *features*, y detección de valores atípicos y datos faltantes. Para el desarrollo de esta tesis se utilizó

Python y en particular para el análisis exploratorio de datos se utilizaron las siguientes bibliotecas de desarrollo:

- *Pandas*: *Pandas* es una biblioteca de análisis de datos de código abierto que proporciona estructuras de datos de alto rendimiento y fáciles de usar, como *DataFrames* que permiten manipular y analizar datos tabulares. (McKinney, 2010)
- *NumPy*: *NumPy* es una biblioteca de Python que proporciona soporte para arreglos y matrices multidimensionales, lo que facilita la manipulación de datos numéricos. (Harris y cols., 2020)
- *Matplotlib*: *Matplotlib* es una biblioteca de visualización de datos de Python que permite crear gráficos de alta calidad, como gráficos de líneas, histogramas y diagramas de dispersión. (Hunter y D., 2007)
- *Seaborn*: *Seaborn* es una biblioteca de visualización de datos de Python basada en *Matplotlib* que proporciona una interfaz de alto nivel para crear gráficos informativos. (Waskom, 2021)
- *YData*: *YData* es una biblioteca de Python que proporciona herramientas para el análisis y visualización de datos, incluida la creación de gráficos interactivos y la generación de informes. (YData, 2023)

YData

Para el análisis exploratorio de datos se utilizó la biblioteca *YData*, que en una sola línea de código permite realizar un análisis exploratorio de datos completo. Genera un reporte en formato HTML que incluye información sobre la distribución de las *features*, la correlación entre las *features* y los *targets*, gráficas y detección de valores faltantes y valores atípicos. Es la herramienta de análisis exploratorio de datos más completa que se utilizó en este proyecto.

Seaborn

Seaborn es una biblioteca de visualización de datos que te permite, a alto nivel, crear gráficas complejas que con *Matplotlib* serían más complicadas de implementar. En particular, se utilizó en este proyecto de grado para la visualización de los mapa de calor de correlación entre las *features* y como estos se relacionan con los *targets*.

Conclusiones del análisis exploratorio de datos

Una de las salidas del proceso de EDA fue la identificación de las *features* más importantes para predecir los *targets*. Esto se logró mediante la visualización de la correlación entre las *features* y los *targets*, y la identificación de las *features* que tenían una correlación más fuerte con los *targets*. Además, se identificaron las *features* que tenían una alta correlación entre sí, lo que podría indicar que estas *features* contienen información redundante.

2.3.2. Técnicas de preprocesamiento de datos

El preprocesamiento de datos es una etapa crítica en el aprendizaje automático que implica transformar y limpiar los datos antes de entrenar un modelo. En esta tesis, las tres técnicas de preprocesamiento de datos más utilizadas fueron la ingeniería de *features* (basada muchas veces en el conocimiento del dominio y la salida de la etapa de EDA), la normalización y una técnica llamada análisis de componentes principales (PCA).

Normalización

Una de las etapas más importantes del preprocesamiento de datos es la normalización. Con pocas excepciones, los algoritmos de aprendizaje automático no tienen un buen rendimiento cuando las *features* tienen diferentes escalas (Géron, 2019). La normalización es el proceso de escalar las *features* a un rango común, generalmente entre 0 y 1 o -1 y 1. En este trabajo se utilizó la normalización *StandardScaler* de *Scikit-learn* (Virtanen y cols., 2020). Esta técnica de normalización escala las *features* de manera que tengan una media de cero y una desviación estándar de uno. El cálculo que realiza es el siguiente:

$$z = \frac{x - \mu}{\sigma} \tag{2.1}$$

donde z es el valor normalizado, x es el valor original, μ es la media de la *features* y σ es la desviación estándar de la *features*.

Análisis de componentes principales (PCA)

El análisis de componentes principales (PCA) es una técnica que es usada comúnmente para reducir la dimensionalidad de los datos, para compresión de datos, para extracción de *features* y para visualización de datos (Bishop, 2006). La técnica PCA es una técnica de reducción de dimensionalidad que se utiliza para (en el caso de este proyecto) simplificar grandes conjuntos de datos mientras se preserva la mayor cantidad de información importante posible. Lo hace identificando las direcciones (llamadas componentes principales) a lo largo de las que la variación en los datos es la mayor.

En muchas aplicaciones (como el actual trabajo), los datos pueden tener un gran número de *features* (o dimensiones). Por ejemplo, si se trabaja con imágenes, cada píxel es una *feature*. A medida que el número de *features* crece, se vuelve costoso computacionalmente analizar o visualizar los datos. Además, muchas *features* pueden ser redundantes o ruidosas, lo que dificulta entrenar modelos de manera efectiva. PCA ayuda a reducir la complejidad encontrando un conjunto más pequeño de "nuevas" *features*, o componentes, que aún capturan la mayor parte de la información de los datos originales. Esta representación reducida hace que los datos sean más fáciles de trabajar mientras se mantiene su estructura central.

La transformación PCA se puede resumir en los siguientes pasos:

- Identificar las direcciones (componentes principales) a lo largo de las cuales la variación en los datos es la mayor.
- Proyectar los datos en estas nuevas direcciones, lo que resulta en una nueva representación de los datos.

Para eso, PCA encuentra ejes ortogonales (direcciones) que capturan tanta varianza (dispersión) en los datos como sea posible. El primer componente principal captura la mayor varianza, el segundo captura la mayor varianza que es ortogonal (no correlacionada) con el primero, y así sucesivamente. Matemáticamente, PCA hace esto encontrando los vectores y valores propios de la matriz de covarianza de los datos. Los vectores propios corresponden a las direcciones de los componentes principales, y los valores propios nos dicen cuánta varianza es capturada por cada componente principal.

En este caso, se utilizó PCA de la biblioteca *Scikit-learn* para reducir la dimensionalidad de los datos.

2.4. Modelos de aprendizaje automático utilizados

En esta sección se va a explicar más en detalle los modelos de aprendizaje automático que se utilizaron en este proyecto de grado. Estos se dividen en modelos arborescentes y redes neuronales.

2.4.1. Aprendizaje basado en árboles de decisión

El aprendizaje basado en árboles de decisión es un método para aproximar funciones objetivo discretas, en el que la función aprendida se representa mediante un árbol de decisión. Los árboles de decisión son populares debido a su capacidad para generar modelos que son fácilmente comprensibles por humanos, y que pueden ser expresados como conjuntos de reglas condicionales (*"if-then"*) para mejorar la interpretación (Mitchell, 1997).

Los árboles de decisión clasifican las instancias ordenándolas desde la raíz hasta algún nodo hoja, que proporciona la clasificación de la instancia. Un árbol de decisión clasifica instancias o ejemplos organizando sus características desde el nodo raíz hasta un nodo hoja, donde cada nodo representa una prueba sobre un atributo del ejemplo, y cada rama corresponde a uno de los valores posibles de ese atributo. A medida que un ejemplo desciende por el árbol, se evalúan diferentes atributos hasta que se llega a una clasificación final en un nodo hoja, donde la clasificación de la instancia es determinada.

En el ejemplo de la Figura 2.14, se ilustra un árbol de decisión que clasifica si un día es adecuado para jugar al tenis. Cada nodo del árbol realiza una pregunta sobre el valor de un atributo. En el caso de la decisión de jugar tenis, el nodo raíz evalúa el atributo *Outlook* (pronóstico del clima), que puede tomar tres valores posibles: *Sunny*, *Overcast*, o *Rain*. Dependiendo del valor del atributo

en un ejemplo particular, el árbol sigue la rama correspondiente: Si *Outlook* es *Sunny*, el siguiente nodo revisa el atributo *Humidity* (humedad), que puede ser *High* o *Normal*. Si la humedad es alta, el resultado es *No* (no se juega), mientras que si es normal, el resultado es *Yes* (sí se juega). Si *Outlook* es *Overcast*, el árbol predice directamente *Yes* sin necesidad de más pruebas, ya que en todos los casos anteriores, esta condición siempre resultó en una decisión positiva. Si *Outlook* es *Rain*, el siguiente nodo revisa el atributo *Wind* (viento), que puede ser *Strong* o *Weak*. Si el viento es fuerte, se predice *No*, y si es débil, se predice *Yes*. Este proceso de clasificación comienza en la raíz y desciende hacia las hojas, haciendo pruebas en los atributos de la instancia en cada nodo, y siguiendo las ramas según los valores de los atributos. Finalmente, en el nodo hoja, se obtiene la clasificación final, ya sea *Yes* o *No*.

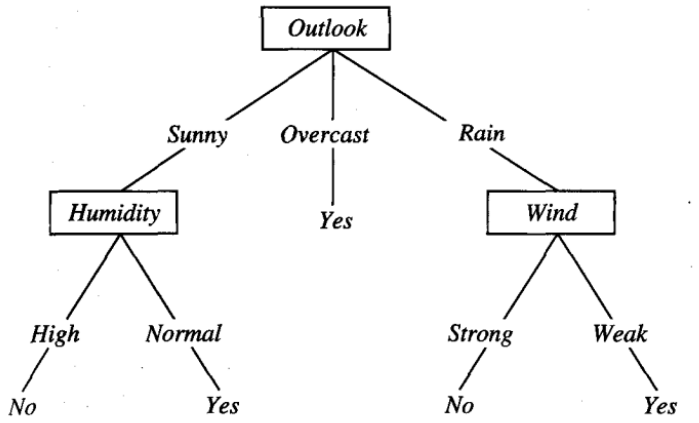

Figura 2.14: Ejemplo de un árbol de decisión que clasifica si un día es adecuado para jugar al tenis. Fuente: (Mitchell, 1997)

Una de las características interesantes de los árboles de decisión es que pueden ser representados como expresiones lógicas formadas por disyunciones de conjunciones. Por ejemplo, el árbol de decisión de la Figura 2.14 se puede expresar como la siguiente regla:

$$\text{Outlook} = \begin{cases} \text{Sunny} & \wedge\ \text{Humidity} = \text{Normal}, \\ \text{Overcast}, & \\ \text{Rain} & \wedge\ \text{Wind} = \text{Weak}. \end{cases} \tag{2.2}$$

2.4.2. Modelos de *Bagging* y *Boosting*

Para el trabajo actual se decidió utilizar un modelo "liviano"de aprendizaje automático para probar su performance y compararlo con otros modelos más complejos como redes neuronales. Para ello se evaluó el uso de modelos de

bagging y *boosting* que son técnicas que se utilizan para mejorar la precisión de los modelos de aprendizaje automático.

Una manera de mejorar los modelos de aprendizaje automático es mediante la combinación o ensamble de múltiples modelos. La manera más simple de construir un ensamble es promediando las predicciones de un conjunto de modelos individuales (Bishop, 2006). La razón detrás de esto se basa en cómo entendemos los errores de predicción. Hay dos fuentes principales de error en un modelo: el sesgo y la varianza. El sesgo es el error que ocurre cuando el modelo es demasiado simple y no captura la complejidad completa del problema. Es como intentar aproximar una curva usando una línea recta. La varianza es el error que ocurre cuando el modelo es demasiado sensible a los puntos específicos en los que fue entrenado. Si un modelo es demasiado flexible o sensible a cada uno de los datos de entrenamiento, puede sobreajustar, lo que significa que se desempeña muy bien en los datos de entrenamiento pero tiene problemas con datos nuevos y no vistos. Al promediar las predicciones de múltiples modelos, podemos reducir el impacto de la varianza. Los modelos individuales pueden cometer diferentes errores, pero cuando se combinan, esos errores pueden cancelarse entre sí, lo que lleva a mejores predicciones en general. Este método ayuda a encontrar un equilibrio entre modelos que son demasiado simples (alto sesgo) y modelos que son demasiado complejos (alta varianza).

En la práctica, se tiene un solo conjunto de datos y se necesita encontrar una manera de introducir variabilidad entre los diferentes modelos dentro del ensamble. Un enfoque es utilizar conjuntos de datos de *bootstrap*. Estos conjuntos de datos se generan tomando muestras aleatorias con reemplazo del conjunto de datos original. Cada conjunto de datos de *bootstrap* se utiliza para entrenar un modelo predictivo diferente. Se considera un problema de regresión en el que se intenta predecir el valor de una variable continua. Supongamos que generamos M conjuntos de datos de *bootstrap* y luego usamos cada uno para entrenar una copia separada $y_m(x)$ de un modelo predictivo, donde $m = 1, \ldots, M$. La predicción del ensamble es dada por:

$$y_{\text{COM}}(x) = \frac{1}{M} \sum_{m=1}^{M} y_m(x). \tag{2.3}$$

Este procedimiento es conocido como *bagging* (Breiman, 1996).

Bajo la asunción de que los errores no están correlacionados, se demuestra que el error esperado del ensamble se reduce en un factor de M simplemente promediando M versiones del modelo. Sin embargo, en la práctica, los errores suelen estar altamente correlacionados, y la reducción en el error general es generalmente pequeña. Se puede demostrar que el error esperado del ensamble no excederá el error esperado de los modelos constituyentes, por lo que $E_{\text{COM}} \leq E_{\text{AV}}$.

Un método más sofisticado para construir un ensamble es mediante el uso de *boosting* (Freund, Schapire, y Abe, 1999). *Boosting* es una técnica para combinar múltiples clasificadores base para producir una forma de ensamble cuyo rendimiento puede ser significativamente mejor que el de cualquiera de los cla-

sificadores base (Bishop, 2006). *Boosting* puede dar buenos resultados incluso si los clasificadores base tienen un rendimiento que es solo ligeramente mejor que el clasificador aleatorio, y por lo tanto a veces los clasificadores base se conocen como *weak learners*. Originalmente fue diseñado para resolver problemas de clasificación, pero *boosting* también se puede extender a la regresión (Friedman, 2001). La principal diferencia entre *boosting* y los métodos de ensamble es que los clasificadores base se entrenan en secuencia, y cada clasificador base se entrena haciendo foco en los datos clasificados erróneamente por los clasificadores anteriores. Esto se logra mediante la asignación de pesos a cada punto de datos, donde los puntos que son clasificados erróneamente por uno de los clasificadores base tienen un mayor peso cuando se utilizan para entrenar el siguiente clasificador en la secuencia. Una vez que todos los clasificadores han sido entrenados, sus predicciones se combinan a través de una forma ponderada como se puede ver en la Figura 2.15.

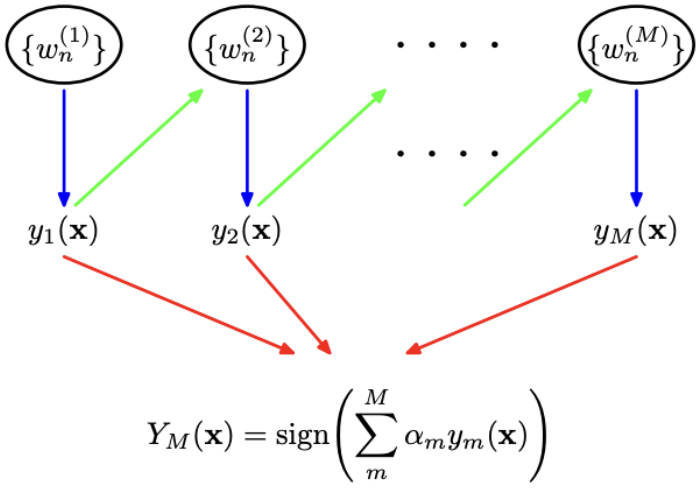

Figura 2.15: Esquema del funcionamiento de *boosting*. Fuente: (Bishop, 2006)

HistGradientBoostingRegressor

En el trabajo actual se utilizó el modelo *HistGradientBoostingRegressor* de la biblioteca *Scikit-learn*. Este modelo es una implementación de *gradient boosting* donde los clasificadores base son árboles de decisión. Pero, a pesar de que los modelos de *gradient boosting* son muy efectivos, pueden ser lentos de entrenar. Esto pasa porque los árboles deben ser creados y agregados secuencialmente, a diferencia de otros modelos de ensamble como *random forest* donde los clasificadores individuales del ensamble pueden ser entrenados en paralelo, utilizando múltiples CPUs. Una forma de acelerar la fase de entrenamiento es mediante la reducción del número de valores para las *features* de entrada. Esto se puede

lograr mediante la discretización o agrupación de valores en un número fijo de grupos. Esto permite que el árbol de decisión opere sobre el grupo en lugar de operar sobre valores específicos en el conjunto de datos de entrenamiento. Esta aproximación de los datos de entrada tiene poco impacto en la habilidad de predicción del modelo y acelera la construcción del árbol de decisión (Bishop, 2006). Además, se pueden utilizar estructuras de datos eficientes para representar la agrupación de los datos de entrada. Por ejemplo, se pueden utilizar histogramas y el algoritmo de construcción del árbol se puede adaptar aún más para el uso eficiente de los histogramas en la construcción de cada árbol. Por lo tanto, es común referirse a un algoritmo de *gradient boosting* que soporta histogramas en las bibliotecas modernas de aprendizaje automático como un *histogram-based gradient boosting* como lo es el caso de *HistGradientBoostingRegressor*.

2.4.3. Redes neuronales

Una familia de funciones muy utilizada en la actualidad para el aprendizaje automático son las redes neuronales. Una red neuronal es un modelo definido como una red de unidades, donde cada una tiene la responsabilidad de propagar entre ellas la entrada x de la red a su salida, realizando cálculos intermedios sobre x hasta transformarla en la salida $\hat{y}$ (Mitchell, 1997) (Jurafsky y Martin, 2009). El objetivo principal de una red neuronal es aproximar una función $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ mediante una función $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

Las redes neuronales se entrenan ajustando los pesos de las conexiones entre neuronas de manera que minimicen la diferencia entre la salida predicha $\hat{y}$ y la salida deseada y. Este ajuste se realiza utilizando algoritmos de optimización como el descenso de gradiente, los cuales buscan minimizar una función de costo o error que mide el desempeño del modelo (Bishop, 2006).

Multilayer Perceptron (MLP)

El *Multilayer Perceptron* (MLP) es una de las arquitecturas más básicas de redes neuronales artificiales y se conoce como una red neuronal *feedforward*. En MLP, las neuronas están organizadas en tres tipos de capas: una capa de entrada, una o más capas ocultas, y una capa de salida. La capa de entrada recibe los datos de entrada x, las capas ocultas procesan estos datos a través de transformaciones no lineales, y la capa de salida produce la predicción final $\hat{y}$. Cada conexión entre las neuronas tiene un peso asociado, que se ajusta durante el entrenamiento.

Una característica clave del MLP es que cada capa oculta introduce no linealidad al sistema mediante el uso de funciones de activación, como la función sigmoide, tangente hiperbólica (tanh), o la función de rectificación lineal unitaria ($ReLU$) (Bishop, 2006). Estas funciones permiten que el modelo capture patrones complejos en los datos. El proceso de aprendizaje de un MLP implica minimizar la función de costo utilizando el algoritmo de retropropagación (*backpropagation*), que calcula el gradiente de la función de error con respecto a los pesos y los ajusta utilizando descenso de gradiente. En la siguiente ecuación se

muestra la forma general de un MLP con dos capas ocultas:

$$h(x) = \sigma(W_2 \cdot \sigma(W_1 \cdot x + b_1) + b_2), \tag{2.4}$$

donde W_1 y W_2 son las matrices de pesos entre las capas de la red, b_1 y b_2 son los vectores de sesgo, y σ es la función de activación. El aprendizaje en MLP consiste en ajustar los valores de W y b para reducir el error de predicción.

En la imagen 2.16 se puede ver la arquitectura de una red MLP típica.

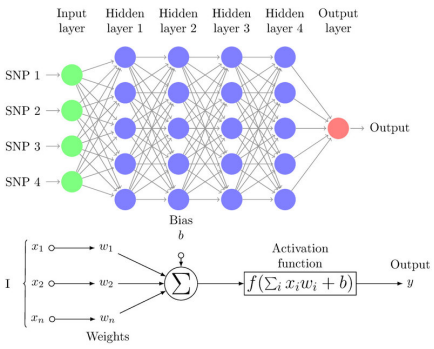

Figura 2.16: Arquitectura de una red MLP. Fuente: (*Multilayer Perceptron*, s.f.)

En el presente trabajo se utilizó la implementación proveída por la biblioteca *PyTorch* (Paszke y cols., 2019) para implementar un MLP.

Redes neuronales recurrentes (RNNs)

Aunque en este trabajo no se utilizaron es imporante introducir brevemente el funcionamiento de un tipo de redes neuronales que comúnmente se utilizan para pronósticos y que suelen trabajar con datos de entrada secuenciales (como puede ser, los estados de una red de fibra óptica a lo largo del tiempo). Las Redes Neuronales Recurrentes (RNNs) son una extensión de las redes neuronales tradicionales que están diseñadas específicamente para manejar datos secuenciales. A diferencia de las redes *feedforward* (como el *Multilayer Perceptron*, donde las conexiones solo van hacia adelante), las RNNs poseen conexiones recurrentes que les permiten mantener un estado interno que refleja información de pasos anteriores en la secuencia. Este estado interno es importante cuando las entradas están organizadas temporal o secuencialmente, como en el caso de series temporales, datos de voz o texto.

Matemáticamente, una RNN puede describirse como una función que toma una secuencia de entrada $\{x_1, x_2, \ldots, x_T\}$ de longitud T y produce una secuencia de salidas $\{y_1, y_2, \ldots, y_T\}$, con una dependencia recurrente entre los estados ocultos en cada paso temporal. El estado oculto en el tiempo t, denotado por h_t, es una función del estado oculto anterior h_{t-1} y la entrada en el tiempo actual x_t:

$$h_t = f(W_h h_{t-1} + W_x x_t + b_h)$$

Aquí, W_h es la matriz de pesos que conecta el estado oculto anterior al actual, W_x es la matriz de pesos que conecta la entrada x_t al estado oculto, y b_h es un vector de sesgo. La función $f(\cdot)$ es una función de activación no lineal, típicamente una tangente hiperbólica $\tanh(\cdot)$ o una función sigmoide, que introduce la no linealidad en el modelo.

El valor de salida en cada paso y_t se obtiene como una función del estado oculto actual:

$$y_t = g(W_y h_t + b_y)$$

donde W_y es la matriz de pesos que conecta el estado oculto a la salida, b_y es un vector de sesgo, y $g(\cdot)$ es la función de activación de salida, que puede ser, por ejemplo, una softmax para problemas de clasificación o una función lineal para regresión.

A lo largo del trabajo también se mencionan las LSTM y las GRU que son extensiones de las RNNs que permiten manejar mejor el problema del desvanecimiento del gradiente. Este problema ocurre cuando se entrenan redes neuronales profundas y se propagan los gradientes a través de muchas capas, lo que puede resultar en gradientes que se vuelven muy pequeños y hacen que el entrenamiento sea inestable. Este problema es particularmente relevante en las RNNs, ya que los gradientes deben propagarse a través de múltiples pasos temporales.

2.5. Interpretación y evaluación de modelos de aprendizaje automático

Una vez que se han entrenado los modelos de aprendizaje automático, es importante evaluar su rendimiento y comprender cómo toman decisiones. En esta sección se presentan algunas técnicas comunes para interpretar y evaluar modelos de aprendizaje automático.

2.5.1. Interpretación de modelos

Cuando se habla de modelos de aprendizaje automático muchas veces se escucha el concepto de modelo de caja blanca y modelo de caja negra. Un modelo de caja blanca es aquel que es fácil de interpretar y explicar, como los árboles de decisión, que pueden ser visualizados y entendidos por humanos. Como ya vimos, muchas veces los árboles de decisión son representados como reglas de decisión que son fáciles de interpretar y que son reproducibles por humanos. Por otro lado, un modelo de caja negra es aquel que es difícil de interpretar y explicar, como las redes neuronales, que son modelos altamente no lineales y que pueden tener miles de parámetros. Esto hace que a pesar de que los modelos de redes neuronales hagan predicciones muy precisas, sea difícil entender cómo llegaron a esas predicciones. Por ejemplo, si una red neuronal

dice que una imagen contiene un gato, es difícil saber qué parte de la imagen contribuyó a esa predicción: si fueron los ojos, la nariz, las orejas, el fondo de la imagen, etc.

En el desarrollo es esta tesis se recurrió mucho a la interpretación de los modelos mediante los valores de *SHAP* (*SHapley Additive exPlanations*) (S. Lundberg, 2017). Los valores de Shapley permiten descomponer una predicción de un modelo de aprendizaje automático en las contribuciones individuales de cada *feature*. Esta descomposición se asegura de que las contribuciones se asignen de manera equitativa, considerando todas las posibles combinaciones de *features*. Matemáticamente, el problema de calcular los valores de Shapley puede describirse de la siguiente manera:

Dado un modelo de predicción f y un conjunto de características $\{x_1, x_2, \ldots, x_n\}$, la predicción para una entrada $x = (x_1, x_2, \ldots, x_n)$ es $f(x)$. Se quiere descomponer esta predicción como la suma de contribuciones individuales de cada característica. El valor de Shapley para la i-ésima característica x_i, denotado $\phi_i(f, x)$, mide su contribución promedio considerando todas las combinaciones posibles de otras características.

Para definir los valores de Shapley, se considera el poder de grupos de características, es decir, subconjuntos $S \subseteq N$, donde N es el conjunto completo de todas las características $\{1, 2, \ldots, n\}$. El valor de Shapley para la característica x_i se define como:

$$\phi_i(f, x) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} \left(f(S \cup \{i\}) - f(S)\right)$$

donde:

- S es un subconjunto de las n características que no incluye a i.
- $f(S)$ es el valor de la predicción del modelo considerando solo las características en S.
- $f(S \cup \{i\})$ es la predicción del modelo cuando la característica i se añade al conjunto S.
- $|S|$ es el tamaño del subconjunto S.
- n es el número total de características.

La fórmula implica que el valor de Shapley de una característica x_i es el promedio de las diferencias en las predicciones cuando la característica x_i se añade a un subconjunto de características S, ponderado por el número de formas de formar ese subconjunto S.

El cálculo exacto de los valores de Shapley puede ser computacionalmente costoso, ya que requiere evaluar todas las combinaciones posibles de características, lo que implica 2^n subconjuntos. En modelos con muchas características, este cálculo es imposible de realizar en la práctica. En (S. Lundberg, 2017) se propone el marco SHAP (*SHapley Additive exPlanations*), que introduce algoritmos

eficientes para aproximar los valores de Shapley en modelos complejos. SHAP no solo mantiene las propiedades deseables de los valores de Shapley, sino que también optimiza su cálculo utilizando técnicas específicas para tipos de modelos comunes. En el trabajo actual se utilizó la implementación de SHAP de la biblioteca *SHAP* (S. M. Lundberg y Lee, 2017) para interpretar los modelos de aprendizaje automático.

Métricas para clasificación

En los problemas de clasificación, se evalúa qué tan bien el modelo predice la clase del conjunto de datos asignado para evaluar. Las siguientes métricas son las más comunes:

Exactitud (*Accuracy*): La exactitud es una de las métricas más comunes y mide la proporción de predicciones correctas sobre el total de predicciones:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

donde:

- TP son los verdaderos positivos.
- TN son los verdaderos negativos.
- FP son los falsos positivos.
- FN son los falsos negativos.

Aunque es útil en problemas balanceados, la exactitud puede no dar la imagen completa del rendimiento del modelo en problemas donde las clases están desbalanceadas.

Precisión (*Precision*): La precisión mide la proporción de verdaderos positivos sobre el total de predicciones positivas hechas por el modelo:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Exhaustividad (Recall): El *recall* mide la proporción de verdaderos positivos sobre el total de elementos que deberían haber sido predichos como positivos:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Puntuación F1 *F1-Score*): La *puntuación F1* es la media armónica entre la precisión y el *recall*, y es útil cuando se necesita un equilibrio entre ambas métricas. La fórmula es:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Es particularmente útil cuando las clases están desbalanceadas.

Métricas para regresión

En los problemas de regresión, donde el objetivo es predecir valores continuos, se utilizan las siguientes métricas para medir el error y la capacidad predictiva del modelo:

Error Absoluto Medio (*Mean Absolute Error*, MAE): El *error absoluto medio* mide el valor promedio de las diferencias absolutas entre las predicciones y los valores reales:

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

donde y_i es el valor real, $\hat{y}_i$ es la predicción del modelo, y n es el número total de observaciones. Esta métrica es fácil de interpretar, ya que indica la magnitud promedio del error sin considerar su dirección.

Error Cuadrático Medio (*Mean Squared Error*, MSE): El *error cuadrático medio* mide el promedio de los errores al cuadrado. Penaliza más los errores grandes debido al término cuadrático:

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

Es útil cuando los grandes errores son más significativos que los pequeños.

Raíz del Error Cuadrático Medio (*Root Mean Squared Error*, RMSE): La *raíz del error cuadrático medio* es la raíz cuadrada del MSE. Devuelve el error en la misma escala que la variable objetivo, lo que facilita la interpretación:

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

El RMSE es particularmente útil cuando los errores grandes deben ser penalizados fuertemente.

Coeficiente de determinación R^2: El R^2 o *coeficiente de determinación* mide la proporción de la varianza de la variable objetivo que es explicada por las predicciones del modelo. Su fórmula es:

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

donde $\bar{y}$ es la media de los valores reales y_i. El R^2 varía entre 0 y 1, donde valores más cercanos a 1 indican que el modelo explica bien la variabilidad de los datos, y valores cercanos a 0 indican lo contrario. Si R^2 es negativo, significa que el modelo es peor que simplemente predecir la media $\bar{y}$.

2.6. Estado del arte: ciencia de datos sobre redes ópticas

Las redes de telecomunicaciones se han convertido en una fuente grande de distinto tipos de datos. Este tipo de información se puede extraer de registros de tráfico de red, alarmas de red, indicadores de calidad de señal, datos de comportamiento de usuarios, etc. Se requieren herramientas matemáticas avanzadas para extraer información significativa de estos datos y tomar decisiones relacionadas con el funcionamiento adecuado de las redes a partir de los datos generados por la red. Entre estas herramientas matemáticas, el aprendizaje automático (aprendizaje automático) se destaca como enfoque para hacer análisis de datos de red y automatizar la configuración de la red y la gestión de fallas.

El crecimiento de las redes ópticas hace que se introduzca más complejidad debido a la gran cantidad de parámetros de la red como configuraciones de enrutamiento, formato de modulación, tasa de símbolos, esquemas de codificación, etc. Esto motiva la adopción y desarrollo de técnicas de aprendizaje automático en el campo de las redes ópticas.

En (Musumeci y cols., 2019) , (Mata y cols., 2018), (Rafique y Velasco, 2018) y (Villa, Tipantuña, Guamán, Arévalo, y Arguero, 2023) se presentan revisiones exhaustivas sobre la aplicación de técnicas de aprendizaje automático en redes ópticas. En la capa física, los casos de uso de aprendizaje automático se dividen en 5 categorías (Musumeci y cols., 2019):

- Estimación de la calidad de transmisión (QoT) de *lightpaths* no establecidos: esto consiste en clasificar la QoT de un *lightpath* antes de su establecimiento en la red, utilizando aprendizaje supervisado entrenado con datos sintéticos o de campo, como un método potencialmente más rápido (en comparación con modelos analíticos) para evaluar la tasa de error de bits (BER), la relación señal-ruido (SNR) y la relación señal-ruido óptica ($OSNR$). Esto también consiste en predecir la QoT de *lightpaths* desplegados utilizando métodos de aprendizaje supervisado entrenados con datos históricos de QoT, para mantenimiento proactivo y optimización de margen de canal.

- Control de amplificadores ópticos o el control de las excursiones de potencia de canal en sistemas de línea óptica utilizando métodos de aprendizaje supervisado, no supervisado y de refuerzo.
- Reconocimiento de formato de modulación en receptores digitales coherentes utilizando aprendizaje supervisado y no supervisado.
- Mitigación de no linealidades en sistemas ópticos coherentes utilizando métodos supervisados y no supervisados.
- Monitoreo del rendimiento óptico que tiene como objetivo estimar los parámetros del sistema de transmisión a partir de un mapeo entre los parámetros del *lightpath* y las propiedades de la señal recibida utilizando métodos de aprendizaje supervisado.

En la capa de red, los casos de uso de aprendizaje automático se pueden dividir en 4 categorías (Musumeci y cols., 2019):

- Predicción de tráfico y tareas de rediseño de topología virtual para permitir el reruteo proactivo del tráfico y la configuración de la red utilizando algoritmos de aprendizaje supervisado y no supervisado.
- Gestión de fallos: esto incluye la detección de fallos, la localización de fallos y los métodos de causa raíz de fallos basados en aprendizaje supervisado para determinar las restauraciones apropiadas, la reconfiguración del tráfico o las intervenciones de campo.
- Clasificación de flujos de tráfico para una asignación eficiente de recursos y la gestión de prioridades de tráfico utilizando métodos de aprendizaje supervisado y no supervisado.
- Cálculo de ruta utilizando métodos supervisados y no supervisados alimentados con demanda de servicio e información de estado de la red para seleccionar rutas apropiadas y recursos de red disponibles que cumplan con la calidad de servicio (QoS) deseada sin afectar los servicios existentes provistos en la red.

A continuación se presentan algunos trabajos que aplican técnicas de aprendizaje automático en redes ópticas, en particular en la estimación de la calidad de transmisión de *lightpaths* no establecidos y en la predicción de la calidad de transmisión de *lightpaths* establecidos. Se analizan diversas soluciones de aprendizaje automático, incluyendo problemas que son atacados, técnicas propuestas y resultados obtenidos.

2.6.1. Artículo 1 - *Machine-learning method for quality of transmission prediction of unestablished lightpaths* (Rottondi y cols., 2018)

Un desafío central en las redes ópticas es el establecimiento de *lightpaths* o conexiones entre dos nodos de la red. Este proceso involucra una serie de

condiciones y etapas interrelacionadas, cada una de las que presenta su propio conjunto de problemas. Entre estos se incluyen el cálculo de rutas óptimas, la selección del formato de modulación y la tasa de información (en Gb/s) para proporcionar el servicio de manera eficiente, la estimación de la QoT esperada para la conexión, así como el monitoreo continuo de su performance, y la detección y predicción de errores y fallos, entre otros aspectos.

En (Rottondi y cols., 2018) se propone un enfoque de aprendizaje automático para la predicción de la calidad de transmisión (QoT) de un *lightpath*.

Al momento de establecer un *lightpath* existe un gran número de parámetros que pueden ser configurados, como el formato de modulación, el *baud rate*, el código de corrección de errores, la transmisión de un solo o múltiples *carriers*, el espaciado de canales, la tecnología *flex-grid*, entre otros. Esto hace que el número de combinaciones posibles para el despliegue de un *lightpath* crezca bastante y que la elección de los parámetros óptimos sea una tarea compleja. En este contexto, predecir la QoT de un *lightpath* antes de su despliegue es esencial para elegir la solución más efectiva y para un diseño y planificación optimizados de la red óptica. Hoy en día existen dos categorías de técnicas de estimación de QoT antes del despliegue. Por un lado, se pueden utilizar modelos analíticos sofisticados que capturan diferentes degradaciones de la capa física para estimar con gran precisión la tasa de error de bits y el alcance de un *lightpath* dado, pero imponen altos requisitos computacionales que no son compatibles con la predicción en tiempo real y no son escalables a topologías de red grandes y a operaciones de red dinámicas. Estos modelos analíticos se basan en la resolución de ecuaciones diferenciales parciales con el método de *Split-Step Fourier Method* por ejemplo (Shao, Liang, y Kumar, 2014). Dentro de esta familia de modelos analíticos existe el modelo de ruido Gaussiano (Poggiolini y cols., 2014) que reduce el tiempo de cómputo pero subestima capacidades de la red para compensar imprecisiones del modelo. Los modelos analíticos se basan en parámetros físicos para calcular la QoT de un *lightpath*. Dado que los parámetros físicos a menudo no se conocen con precisión, se necesitan márgenes de diseño adicionales para compensar las inexactitudes de los parámetros del modelo. Es por ello que se recurre a técnicas de aprendizaje automático para predecir la QoT de *lightpaths* no establecidos utilizando datos recogidos por monitores de red y explotando el conocimiento extraído de los datos de campo.

En el trabajo presentado en (Rottondi y cols., 2018) se proponen distintas técnicas para predecir si el *bit error rate* (BER) de un *lightpath* será menor o mayor que un umbral dado. En él, se trabaja sobre una red óptica con tecnología *flex-grid* con *slice width* de 12.5 GHz. Permite gestionar ancho de banda dinámicamente. Los transceptores, que son los dispositivos responsables de enviar y recibir señales ópticas, operan a 28 Gbaud, requiriendo un ancho de banda óptico de 37.5 GHz (equivalente a tres *slices*). Cuando la capacidad de un solo transceptor es insuficiente, se agrupan múltiples transceptores para formar súper canales, lo que permite a la red manejar demandas de tráfico más grandes. Las señales ópticas viajan a través de enlaces de fibra de larga distancia sin compensación de dispersión, lo que significa que la propagación natural de la señal de luz no se corrige en el camino. Para mantener la intensidad de la señal,

se colocan amplificadores ópticos cada 100 km a lo largo de la fibra, aumentando la señal mientras se agrega una pequeña cantidad de ruido. En el extremo receptor, el sistema utiliza detección coherente, que reconstruye con precisión la señal transmitida, seguida de procesamiento digital para corregir cualquier dispersión y otras distorsiones lineales. Finalmente, se realiza una verificación de errores para evaluar la calidad de la señal antes de aplicar técnicas de corrección de errores. Se utiliza un umbral de BER de 10^{-4} para determinar si un *lightpath* es aceptable o no. En el análisis presentado en (Rottondi y cols., 2018) se consideran las siguientes *features* para predecir la QoT de un *lightpath*:

- Número de enlaces del *lightpath*.
- Longitud total del *lightpath* (en km).
- Longitud del enlace más largo (en km).
- Volumen de tráfico que sirve (en Gb/s).
- Formato de modulación utilizado para la transmisión.
- *Guard-bands* izquierdo y derecho más pequeños que te separan del vecino más cercano.
- Volumen de tráfico y formato de modulación de los vecinos izquierdo y derecho más cercanos.

Se utilizan distintos algoritmos de clasificación para predecir la QoT de un *lightpath*. Entre ellos se encuentran, *Random Forest*, *K-nearest neighbors* (KNN) y un modelo base que predice la clase más frecuente. Para evaluar se utilizan las métricas de *AUC* y *accuracy*. Los resultados obtenidos se pueden ver en la Tabla 2.2.

Clasificador	*AUC*	*Accuracy*
Dummy Cl.	0.50	0.51
kNN k = 1	0.86	0.86
kNN k = 5	0.95	0.89
kNN k = 25	0.97	0.91
RF 1 est.	0.92	0.92
RF 5 est.	0.98	0.96
RF 25 est.	0.99	0.96
RF 100 est.	0.99	0.96
RF 500 est.	0.99	0.96

Tabla 2.2: Comparación de los distintos algoritmos de clasificación para predecir la QoT de un *lightpath*. Fuente: (Rottondi y cols., 2018)

2.6.2. Artículo 2 - *Quality of transmission estimation and short-term performance forecast of lightpaths* (Aladin y cols., 2020)

Otro trabajo que ataca el problema de la predicción de la QoT de un *lightpath* es (Aladin y cols., 2020). En él se trabajó en una red óptica con, de vuelta, tecnología *flex-grid* con *slice width* de 12.5 GHz. Se trabaja con ciertos parámetros fijos de la red como la tasa de atenuación de la fibra, el coeficiente de dispersión, los efectos no lineales y el ruido introducido por los amplificadores ópticos. Se consideran los siguientes parámetros para establecer un *lightpath* que constituyen las *features* para predecir la QoT:

- Longitud del enlace (rango de 80 a 7500 km).
- Largo del span (rango de 80, 100, 120, 150 km).
- Número de spans (rango de 1 a 50).
- Formato de modulación (PM-BPSK, PM-QPSK, PM-16QAM, PM-64QAM).
- Potencia del canal (rango de -10 a 5 dBm).
- Tasa de datos (40, 50, 100 Gb/s).
- *Polarización-Multiplexed (PM); Binary Phase-Shift.*

En este trabajo también se utilizan distintos modelos de aprendizaje automático para predecir si el BER de un *lightpath* será menor o mayor que un umbral dado. Se utilizó un proceso de *feature engineering* donde se utilizó un algoritmo SVM para clasificar el BER de un *lightpath* con cada *feature* individualmente. El resultado arrojó que la longitud del enlace es el parámetro más importante para predecir la QoT de un *lightpath* seguido por el número de spans, el formato de modulación, la potencia del canal y la tasa de datos. Se utilizaron dos algoritmos de aprendizaje automático, SVM y ANN. Para evaluar se utilizó la métrica de *accuracy*. La red neuronal utilizada fue una red con dos capas ocultas de 512 y 256 neuronas respectivamente. Los resultados obtenidos se pueden ver en la Tabla 2.3.

Algoritmo	*Accuracy*
ANN 6 *features*	99.56 %
SVM 6 *features*	99.38 %
ANN mejores 4 *features*	92.30 %
SVM mejores 4 *features*	93.30 %
ANN mejores 3 *features*	85.03 %
SVM mejores 3 *features*	88.52 %

Tabla 2.3: Comparación de los distintos algoritmos de clasificación para predecir la QoT de un *lightpath*. Fuente: (Aladin y cols., 2020)

Aparte de los modelos de aprendizaje automático, en (Aladin y cols., 2020) se propone un modelo *long short-term memory* LSTM específicamente diseñado para la predicción a corto plazo del *Signal-to-Noise Ratio* (SNR) en *lightpaths*. El modelo *LSTM* fue entrenado utilizando 13 meses de datos de monitoreo de rendimiento del mundo real recopilados en intervalos de 15 minutos de un enlace óptico de 1500 km dentro de la red CANARIE. El objetivo era pronosticar el SNR en diferentes ventanas temporales, que van desde 1 hora hasta 4 días. El estudio encontró que el modelo *LSTM* funcionó particularmente bien para ventanas de pronóstico más largas (24 a 96 horas), donde superó al método base de referencia (que asume que los valores futuros de SNR serán idénticos a los valores observados más recientes). El modelo *LSTM* demostró un valor R^2 positivo y un error cuadrático medio (RMSE) bajo, lo que indica su eficacia para predecir el SNR durante períodos prolongados. Aunque el modelo mostró las mejoras más significativas para pronósticos más largos, su rendimiento fue ligeramente menos preciso para ventanas más cortas (1 a 12 horas) en comparación con el modelo de referencia. Además del *LSTM*, el estudio también exploró otros modelos basados en RNN como el *Encoder-Decoder LSTM* y el *Gated Recurrent Unit* (GRU). Si bien el *LSTM* mostró el mejor rendimiento para ventanas de pronóstico más altas, el modelo GRU logró resultados ligeramente mejores para predicciones a muy corto plazo, gracias a su arquitectura más simple, que permite una computación más rápida.

2.6.3. Artículo 3 - *Machine-learning-based lightpath QoT estimation and forecasting* (Allogba y cols., 2022)

En (Allogba y cols., 2022) se presenta un enfoque basado en aprendizaje automático para la estimación y predicción de la calidad de transmisión (QoT) de *lightpaths*, abordando tanto la predicción de QoT para *lightpaths* no establecidos como el forecasting de la QoT en *lightpaths* ya desplegados. Este trabajo utiliza varias técnicas de aprendizaje supervisado, incluyendo KNN, *Support Vector Machine* (SVM), *Random Forest* (RF) y redes neuronales artificiales (ANN), para clasificar la QoT de un *lightpath* antes de su despliegue. Estos clasificadores son entrenados con datos sintéticos generados mediante modelos analíticos como el modelo de canal gaussiano (GN). En la Tabla 2.4 se pueden ver los resultados obtenidos. Los resultados demuestran que el modelo basado en SVM presenta la mejor *accuracy* entre los modelos más simples. Sin embargo, cuando se comparan modelos más complejos como las ANN, se observa que estos últimos pueden ofrecer un rendimiento ligeramente superior en términos de *accuracy* y tiempo de cómputo.

Algoritmo	*Accuracy*
KNN	81.8 %
SVM	99.4 %
RF	96.3 %
ANN	99.6 %

Tabla 2.4: Comparación de los distintos algoritmos de clasificación para predecir la QoT de un *lightpath*. Fuente: (Allogba y cols., 2022)

En segundo lugar, en el artículo se ataca el problema de la predicción de la QoT en *lightpaths* ya desplegados. Una vez que un *lightpath* está establecido, se monitorean diversas métricas de rendimiento, como la relación señal-ruido (SNR), la tasa de error de bits (BER) y la potencia óptica recibida (PRX) en el receptor para garantizar la calidad de la transmisión. Pero, debido a condiciones cambiantes (por ejemplo, degradación de la fibra, factores ambientales y deterioro del equipo), la QoT puede degradarse con el tiempo, lo que puede provocar interrupciones del servicio si no se gestiona de manera proactiva.

El objetivo en (Allogba y cols., 2022) es predecir la QoT futura en un horizonte de tiempo determinado, lo que permite a los operadores de red tomar acciones preventivas como el reruteo del tráfico o el ajuste de los parámetros de la red. Los modelos teóricos tradicionales fallan en capturar este dinamismo, especialmente para patrones complejos y a largo plazo (Allogba y cols., 2022). Por lo tanto, las técnicas de aprendizaje automático han ganado espacio para pronosticar la QoT aprovechando los datos históricos de rendimiento.

Las técnicas de aprendizaje automático utilizadas en (Allogba y cols., 2022) para predecir la QoT de *lightpaths* desplegados incluyen RNN, LSTM, *Encoder-Decoder LSTM*, GRU y MLP y un método base que asume que el rendimiento futuro será igual al último valor observado. Estos modelos se entrenan con datos históricos de rendimiento de *lightpaths* y se evalúan utilizando distintas métricas.

El desempeño de los distintos modelos de aprendizaje automático fue evaluado utilizando datos históricos de tres redes ópticas: CANARIE, North American Service Provider (NASP) y Microsoft (MS). Cada red proporcionó datos de series temporales como SNR, PRX y BER recopilados en intervalos de 15 minutos. Se utilizaron las métricas RMSE y AME para evaluar los modelos. Los resultados mostraron que los modelos LSTM multivariados tuvieron el mejor rendimiento para pronósticos a largo plazo, especialmente para horizontes de 96 horas, donde lograron el menor RMSE y AME. Por otro lado, los modelos univariados, incluidos LSTM y GRU, funcionaron adecuadamente para predicciones a corto plazo (horizonte de 1 hora), pero su ventaja disminuyó con el tiempo en comparación con el método base. También probaron modelos más simples como MLP y regresión lineal y, en algunos casos, superaron a los modelos RNN más complejos, especialmente cuando se entrenaron con conjuntos de datos más grandes. Esto muestra que si bien los modelos más complejos como RNN son poderosos, los modelos más simples y livianos pueden ser más efectivos dependiendo del tamaño del conjunto de datos y la presencia de valores atípicos. En la Tabla 2.5 se pueden ver los resultados obtenidos comparando LSTM

multivariados, LSTM univariados, GRU multivariados, GRU univariados.

Modelo	Horizonte de pronóstico	RMSE (dB)	AME (dB)
Univariate LSTM	1 hora	0.04	1.10
	24 horas	0.14	1.13
	96 horas	0.29	1.21
Univariate GRU	1 hora	0.04	1.10
	24 horas	0.14	1.13
	96 horas	0.27	0.92
Multivariate LSTM	1 hora	0.06	1.00
	24 horas	0.14	0.97
	96 horas	0.27	0.92
Multivariate GRU	1 hora	0.04	1.08
	24 horas	0.14	0.99
	96 horas	0.27	0.93

Tabla 2.5: Comparación de los distintos modelos de aprendizaje automático para predecir la QoT de un *lightpath*. Fuente: (Allogba y cols., 2022)

2.7. Otras herramientas y tecnologías utilizadas

En el desarrollo de este proyecto de grado, se utilizaron varias herramientas y tecnologías adicionales que facilitaron la gestión de datos, la visualización de procesos y la optimización de modelos. A continuación, se describen las más relevantes:

2.7.1. Alembic y SQLAlchemy para el almacenamiento de datos

Para manejar el almacenamiento y la migración de datos, se emplearon *SQLAlchemy* y *Alembic*. *SQLAlchemy* es una biblioteca de Python que proporciona un *Object Relational Mapping* (ORM), permitiendo interactuar con bases de datos relacionales de manera más intuitiva mediante objetos de Python (Bayer, 2012). Esta herramienta facilitó la abstracción de las consultas SQL, permitiendo centrar el enfoque lógica de alto nivel y en la manipulación de datos sin preocuparse por las particularidades del lenguaje SQL.

Por su parte, *Alembic* es una extensión de *SQLAlchemy* que permite gestionar y mantener las migraciones de esquemas de bases de datos de forma eficiente (Project, 2023). Esto es especialmente útil en proyectos donde el modelo de datos evoluciona con el tiempo, ya que asegura que todos los cambios en la estructura de la base de datos se apliquen de manera consistente y controlada.

En conjunto, estas herramientas fueron fundamentales para almacenar grandes volúmenes de datos generados durante el preprocesamiento y las etapas de entrenamiento de los modelos, garantizando integridad y facilitando futuras actualizaciones en el esquema de la base de datos.

2.7.2. Streamlit para la creación de interfaces de usuario

Para visualizar y monitorear las diferentes etapas del flujo de datos, se utilizó *Streamlit*, una biblioteca de Python que permite crear aplicaciones web interactivas de manera sencilla y rápida (Team, 2023). *Streamlit* está diseñado específicamente para desarrolladores y científicos de datos que buscan convertir sus programas en aplicaciones web sin necesidad de conocimientos profundos en desarrollo web.

En este proyecto, *Streamlit* se utilizó para construir una interfaz de usuario que permitió visualizar los datos y algunas etapas del proyecto de mejor manera. Esto incluyó la visualización de estadísticas descriptivas y gráficos generados durante el análisis exploratorio de datos.

2.7.3. Optuna para la optimización de hiperparámetros

La optimización de hiperparámetros es una etapa crucial en el desarrollo de modelos de aprendizaje automático, ya que los valores adecuados de estos parámetros pueden mejorar significativamente el rendimiento del modelo (Bergstra, Bardenet, Bengio, y Kégl, 2011). Para automatizar y hacer más eficiente este proceso, se empleó *Optuna*, una biblioteca de Python diseñada para la optimización automática de hiperparámetros mediante técnicas avanzadas como búsqueda bayesiana y estrategias adaptativas de muestreo (Akiba, Sano, Yanase, Ohta, y Koyama, 2019).

Optuna permite definir espacios de búsqueda para los hiperparámetros y proporciona una interfaz flexible para integrar la optimización en los flujos de trabajo existentes. En este proyecto, se utilizó para ajustar parámetros de los modelos, como las capas en las redes neuronales, las tasas de aprendizaje y el tiempo de entrenamiento. Esto se tradujo en modelos más precisos y generalizables, al encontrar configuraciones óptimas que hubieran sido difíciles de identificar manualmente.

2.7.4. Apache Kafka para la gestión de flujos de datos

En el ámbito de los sistemas distribuidos y el procesamiento de flujos de datos en tiempo real, *Apache Kafka* surgió como una de las tecnologías más robustas y ampliamente adoptadas. Kafka es una plataforma distribuida de mensajería que permite publicar, almacenar y procesar grandes volúmenes de datos de manera eficiente y confiable (Kreps, Narkhede, y Rao, 2011).

Apache Kafka se basa en una arquitectura distribuida que incluye tres componentes principales: productores, consumidores y corredores (*brokers*). Los productores envían mensajes a los *topics*, que actúan como canales o categorías de mensajes. Los corredores almacenan estos mensajes de forma persistente y garantizan su disponibilidad para los consumidores, quienes los procesan en tiempo real o bajo demanda. Una característica clave de Kafka es su capacidad para manejar flujos de datos en tiempo real, asegurando tolerancia a fallos y escalabilidad horizontal mediante la distribución de la carga entre múltiples nodos.

Una de las principales ventajas de Kafka es su capacidad para desacoplar los componentes del sistema, facilitando la modularidad y simplificando el mantenimiento. Además, su diseño permite almacenar datos de forma persistente durante un periodo configurable, actuando como un sistema híbrido de mensajería y almacenamiento. Esto resulta especialmente útil para escenarios en los que los consumidores no pueden procesar los datos en tiempo real y necesitan acceder a ellos posteriormente.

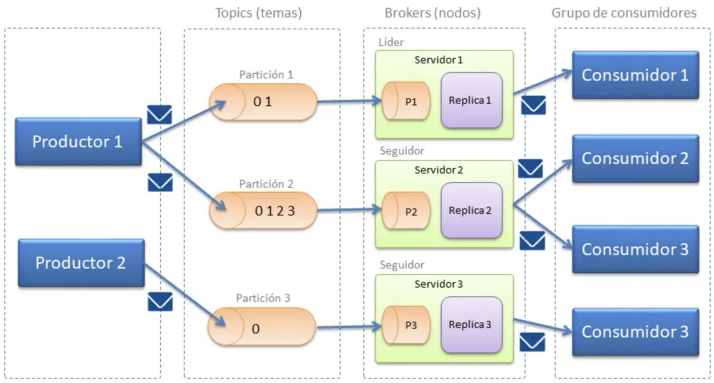

Figura 2.17: Arquitectura básica de Apache Kafka, mostrando productores, consumidores, corredores y *topics*. Fuente: (Calvo, 2024)

La Figura 2.17 ilustra la arquitectura básica de Kafka, destacando cómo los datos fluyen desde los productores hacia los consumidores a través de los *topics* gestionados por los corredores.

Capítulo 3

Caso de estudio: Antel

En este capítulo se describen las actividades realizadas en el marco de la colaboración con Antel. Los objetivos principales de este proyecto fueron:

- Diseñar un modelo de datos relacional para almacenar la información de la red de Antel.
- Implementar un *pipeline* de datos que permita consumir información de distintas fuentes.
- Visualizar la información de la red de Antel.
- Generar un conjunto de datos que sirva como entrada para modelos de aprendizaje automático.

El desarrollo del proyecto se organizó en tres fases principales:

- **Estudio de la estructura de datos y dominio de Antel:** Realizamos un análisis exhaustivo de la información disponible en la red de Antel, identificando los datos relevantes y las posibles fuentes de información. Este estudio lo hicimos con el objetivo de tener una base de conocimiento a partir de trabajos anteriores (Bianchi y Donnangelo, 2022; García, Pirri, Castro, y Grampín, 2023) y de información provista por Antel.
- **Diseño y creación de la base de datos y el *pipeline* de datos:** Diseñamos la base de datos relacional que albergaría la información de la red e implementamos un *pipeline* que permite la ingestión y procesamiento de datos desde diversas fuentes.
- **Implementación y validación de visualizaciones y generación de datos:** Desarrollamos herramientas de visualización que permitieron la interpretación de los datos almacenados.

Primero se presenta una introducción al dominio de Antel, luego se describen los materiales y herramientas utilizadas, se presentan los resultados obtenidos y finalmente se discuten los mismos.

3.1. Dominio de Antel

La red óptica de Antel utiliza una grilla fija de 50 GHz y un formato de modulación de QPSK o 16-QAM. Las solicitudes provienen de diferentes clientes según sus necesidades por lo que no siguen ninguna distribución o patrón específico. El bitrate solicitado es de 10 Gb/s en su mayoría aunque también existen solicitudes de 100 Gb/s, muy esporádicamente de 200 Gb/s y cuentan con algunas de 1 Gb/s aunque ya no se crean nuevas.

La red está compuesta de 120 nodos y 125 enlaces como se ve en la Figura 3.1 que muestra la distribución de los sitios y enlaces físicos ubicados en sus coordenadas relativas dentro del mapa de Uruguay. Cada nodo cuenta con una capacidad de switching (a nivel óptico) y de routing (a nivel OTN) diferente, determinada por diferentes elementos de hardware, como las IO/Cards, Uplinks y Muxponders instalados. Los nodos cuentan con dispositivos ROADM, es decir que son capaces de switchear lightpaths o, según la nomenclatura OTN de Antel, Optical CHannels (OCH) de forma pasiva y también saben que frecuencias deben ser transformadas a bits (transformación OEO) para que sean procesadas a nivel de capa de red en el mismo nodo. Es por esto que más adelante, características de un nodo como la capacidad y el uso (en Gb/s) se separarán en cantidad switcheada y enrutada.

Figura 3.1: Gráfica de nodos y segmentos de Antel

3.1.1. Componentes clave en la red OTN de Antel

La implementación de Optical Transport Networking (OTN) en la red de Antel depende de varios componentes esenciales:

- **I/O Cards:** Estas tarjetas de entrada y salida son responsables de recibir el tráfico desde las interfaces cliente, como Ethernet o IP/MPLS, y mapearlo a Optical Data Units (ODU). Una vez mapeados, los datos son enviados a la matriz de conmutación óptica (OTN matrix) dentro del nodo, donde se preparan para su transporte a través de la red.
- **Uplinks:** Los uplinks actúan como el punto de salida desde el nodo hacia otros nodos en la red. Estos reciben las Optical Transport Units (OTU) generadas en la matriz OTN y las transmiten a través de los canales ópticos. Por ejemplo, un uplink como el modelo 130SCUP es capaz de manejar OTU con diferentes capacidades, ofreciendo una transmisión eficiente y flexible.
- **Muxponders:** Los muxponders son dispositivos que combinan múltiples flujos de datos (ODU) en una única OTU, optimizando así el uso del ancho de banda disponible. En la red de Antel, el proceso completo de multiplexación dentro de un nodo OTN funciona de manera similar a un gran muxponder, con la ventaja de ofrecer mayor flexibilidad en la asignación de clientes a los uplinks.

3.2. Descripción de entidades del dominio

La Figura 3.1 muestra la topografía física de la red óptica core de Antel. La red core es la base del servicio de fibra óptica de Antel, de ella dependen servicios como la red de agregación y clientes específicos con grandes demandas de throughput (p.e., Datacenters).

Un servicio se define como la disponibilización a un cliente de conectividad entre dos nodos asegurando un bitrate predefinido.

En el caso de Antel, los OCH son vistos como recursos fijos en la red para transportar tráfico perteneciente a diferentes servicios. Un OCH puede transportar el tráfico de varios servicios y un servicio puede recorrer varios OCH hasta llegar al nodo destino. Un Optical Transport Segment (OTS), es el equivalente a un cable de fibra óptica que conecta dos nodos. Sobre los tramos OTS, se establecen OCH, un OCH recorre uno o más OTS, pasando por switches ópticos en nodos intermedios (dispositivos WSS). Así como un tramo de fibra contiene muchos canales y por ende muchos lightpaths, un OTS contiene varios OCH. En la Figura 3.2 se muestra un diagrama ejemplificando la interacción entre Servicios, OCH y OTS recién descrita. Se observa como algunos servicios recorren múltiples OCH y algunos OCH recorren múltiples OTS. Además, el diagrama muestra de forma esquemática lo que sería la composición interna de un nodo ROADM con switches pasivos (prismas) que seleccionan cuáles colores (OCH) dejar pasar y cuáles enviar a capas superiores.

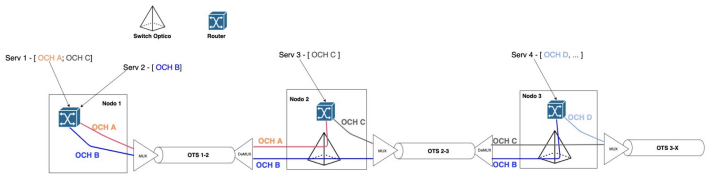

Figura 3.2: Diagrama de servicios, OCH y OTS de Antel

3.3. Características Operativas de Antel

El proceso de establecimiento de un nuevo servicio entre los nodos A y B consiste en observar manualmente si hay ancho de banda disponible en los OCH que conectan estos nodos. Si lo hay, se establece el servicio y el tráfico nuevo se agrega a la lambda (o color) de ese/os OCH que conectan A con B. Si no hay, se crean nuevos OCH o se reasigna el tráfico existente como mejor convenga (también de forma manual) para optimizar el uso de los OCH que suelen estar sub-utilizados. Además, algunos OCH cuentan con restauración, es decir que en caso de que alguno de los OTS que recorre ese OCH deje de funcionar, se cuenta con ancho de banda disponible de forma inmediata en otro camino para transportar esa frecuencia.

En el dominio de Antel existen los denominados ramales, tramos de la red que aún utilizan el medio eléctrico para transmitir información. En algunos casos, la red óptica no está totalmente conectada y la información viaja por el medio eléctrico entre dos sitios para luego volver a lo óptico.

3.4. OTN en la red de Antel

La Figura 3.3 permite entender como se implementa OTN dentro de un nodo. La placa IO/Card mostrada en gris, se encarga de recibir tráfico de las interfaces cliente (p.e., Ethernet), mapearlo a ODU y enviarlo a la matriz OTN (OCS NE o switch matrix en la figura). La matriz OTN se encarga de recibir las ODU, agregarlas si es necesario y entregarle al transponder/uplink una OTU (Uplik en figura, 130SCUP es un tipo de Uplink). En resumen, transforma tráfico IP/MPLS o Ethernet (ODU) a lightpaths. Todo este proceso tiene un funcionamiento similar al de un gran Muxponder (como en el nodo WDM NE en la figura), con la ventaja de tener más flexibilidad a la hora de asignar clientes a Uplinks.

En la Figura 3.4 se observa de forma más detallada el mapeo de Servicios a conjunto de ODU y a OTN que utiliza Antel en su operativa. Se listan en la primer capa los tipos de servicios disponibles a los clientes, luego estos se relacionan con las ODU descritas en la Tabla 2.1, estas ODU se mapea por medio de flechas multicolores a otras ODU, ya que múltiples ODU se pueden agregar en una sola para maximizar el uso de recursos. Finalmente se mapean las

ODU finales a OTU. Se observa por ejemplo como un servicio de 100GbE solo puede ser transportado en una ODU4 y por ende no se agrega en ninguna otra ODU y se termina transformando directo a una OTU4 u OTU4x2 (200Gb/s). A diferencia por ejemplo de servicios un servicio de tipo STM-16(3Gb/s) que se puede transportar en una ODU2 y agregarse en una ODU3 más tarde junto a otras ODU para finalmente ser transportado en una OTU3 (40Gb/s).

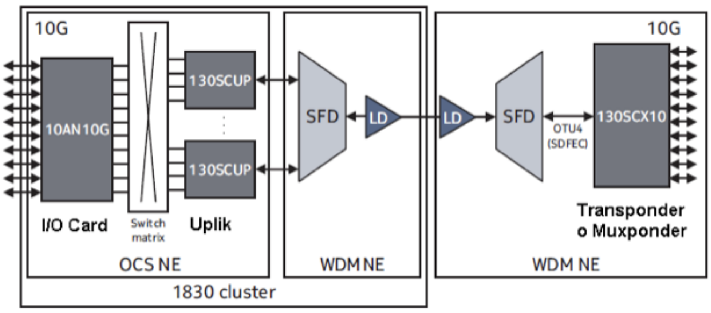

Figura 3.3: Implementación de OTN en un nodo de Antel

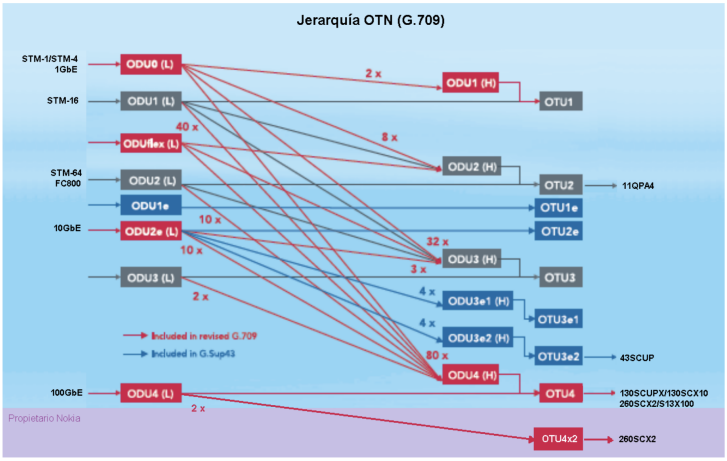

Figura 3.4: Relación entre los diferentes tipos de clientes (100GbE, 10GbE, STM-n), las ODU y luego las OTU utilizadas dentro de la red. El proceso de pasaje de las OTU a cada tipo de cliente en el nodo origen es análogo

3.5. Materiales

Para el trabajo se contó con dos fuentes de datos principales: el proyecto de grado previo y la base de datos de Antel. El proyecto de grado previo (Bianchi

y Donnangelo, 2022) sirvió como punto de partida para el trabajo realizado en este proyecto.

3.6. *Pipeline* de datos

El diagrama del *pipeline* de datos que pensamos desde un principio se puede ver en la Figura 3.5.

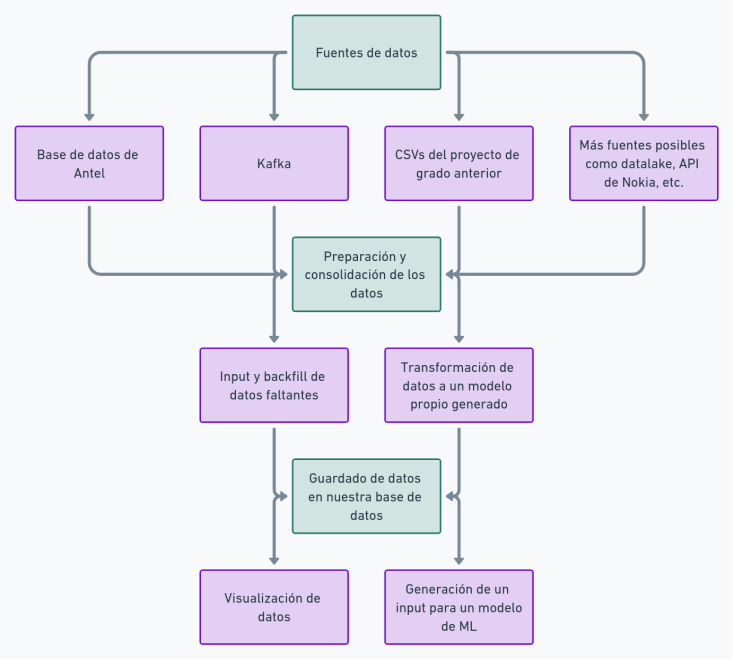

Figura 3.5: *Pipeline* de datos de Antel

Como se puede observar en la figura, la arquitectura del *pipeline* de datos se divide en cuatro partes:

- Consumo de datos de distintas fuentes.
- Transformación de los datos.
- Almacenamiento de los datos en un modelo de datos propio.
- Visualización de los datos.

3.6.1. Consumo y transformación de datos

Para el consumo de datos diagramamos una estrategia que permite consumir de distintas fuentes de datos con distintos propósitos. En un primer lugar, se

tuvieron en cuenta los datos estáticos que se iban a obtener de los archivos *csv* y de la base de datos de Antel. Estos datos tienen una naturaleza más estática ya que se refieren a datos topológicos de la red de Antel como los nodos, puertos, servicios, *lightpaths* y segmentos.

En un segundo lugar, se tuvieron en cuenta los datos más dinámicos que se iban a obtener a través de la API de Nokia y de Kafka.

Para el consumo de datos estáticos se creó un procedimiento que lee los archivos *csv* y la base de datos de Antel y los almacena en un modelo de datos propio. El procedimiento se ejecuta una sola vez y se encarga de poblar la base de datos con los datos estáticos.

A continuación, se describe el proceso de *backfilling* para cada una de las entidades principales. Aunque sean datos estáticos, se construyó un *pipeline* de datos para correrlo periódicamente y mantener actualizada la base de datos. Este *pipeline* en un primer lugar crea los registros que no existen en la base de datos y en un segundo lugar actualiza los registros si es que ya existen

Nodos El método *backfill_nodos* se encarga de normalizar y poblar la tabla de nodos en nuestra base de datos. Primero se obtienen los nodos existentes en la base de datos y se almacenan en la variable *existing_nodos*. Luego, se recorre el *DataFrame* de nodos y se compara con los nodos existentes. Si el nodo ya existe en la base de datos y hay cambios en los datos, se almacenan en la lista *to_update*. Si el nodo no existe en la base de datos, se almacenan en la lista *to_create*. Finalmente, se crean los nodos que no existen y se actualizan los nodos que han cambiado.

A su vez, el método utiliza la función *interpolate_relative_position* para obtener las coordenadas de los nodos a partir de un archivo *csv* auxiliar llamado *site_positions.csv* proveído por el proyecto de grado previo.

Puertos El método *backfill_puertos* se encarga de normalizar y poblar la tabla de puertos en nuestra base de datos. Se sigue un proceso similar al de los nodos para crear y actualizar los puertos en la base de datos.

Se creó una correspondencia para unir los distintos tipos de tasa a un valor numérico que se pueda utilizar en los modelos de aprendizaje automático y en la visualización de los datos. El mapeo se puede ver en el siguiente código:

```
RATE_MAP = rate_map = {
    "FC800": 0.8,
    "FC1600": 1.6,
    "STM-1": getSTMRate(1),
    "STM-16": getSTMRate(16),
    "STM-4": getSTMRate(4),
    "1GbE": 1,
    "10GbE": 10,
    "100GbE": 100,
    "STM-64": getSTMRate(64),
    "OTU2": 10,
```

```
    "OTU2E": 10,
    "OTU4": 100,
    "None": None,
    "OTU3E2": 44.58,
    "OCH": None,
    "OTU4x2": 200,
}

def getSTMRate(n):
    return 155.52 / 1024 * n
```

Segmentos Para el *backfilling* de los segmentos creamos la función *backfill_segmentos* que se encarga de poblar la tabla de segmentos en nuestra base de datos sin ninguna complejidad adicional. A su vez, y en base a lo hecho por trabajos anteriores, se creó la función *backfill_segmentos_series_temporales* para agregar datos temporales asociados a los segmentos. Este método se encarga de leer los archivos *csv* con los datos temporales que van desde el 2021-02-18 al 2021-04-22 y contiene información sobre la pérdida promedio, la potencia de entrada, la potencia de salida y la tasa de uso de los segmentos. Estos datos se almacenan en la tabla *segmento_series_temporales* de nuestra base de datos.

Lightpaths La información de los *lightpaths* se obtiene de la tabla *dwdm_och*. La función *backfill_lightpaths* normaliza y almacena estos datos en nuestra base de datos.

Servicios Para el *backfilling* de los servicios se creó el método *backfill_servicios*. Como particular de este método, se creó una función *parse_rate_contratado* que se encarga de mapear los nombres de los servicios para obtener el rate contratado.

Por ejemplo:

- "MLG-SBT 10000 RARW 02"→ 10
- "PAP-POD 10K GBDW 7"→ 10
- ÇOR-PA5 100000 CORW 02"→ 100

Para ello se utilizó este esquema de como descomponer los nombres de los servicios y obtener la tasa contratada que se puede ver en la Figura 3.6.

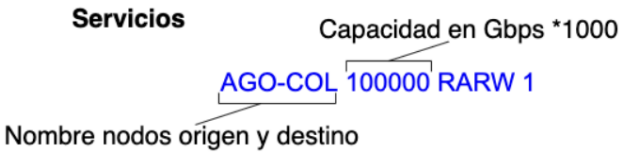

Figura 3.6: Descomposición de los nombres de los servicios de Antel

Tablas de relaciones Por último, se crearon las tablas de relaciones entre las entidades principales.

- Relaciones *Lightpath*-Segmento: La tabla *dwdm_opticalxc* contiene las relaciones entre *lightpaths* y segmentos. La función *backfill_lightpath_segmentos* se utilizó para insertar estas relaciones en la base de datos.
- Relaciones Servicio-Lightpath: Las relaciones entre servicios y *lightpaths* se obtienen de la tabla *dwdm_electricxc*. La función *backfill_servicio_lightpaths* inserta estos datos en la base de datos.

Consumo de datos dinámicos No pudimos implementar el consumo de datos dinámico Sin embargo diseñamos una estrategia que permita consumir información. Se dejaron variables de entorno para poder configurar la conexión a la API de Nokia y a Kafka. Decidimos intervalos de tiempo para cada fuente distinta de datos y en base a eso ejecutar trabajos que consuman la información de las fuentes.

Para el consumo de datos de Kafka se hubiera utilizado el trabajo realizado en (García y cols., 2023) como punto de partida.

3.6.2. Creación de un modelo de datos

A partir de los datos obtenidos creamos un modelo de datos que permitió almacenar la información de la red de Antel. El modelo de datos trató de abstraerse de la estructura de los datos originales y proveer un modelo que sirva para futuras iteraciones de este proyecto. Cuando se empezó a idear el modelo de datos, se tuvo en cuenta la posibilidad de que a futuro se consuma en tiempo real la información de la red de Antel a través de varias fuentes.

Las entidades que se definieron en el modelo de datos son las siguientes (ver Figura 3.7):

- Nodos: representa los nodos de la red de Antel.
- Puertos: representa los puertos de los nodos de la red de Antel.
- Servicio: representa los servicios de la red de Antel.
- Lightpath: representa los lightpaths de la red de Antel.

- Segmento: representa los segmentos de la red de Antel.
- SegmentoSerieTemporal: representa datos temporales asociados a los segmentos.
- Lightpath_Segmento: representa la relación entre los lightpaths y los segmentos.
- Servicio_Lightpath: representa la relación entre los servicios y los lightpaths.

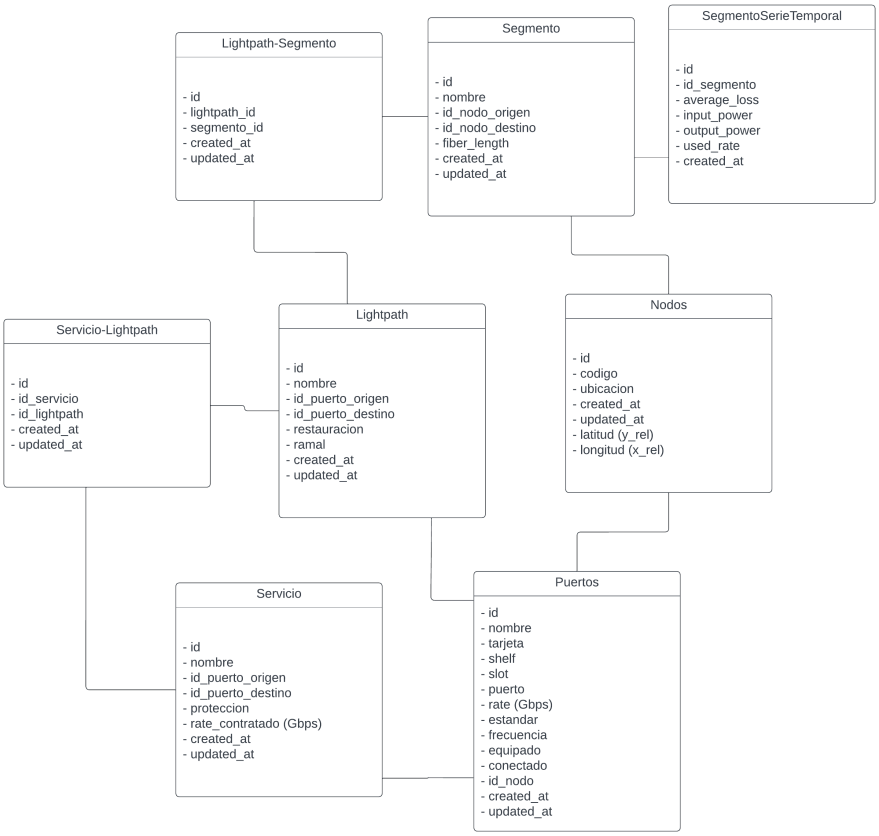

Figura 3.7: Diagrama de entidades de Antel

La base de datos creada se llama *antel_database* y se utilizó en PostgreSQL. En un principio se alojó en un servidor local. Como manejador de base de datos se utilizó *alembic* para manejar las migraciones de la base de datos y *SQLAlchemy* para interactuar con la base de datos.

3.6.3. Visualización y análisis de datos

Desarrollamos una aplicación web interactiva para visualizar los diversos resultados del proyecto, incluyendo tanto el análisis de datos como el *pipeline* de datos mencionado anteriormente. Esta aplicación fue construida utilizando Streamlit.

La aplicación tiene dos secciones principales: una sección donde se puede ejecutar el *pipeline* de datos manualmente (como también borrar los datos de la base de datos) y otra donde se puede hacer un análisis exploratorio de los datos y visualizar los resultados obtenidos. La interfaz gráfica de la aplicación se puede ver en la Figura 3.8.

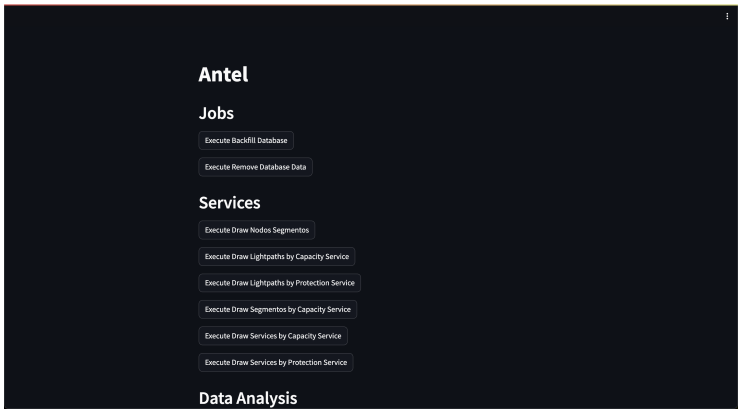

Figura 3.8: Frontend de la aplicación de Antel

A continuación se describen todas las funcionalidades de la aplicación:

***Pipeline* de datos** En la sección de *pipeline* de datos se pueden ejecutar los procedimientos de *backfilling* manualmente y borrar los datos de la base de datos.

Gráficas donde se puede visualizar la información topológica de la red de Antel En la sección de análisis exploratorio de datos se pueden visualizar distintas gráficas que muestran información de la red de Antel. La primera gráfica observada en la figura 3.1 muestra un mapa con los nodos y segmentos de la red de Antel. La segunda gráfica observada en la figura 3.9 muestra un grafo con los*lightpaths* de la red de Antel donde se puede observar la capacidad en Gbps de cada lightpath. También se generaron otras gráficas que muestran más información como los *lightpaths* con el tipo de protección, los servicios con la tasa de bits contratado, entre otros.

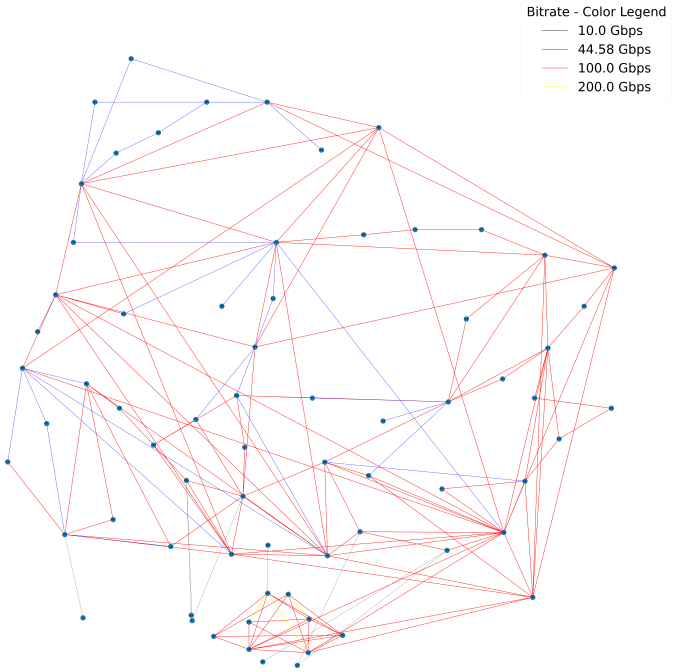

Figura 3.9: Gráfica de *lightpaths* de Antel con la capacidad en Gbps

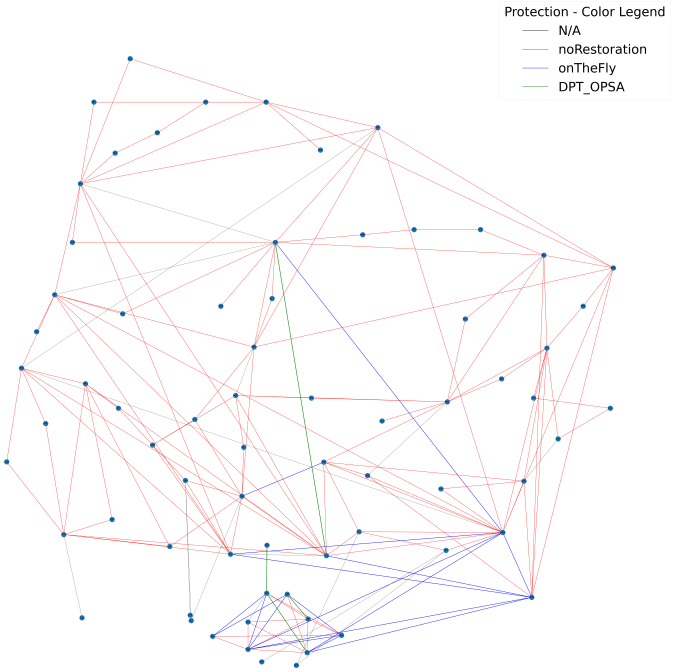

Figura 3.10: Gráfica de *lightpaths* de Antel con el tipo de protección

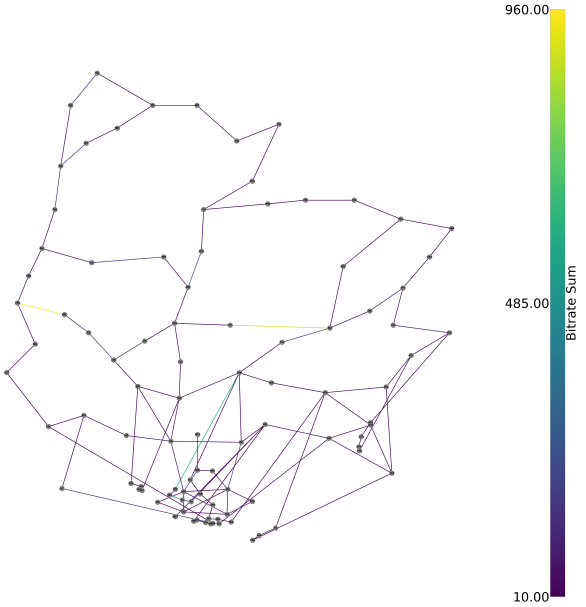

Figura 3.11: Gráfica de segmentos de Antel con la saturación de Gbps consumida por los servicios

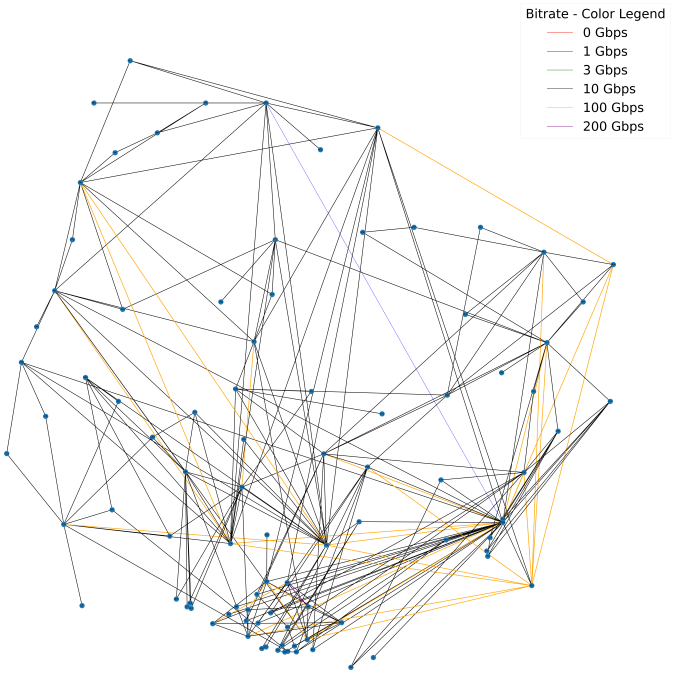

Figura 3.12: Gráfica de servicios de Antel con la tasa de bits contratada

Análisis exploratorio de datos En la sección de análisis exploratorio de datos se pueden visualizar distintas datos que permiten analizar cada entidad del modelo de datos generado. Como ejemplo de lo que muestra la aplicación, se puede ver en la figura 3.13 un análisis exploratorio de los servicios de Antel.

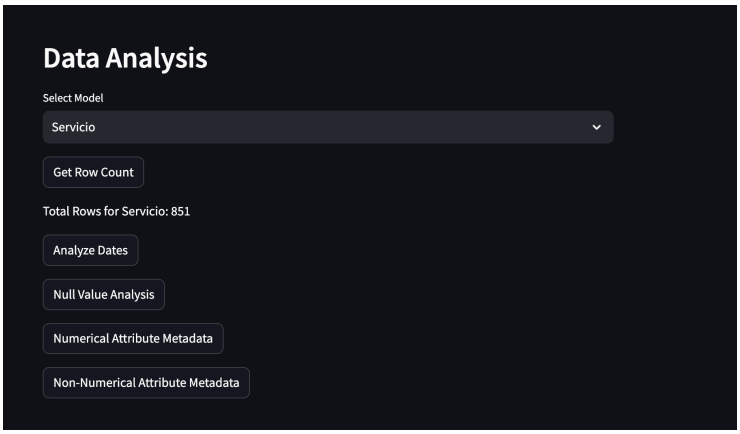

Figura 3.13: Análisis exploratorio de los servicios de Antel

La aplicación te permite las siguientes funcionalidades:

- Obtener la cantidad de registros de esa entidad en la base de datos.
- Visualizar cuando se agregó ese registro a la base de datos como se puede ver en la figura 3.14.
- Ver que valores nulos tiene esa entidad para que campos.
- Análisis numérico de los campos de la entidad como la media, mediana, desviación estándar, mínimo y máximo.
- Análisis de los campos de la entidad.

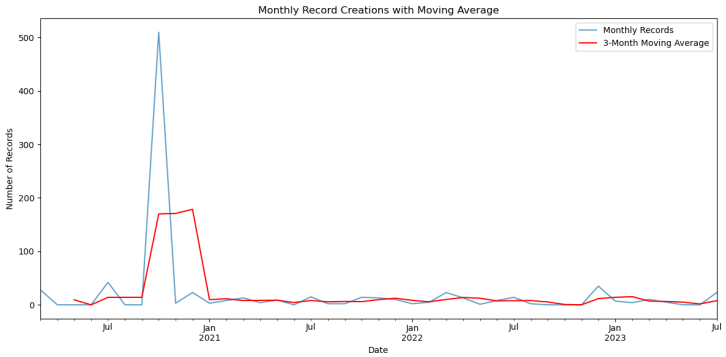

Figura 3.14: Fecha de creación de los servicios de Antel

3.7. Herramientas utilizadas

Para esta etapa del proyecto se utilizó una MacBook Pro Apple M1 Pro con 16 GB de memoria RAM. El código fuente de todo el trabajo realizado se encuentra en (Facultad de Ingeniería - MINA, s.f.-a).

3.8. Conclusiones

En este capítulo se describieron las actividades realizadas en el marco de la colaboración con Antel. Presentamos los objetivos principales del proyecto, describimos el dominio de Antel y presentamos los materiales y herramientas utilizadas para los objetivos propuestos. Describimos el *pipeline* de datos que se implementó para consumir información de distintas fuentes y presentamos el modelo de datos que se creó para almacenar la información de la red de Antel. Finalmente, presentamos la aplicación web interactiva que desarrollamos para visualizar los resultados obtenidos y se discutieron los mismos.

Debido a la falta de acceso tanto a la API de Nokia como a Kafka, no pudimos implementar el *pipeline* de datos para consumir información dinámica de la red de Antel. Sin embargo, logramos implementar un *pipeline* de datos que sirve como punto de partida para futuras iteraciones de este proyecto.

Creamos un modelo de datos que permite almacenar la información de la red de Antel y se desarrolló una aplicación web interactiva para visualizar los resultados obtenidos. La aplicación permite visualizar la información topológica de la red de Antel y realizar un análisis exploratorio de los datos.

Capítulo 4

Caso de estudio: HHI - Estimación de calidad de transmisión

Este capítulo describe el proceso de resolución de la estimación del calidad de transmisión para un conjunto de datos proveído por la Fraunhofer Heinrich Hertz Institute. Se comienza por la descripción del conjunto de datos y del dominio sobre el que se trabajó. Luego se explican las metodologías para la resolución del problema y para finalizar se hace un análisis de las pruebas y los resultados obtenidos.

Con el objetivo de analizar como responden distintos modelos y metodologías a la hora de estimar la calidad de transmisión (QoT) y con la base comparativa de los resultados obtenidos por HHI (Bergk, Shariati, Safari, y Fischer, 2022; Safari, Shariati, Bergk, y Fischer, 2021) fue que empezamos a procesar la información como muestra la Figura 4.1:

- En primer lugar empezamos con la obtención y análisis exploratorio de los datos disponibles. Esta etapa ayudó a familiarizarnos con la estructura y la cantidad de los datos.

- En segundo lugar y también de manera exploratoria fuimos probando metodologías de aprendizaje automático para la resolución del problema. Esta etapa resultó en métodos fiables y comparables con los resultados obtenidos por HHI.

- Por último lugar analizamos los resultados y las metodologías propuestas y el por qué de los resultados. Siempre el proceso que aplicamos fue iterativo y luego de evaluar los resultados se volvió a aplicar metodologías hasta que obtuvimos resultados satisfactorios. En toda etapa nos centramos en entender las razones y la variabilidad de los resultados en base a la entrada

de los métodos aplicados. Siempre favorecimos a los métodos que proveen interpretabilidad con este propósito.

El flujo de trabajo se describe en la Figura 4.1.

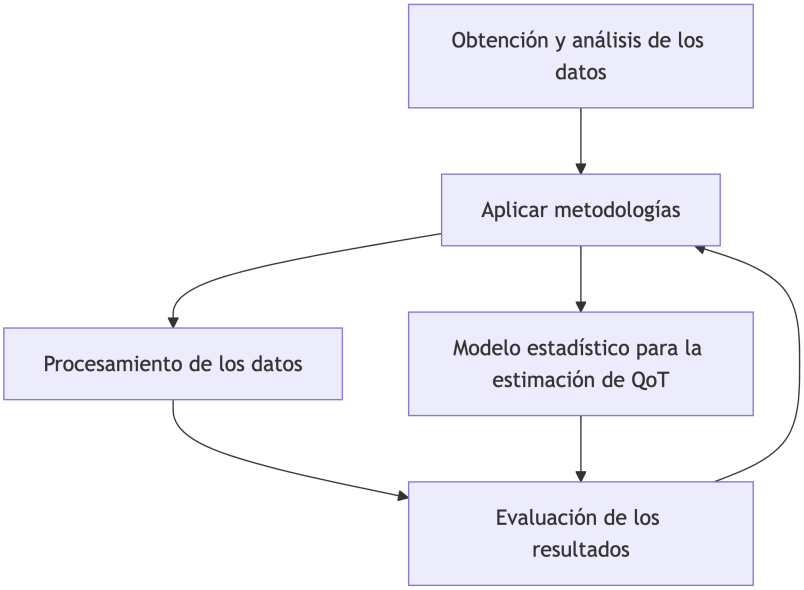

Figura 4.1: Diagrama de trabajo

4.1. Materiales

Como se explica en (Bergk y cols., 2022), la estimación de la QoT basada en métodos de aprendizaje automático ha crecido en los últimos años. La poca disponibilidad de grandes conjuntos de datos y la poca homogeneidad de los datos hacen que los resultados que se obtienen sobre estos sean difíciles de comparar y reproducir.

Gracias a la colección de datos publicados por Fraunhofer HHI (Bergk y cols., 2022) se puede tener un mismo punto de partida para los distintos métodos de estimación de calidad de transmisión propuestos. Esto facilita la tarea de comparar resultados y metodologías presentadas.

Se cuenta con una colección de conjuntos de datos generados en base a simulaciones y creados a partir de dos topologías distintas de red. Las topologías son Continental Core Optical Network of the United States (CORONET) (Lehmen, Doverspike, y Clapp, 2015) y TSNN (Shariati, Zervas, Nejabati, y Simeonidou, 2017).

Este problema nos permitió un acercamiento con un conjunto de datos establecido y el análisis y entendimiento de metodologías de aprendizaje automático

propuestas por otros grupos de estudio. A su vez, nos permitió analizar la estimación de la calidad de transmisión basado en tres métricas (OSNR, SRN y BER) y la sensibilidad de estas ante las distintas variables inherentes a una red de fibra óptica.

4.1.1. Descripción de los datos

La colección de datos calidad de transmisión publicada por Fraunhofer HHI incluye cuatro conjuntos de datos (Fraunhofer HHI, 2024). Estos conjuntos de datos fueron creados con el objetivo de construir modelos de aprendizaje automático para la estimación del calidad de transmisión en una EON incluyendo problemas de regresión y clasificación.

Debido a la extensión de los conjuntos de datos y las características de los mismos decidimos trabajar con el primer conjunto de datos. Como se puede ver en las Tablas 4.4, 4.11, 4.12 y 4.13 el primer conjunto de datos es el más grande y junto al tercero el más balanceado. El balanceo de datos hizo que nos decantaramos por el primer y tercer conjunto de datos. El tercer conjunto de datos presenta una topología diferente al resto y el análisis hecho por HHI muestra que el primer conjunto de datos es más rico en características topológicas: "Los conjuntos de datos 01 y 03 comparten un escenario de red idéntico, pero representan dos topologías de red diferentes. La red TSNN considerada en el conjunto de datos 03 es mucho más pequeña y menos diversa en longitudes de *lightapth* que la CONUS considerada en el conjunto de datos 01. La CONUS incluye *lightapths* de longitudes similares a los presentes en la TSNN, pero esta última no puede representar la diversidad de *lightapths* de la CONUS. En consecuencia, las precisiones de los modelos entrenados en el conjunto de datos 03 son inferiores al 60 % en cada uno de los otros conjuntos al considerar la topología de la CONUS"(traducción propia, (Bergk y cols., 2022, p. 52)). A su vez, los resultados obtenidos por HHI al aplicar sus metodologías hizo que solo consideraramos al primer conjunto de datos para el análisis: "Según los valores de precisión presentados, los modelos entrenados en el conjunto de datos 01 logran precisiones más altas en conjuntos de datos externos que cualquiera de los otros cuatro conjuntos. Por lo tanto, el conjunto de datos 01 resulta ser el más completo, versátil y rico en datos"(traducción propia, (Bergk y cols., 2022, p. 52)). Describiremos únicamente el primer conjunto de datos en este informe aunque los otros conjuntos de datos siguen una estructura similar y se pueden encontrar en (Fraunhofer HHI, 2024).

Las muestras de los conjunto de datos fueron generadas utilizando PLATON (Bergk y cols., 2022), una herramienta de planificación para redes ópticas desarrollada por Fraunhofer HHI. Cada uno de los 4 conjuntos de datos incluye dos representaciones de datos diferentes: una representación unidimensional que es un conjunto de datos basado en *lightpaths* y una representación multidimensional que es un conjunto de datos basado en el estado de la red.

El conjunto de datos basado en el estado de la red describe el estado de la red y contiene las características de una red funcionando al momento que se quiere aprovisionar un nuevo *lightpath* llamado LUT (*lightpath under test*). Por

otro lado, el conjunto de datos basado en *lightpaths* contiene las características de todos los LUT al momento de ser establecidos junto a algunas características descriptivas de la red. Ambas representaciones de datos se obtienen de instancias idénticas de la red.

Todas las muestras están etiquetadas con las siguientes métricas:

- OSNR
- SNR
- BER
- una clase que indica si el BER de la muestra está por encima (0: clase negativa) o por debajo (1: clase positiva) de un cierto umbral

Las muestras de ambas representaciones de datos, es decir, el conjunto de datos basado en el estado de la red y el conjunto de datos basado en *lightpaths*, se complementan con la matriz objetivo $Y \in \mathbb{R}^{D \times T}$, donde D es el número de muestras y T es el la dimensión del vector objetivo $y_t \in \mathbb{R}^D$. En consecuencia, el elemento d-ésimo de cada vector objetivo y_t corresponde al d-ésimo punto de datos. Cada uno de estos elementos es una variable objetivo $y_t(d)$, o simplemente y_t. La Tabla 4.1 lista las variables objetivo y_t con un nombre descriptivo y el dominio de definición.

El OSNR contenido en el vector objetivo se calcula como la suma de OSNR calculado por cada link que pasa el LUT. El OSNR en cada link k se calcula como

$$OSNR_{db} = log_{10}\left(\left(\sum_{k \in K_d} \frac{1}{OSNR_k}\right)^{-1}\right),$$

donde $K_d \subset \{k \in \mathbb{N} \mid 0 \leq k < K\}$ es el conjunto de links por los que pasa el LUT. A su vez, el SNR se calcula a partir del OSNR y el BER se calcula a partir del SNR y la cardinalidad del formato de modulación. Para más información de como se calculan estas métricas consultar (Fraunhofer HHI, 2024).

y_n	**Unidad**	**Nombre**	**Dominio de definición**
y_1	1	Class	$\{0, 1\}$
y_2	dB	OSNR	$x \in \mathbb{R} \mid 12.47 \leq x \leq 33.49$
y_3	dB	SNR	$x \in \mathbb{R} \mid 8.96 \leq x \leq 29.98$
y_4	1	BER	$x \in \mathbb{R} \mid 1.70 \cdot 10^{-12} \leq x \leq 1.98 \cdot 10^{-2}$

Tabla 4.1: Descripción de las variables objetivo del conjunto de datos

En resumen, cada muestra de cada conjunto de datos está asociada al mismo vector de variables objetivo y_t. Por lo tanto, tomando como ejemplo la primera muestra de ambas representaciones de datos se puede observar lo siguiente.

El conjunto de datos basado en *lightpaths*, nos da las características completas del primer LUT que se aprovisiona en la red y sus métricas de error asociado. Por otro lado, el conjunto de datos basado en el estado de la red nos da las características de la red al momento de la configuración del primer LUT y las mismas métricas de error asociadas al LUT en cuestión. Cabe destacar, que en el conjunto de datos utilizado se omitieron las primeras 1000 muestras porque correspondían a una red con alta disponibilidad de recursos y no era representativa de la realidad. Por lo tanto cuando se refiere al primer LUT, quiere decir el primer LUT que se quiere aprovisionar en una red con 1000 conexiones ya establecidas y andando. Entonces, ambas representaciones de datos fueron obtenidas de la misma instancia de la red y están asociadas a las mismas métricas de error solo que las características nos dan distinta información.

Información sobre la configuración de la red Los conjuntos de datos fueron generados a partir de ocho simulaciones de una EON utilizando PLATON. PLATON es un simulador de eventos discretos que simula una red óptica elástica basado en un conjunto de algoritmos de enrutamiento y asignación de espectro. La herramienta registra numerosas estadísticas, incluyendo métricas de calidad de transmisión, de cada solicitud de conexión establecida y rechazada en su base de datos de ingeniería de tráfico. Las métricas de calidad de transmisión se calculan utilizando un modelo de canal no lineal que calcula analíticamente el OSNR, SNR y BER.

PLATON tiene un set de configuraciones que permiten habilitar o deshabilitar ciertas características y modificar los parámetros de simulación. Para información más detallada sobre las configuraciones y características de PLATON, se puede consultar (Bergk y cols., 2022).

Como se mencionó anteriormente, tres de los cuatro conjuntos de datos, incluido el primero que se utiliza en este trabajo, fueron generados en base a la topología de la red CORONET (Figura 4.2) extraída de (Monarch Network Architects, 2020) y se describen en la Tabla 4.2.

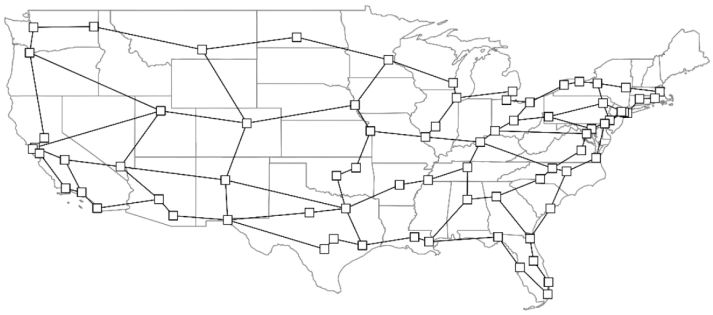

Figura 4.2: Topología de la red CORONET (Monarch Network Architects, 2020)

Parámetro	Valor
Número de nodos	75
Número de enlaces	99
Número de pares origen-destino	5550
Longitud promedio de enlace	395.81 km
Longitud máxima de enlace	1221.19 km
Longitud mínima de enlace	24.21 km
Longitud del tramo	80.0 km
Número mínimo de EDFAs por enlace	0
Número máximo de EDFAs por enlace	15
Grado promedio de los nodos	2.64
Grado máximo de los nodos	5
Grado mínimo de los nodos	2
Diámetro de la red	17
Diámetro de la red en km	6472.18 km

Tabla 4.2: Datos de la topología de la red CORONET

A continuación se describen las características del primer conjunto de datos que es el que se utilizó para el caso de estudio.

De las ocho simulaciones, cada una incluye 100,000 solicitudes de conexión. La red se simula como una red EON de grilla de frecuencia fija y tasa de datos flexible operando en la banda C. La tasa de datos flexible permite que las longitudes de onda establecidas transporten distintas tasas de datos a lo largo de los *lightpaths*. Se utilizó una longitud de tramo fija de 80 km para el cálculo de las métricas de calidad de transmisión. Los *lightpaths* utilizan los formatos de modulación PM-QPSK, PM-8QAM, PM-16QAM, PM-32QAM y PM-64QAM. La grilla de frecuencia es fija con un espaciado de canal de 37.5 GHz y 96 canales disponibles en la banda C. Todos los *lightpaths* transmiten con una tasa de símbolos de 28 GBd y una potencia de lanzamiento de -3dBm.

La Tabla 4.3 muestra las características del primer conjunto de datos.

Símbolo	Valor	Unidad	Descripción
z_s	80	km	Longitud de tramo
NF	5	dB	Figura de ruido de EDFA
G_{dB}	16	dB	Ganancia del amplificador
α_{dB}	0.2	dB/km	Atenuación de potencia
D_0	17	ps/nm/km	Dispersión de fibra
γ	1.3	1/W/km	Coeficiente de no linealidad
B_N	12.5	GHz	Ancho de banda de ruido
λ_{ref}	1550	nm	Longitud de onda de referencia

Tabla 4.3: Características del primer conjunto de datos

4.1.2. Variables objetivo

A continuación se pueden ver las gráficas de la distribución y la variación de los valores objetivo (*targets*).

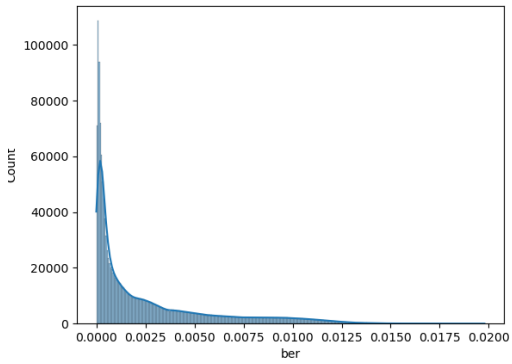

Figura 4.3: Distribución de BER en el primer conjunto de datos

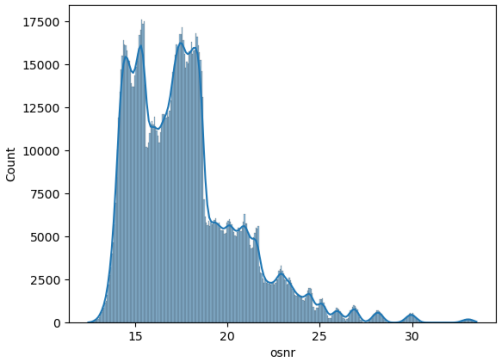

Figura 4.4: Distribución de OSNR en el primer conjunto de datos

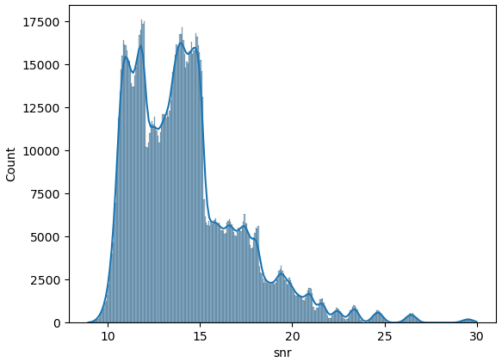

Figura 4.5: Distribución de SNR en el primer conjunto de datos

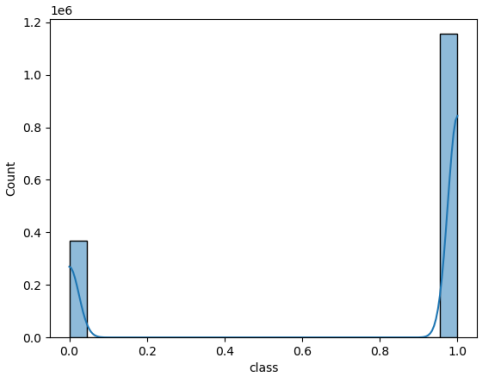

Figura 4.6: Distribución de la clase en el primer conjunto de datos basado en el estado de la red

4.1.3. Conjunto de datos basado en *lightpaths*

Primer conjunto de datos El primer conjunto de datos tiene 1,524,755 filas y se puede observar su tamaño y balance de clases en la Tabla 4.4.

Muestras	**Total**		**Proporción**		**Tamaño del conjunto de datos**	
	Positiva	**Negativa**	**Pos.**	**Neg.**	**Lightpaths**	**Estado de red**
Total	1,155,635	369,120	75.79 %	24.21 %	465.3 MB	1.6 TB

Tabla 4.4: Tamaño y balance de clases del conjunto de datos

Los cuatro conjuntos de datos basados en *lightpaths* presentan el mismo

formato con las *features* asociadas al vector objetivo previamente explicado. Los metadatos del primer conjunto de datos se pueden observar en la Tabla 4.5.

x_n	**Unidad**	**Nombre**	**Dominio de definición**
x_1	1	ID de conexión	$x \in \mathbb{N} \mid 1001 < x < 100,000$
x_2	1	ID del nodo fuente	$x \in \mathbb{N} \mid 1 \leq x \leq 75$
x_3	1	ID del nodo destino	$x \in \mathbb{N} \mid 1 \leq x \leq 75$
x_4	km	Longitud del *lightpath*	$x \in \mathbb{R} \mid 24,214 \leq x \leq 7,834,746$
x_5	km	Longitud promedio del enlace	$x \in \mathbb{R} \mid 24,214 \leq x \leq 1,221,189$
x_6	km	Longitud del enlace más corto	$x \in \mathbb{R} \mid 24,214 \leq x \leq 1,221,189$
x_7	km	Longitud del enlace más largo	$x \in \mathbb{R} \mid 24,214 \leq x \leq 1,221,189$
x_8	1	Número de enlaces	$x \in \mathbb{N} \mid 1 \leq x \leq 25$
x_9	1	Número de tramos	$x \in \mathbb{N} \mid 1 \leq x \leq 106$
x_{10}	THz	Frecuencia	$x \in \mathbb{R} \mid x = 192.2 + 0.0375n, n \in \mathbb{N} \mid n \leq 96$
x_{11}	1	Orden de modulación	$\{4, 8, 16, 32, 64\}$
x_{12}	Gb/s	Tasa de línea del *lightpath*	$\{56, 112, 168, 224, 280, 336\}$
x_{13}	Gb/s	Tasa de línea de la conexión	$\{112, 224, 448\}$
x_{14}	1	Grado del nodo fuente	$x \in \mathbb{N} \mid 2 \leq x \leq 5$
x_{15}	1	Grado del nodo destino	$x \in \mathbb{N} \mid 2 \leq x \leq 5$
x_{16}	1	Ocupación del enlace más corto	$x \in \mathbb{N} \mid 1 \leq x \leq 93$
x_{17}	1	Ocupación del enlace más largo	$x \in \mathbb{N} \mid 1 \leq x \leq 95$
x_{18}	1	Ocupación promedio del enlace	$x \in \mathbb{N} \mid 1 \leq x \leq 93$
x_{19}	1	Desviación estándar de la ocupación del enlace	$x \in \mathbb{R} \mid 0 \leq x \leq 39,064$
x_{20}	1	Suma de la ocupación del enlace	$x \in \mathbb{N} \mid 1 \leq x \leq 1155$
x_{21}	1	BER máximo de los *lightpaths* interferentes	$x \in \mathbb{R} \mid 0 \leq x \leq 3,8 \cdot 10^{-3}$
x_{22}	1	BER mínimo de los *lightpaths* interferentes	$x \in \mathbb{R} \mid 0 \leq x \leq 3,5 \cdot 10^{-3}$
x_{23}	1	BER promedio de los *lightpaths* interferentes	$x \in \mathbb{R} \mid 0 \leq x \leq 3,5 \cdot 10^{-3}$
x_{24}	1	Cardinalidad mínima del formato de modulación (izquierda)	$\{0, 4, 8, 16, 32, 64\}$
x_{25}	1	Cardinalidad máxima del formato de modulación (izquierda)	$\{0, 4, 8, 16, 32, 64\}$
x_{26}	1	Cardinalidad mínima del formato de modulación (derecha)	$\{0, 4, 8, 16, 32, 64\}$
x_{27}	1	Cardinalidad máxima del formato de modulación (derecha)	$\{0, 4, 8, 16, 32, 64\}$
x_{28}	Gb/s	Tasa de línea mínima del *lightpath* (izquierda)	$\{0, 56, 112, 168, 224, 280, 336\}$
x_{29}	Gb/s	Tasa de línea máxima del *lightpath* (izquierda)	$\{0, 56, 112, 168, 224, 280, 336\}$
x_{30}	Gb/s	Tasa de línea mínima del *lightpath* (derecha)	$\{0, 56, 112, 168, 224, 280, 336\}$
x_{31}	Gb/s	Tasa de línea máxima del *lightpath* (derecha)	$\{0, 56, 112, 168, 224, 280, 336\}$
x_{32}	1	BER mínimo (izquierda)	$x \in \mathbb{R} \mid 0 \leq x \leq 3.8 \cdot 10^{-3}$
x_{33}	1	BER máximo (izquierda)	$x \in \mathbb{R} \mid 0 \leq x \leq 3.8 \cdot 10^{-3}$
x_{34}	1	BER mínimo (derecha)	$x \in \mathbb{R} \mid 0 \leq x \leq 3.8 \cdot 10^{-3}$
x_{35}	1	BER máximo (derecha)	$x \in \mathbb{R} \mid 0 \leq x \leq 3.8 \cdot 10^{-3}$

Tabla 4.5: Descripción de las características del conjunto de datos

Con el objetivo de tener una base sólida de conocimiento y familiarización con el conjunto de datos antes de aplicar metodologías de aprendizaje automático es que hicimos una etapa previa de relevamiento y análisis exploratorio de los datos. Hasta ahora nos centramos en los metadatos, la distribución y las características y etiquetas de los datos. Esto nos permitió entender la topología de la red y fue un primer acercamiento a un conjunto de datos establecido.

Luego del análisis mencionado, nos propusimos ir un paso más allá y ver como interactuaban entre sí las *features* y las *features* con los valores objetivo. Debido al gran volumen de datos y la cantidad de features resultó un poco abrumador interactuar con los datos y sacar conclusiones o tener hipótesis sobre ellos. Es por esto que ver las correlaciones entre las features y tener una herramienta visual para esto fue un avance significativo para la comprensión de los datos.,

Ver los mapas de calor que relacionan las *features* entre sí es útil para ver si hay *features* que tienen una correlación muy alta o si hay *features* que tienen

una correlación negativa. Cualquiera de estos dos casos puede ser un indicio de que una de las *features* es redundante y se puede eliminar. Lo que se muestra a continuación son mapas de calor que por cada valor objetivo muestran las *features* que tienen mayor correlación con el valor, ya sea positiva o negativa.

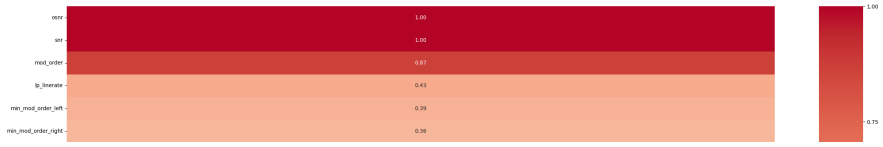

Figura 4.7: Mapa de calor con los valores con influencia directamente proporcional para el valor objetivo OSNR del primer conjunto de datos

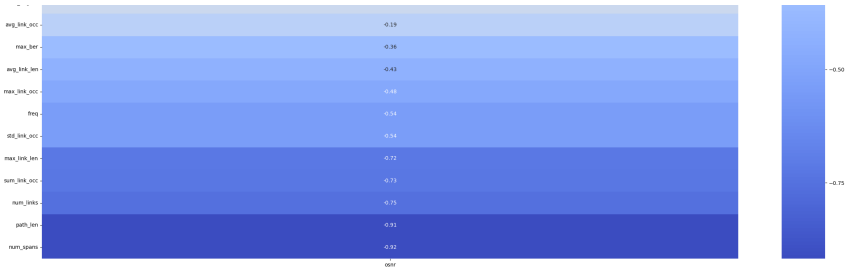

Figura 4.8: Mapa de calor con los valores con influencia inversamente proporcional para el valor objetivo OSNR del primer conjunto de datos

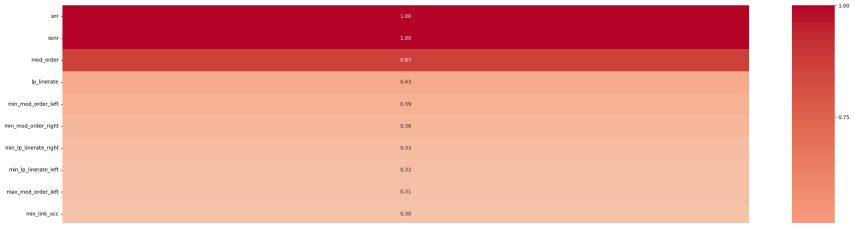

Figura 4.9: Mapa de calor con los valores con influencia directamente proporcional para el valor objetivo SNR del primer conjunto de datos

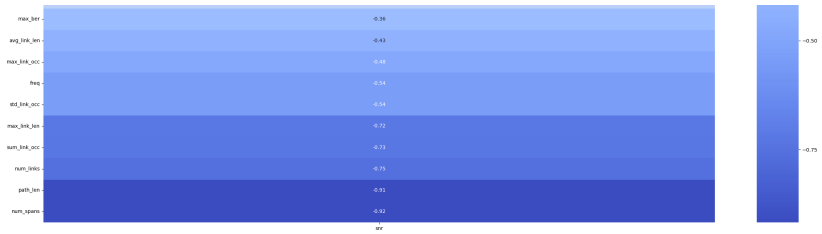

Figura 4.10: Mapa de calor con los valores con influencia inversamente proporcional para el valor objetivo SNR del primer conjunto de datos

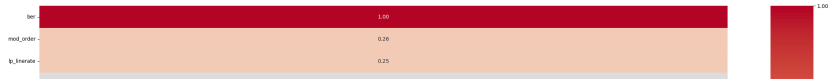

Figura 4.11: Mapa de calor con los valores con influencia directamente proporcional para el valor objetivo BER del primer conjunto de datos

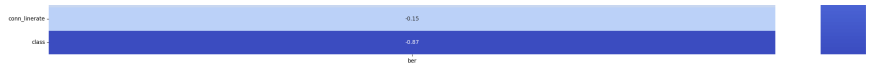

Figura 4.12: Mapa de calor con los valores con influencia inversamente proporcional para el valor objetivo BER del primer conjunto de datos

Figura 4.13: Mapa de calor con los valores con influencia directamente proporcional para el valor objetivo CLASE del primer conjunto de datos

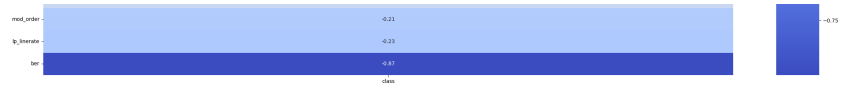

Figura 4.14: Mapa de calor con los valores con influencia inversamente proporcional para el valor objetivo CLASE del primer conjunto de datos

OSNR Como observamos en las Figuras 4.7 y 4.8, las *features* que más se correlacionan con el OSNR son la cardinalidad del formato de modulación y la tasa de datos del *lightpath* (positivamente) y el número de tramos, la longitud del *lightpath*, el número de enlaces, la suma de la ocupación del enlace y la longitud del enlace más largo (negativamente). Tiene sentido que las últimas *features* mencionadas tengan una correlación negativa con OSNR ya que a mayor cantidad de enlaces, mayor distancia recorre la señal, mayor ocupación de los enlaces, mayor longitud de los enlaces, más se degrada la señal y por lo tanto menor OSNR.

A su vez, los distintos tipos de modulación son:

Modulación	**Modulation Order**
BPSK	2
QPSK	4
4QAM	4
8QAM	8
16QAM	16
32QAM	32
64QAM	64

Tabla 4.6: Modulaciones

Se puede ver que a formato de modulación más complejo, mayor OSNR. Es decir, menor ruido en la señal. Otra cosa interesante de ver es que a mayor tasa de datos del *lightpath*, mayor OSNR. Esto se puede interpretar como a mayor tasa de bits por segundo, más robusta es la señal.

snr De las Figuras 4.9 y 4.10 se puede observar que su mapa de calor es casi que idéntico al de OSNR y a su vez tienen una correlación de 1.0 entre ellos. Esto tiene sentido ya que SNR depende del OSNR y observamos anteriormente que sus distribuciones son muy parecidas. En la Tabla 4.7 se puede ver lo que resumimos del análisis sobre el OSNR y SNR.

Correlación	***Feature***	**Explicación**
Positiva	Formato de modulación	A mayor complejidad del formato de modulación, menos ruido en la señal.
Positiva	Tasa de datos del *lightpath*	A mayor tasa de bits por segundo, más robusta es la señal.
Negativa	Número de tramos	Más tramos aumentan la distancia recorrida, degradando la señal.
Negativa	Longitud del *lightpath*	Mayor longitud implica mayor atenuación en la señal.
Negativa	Número de enlaces	Más enlaces degradan la señal acumulativamente.
Negativa	Suma de la ocupación del enlace	Mayor ocupación provoca más interferencia.
Negativa	Longitud del enlace más largo	Los enlaces largos incrementan la degradación de la señal.

Tabla 4.7: Características más y menos correlacionadas con OSNR y SNR.

ber Como se puede ver en las Figuras 4.11 y 4.12, las *features* que más se correlacionan con BER son la cardinalidad del formato de modulación y la tasa de datos del *lightpath* (positivamente) y la tasa de datos de la conexión (negativamente). Es interesante ver que las otras *features* casi que no tienen relación

con BER. Se puede ver en la Tabla 4.8 las correlaciones más importantes.

Correlación	*Feature*	Explicación
Positiva	Formato de modulación	A mayor complejidad del formato de modulación, menos ruido en la señal.
Positiva	Tasa de datos del *lightpath*	A mayor tasa de bits por segundo, más robusta es la señal.
Negativa	Tasa de datos de la conexión	Una mayor tasa de datos de la conexión parece degradar la señal.

Tabla 4.8: Características más y menos correlacionadas con BER.

class Para la clase, las *features* que más se correlacionan son la tasa de datos de la conexión (positivamente) y la cardinalidad del formato de modulación y la tasa de datos del *lightpath* (negativamente) (ver Figuras 4.13 y 4.14). Viendo las *features* que se relacionan con BER, tienen sentido su correlación con class ya que class y BER son inversamente proporcionales dada su definición como muestra la Tabla 4.9

Correlación	*Feature*	Explicación
Positiva	Tasa de datos de la conexión	Una mayor tasa de datos de la conexión hace que la clase sea positiva
Negativa	Formato de modulación	A mayor complejidad del formato de modulación, menos ruido en la señal.
Negativa	Tasa de datos del *lightpath*	A mayor tasa de bits por segundo, más robusta es la señal.

Tabla 4.9: Características más y menos correlacionadas con la clase.

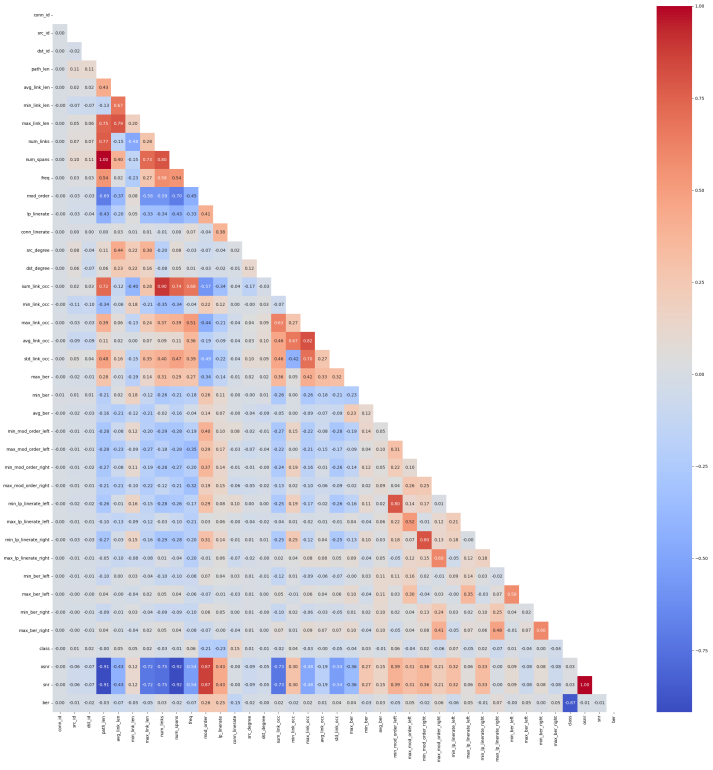

Figura 4.15: Mapa de calor del primer conjunto de datos

Luego de analizar los mapas de calor específicos de como interactúa cada característica de la red con las etiquetas de ruido nos interesó ver como interactuaba cada característica entre sí. Esto es un paso más para entender e inferir los patrones claros dentro de los datos. Todo este proceso lo hicimos en el entendimiento de que nos aportaba información clave en la posterior comprensión del funcionamiento de métodos estadísticos para estimar la QoT. El mapa de calor general en la Figura 4.15 describe las correlaciones. Vemos la fuerte correlación positiva entre el número de tramos y la longitud del *lightpath*, la longitud del enlace más largo y número de enlaces. Por lo tanto, el número de tramos puede ser considerada redundante con la longitud del *lightpath*, la longitud del enlace más largo y número de enlaces. Estas correlaciones tienen sentido ya que todas estas características son semánticamente similares de un punto de vista topológico.

De la misma manera se puede observar que la cardinalidad del formato de modulación y la tasa de datos del *lightpath* tienen una mediana correlación positiva. También se observa que la cardinalidad del formato de modulación tiene una correlación negativa con el número de tramos, la longitud del *lightpath*, la longitud del enlace más largo, número de enlaces y la frecuencia.

Esta sección del análisis nos arrojó información útil para los posteriores modelos de aprendizaje automático que luego vamos a contrastar entre sí. El resumen de las correlaciones entre las *features* se puede ver en la Tabla 4.10

Característica 1	**Característica 2**	**Tipo de correlación**
Número de tramos	Longitud del *lightpath*	Positiva
Número de tramos	Longitud del enlace más largo	Positiva
Número de tramos	Número de enlaces	Positiva
Formato de modulación	Tasa de datos del *lightpath*	Positiva
Formato de modulación	Número de tramos	Negativa
Formato de modulación	Longitud del *lightpath*	Negativa
Formato de modulación	Longitud del enlace más largo	Negativa
Formato de modulación	Número de enlaces	Negativa
Formato de modulación	Frecuencia	Negativa

Tabla 4.10: Correlaciones destacadas entre características de la red.

El análisis de los conjuntos de datos 2, 3 y 4 es muy parecido al del primer conjunto de datos por ello se omite.

Segundo conjunto de datos La Tabla 4.11 muestra las características del segundo conjunto de datos.

Muestras	**Total**		**Proporción**		**Tamaño del conjunto de datos**	
	Positiva	**Negativa**	**Pos.**	**Neg.**	**Lightpaths**	**Estado de red**
Total	1,187,330	93,243	92.72 %	7.28 %	390.8 MB	1.3 TB

Tabla 4.11: Tamaño y balance de clases del segundo conjunto de datos

Tercer conjunto de datos La Tabla 4.12 muestra las características del tercer conjunto de datos.

Muestras	**Total**		**Proporción**		**Tamaño del conjunto de datos**	
	Positiva	**Negativa**	**Pos.**	**Neg.**	**Lightpaths**	**Estado de red**
Total	1,321,452	949,682	71.87 %	28.13 %	403.3 MB	0.8 TB

Tabla 4.12: Tamaño y balance de clases del tercer conjunto de datos

Cuarto conjunto de datos La Tabla 4.13 muestra las características del cuarto conjunto de datos.

Muestras	Total		Proporción		Tamaño del conjunto de datos	
	Positiva	Negativa	Pos.	Neg.	Lightpaths	Estado de red
Total	1,348,293	85,172	93.68 %	6.32 %	411.5 MB	1.4 TB

Tabla 4.13: Tamaño y balance de clases del cuarto conjunto de datos

4.1.4. Conjunto de datos basado en el estado de la red

Anteriormente explicamos todo el análisis hecho a los conjuntos de datos basados en *lightpaths*. Los mencionados conjuntos fueron directos y simples de entender. Luego de haber entendido y habernos familiarizado con los datos procedimos a hacer lo mismo parael conjunto de datos basados en los estados de la red entera.

El conjunto de datos basado en el estado de la red $\Xi \in \mathbb{R}^{D \times N \times L \times F}$ incluye D estados de la red $\mathbf{X} \in \mathbb{R}^{N \times L \times F}$. El descriptor de topología de red $\Theta \in \mathbb{R}^{L \times M}$ complementa el conjunto de datos con características de enlace que describen la topología de la red. N es el número de características de *lightpath*; L es el número de enlaces en la red; F es el número de frecuencias por enlace; y M es el número de características de topología de red. Cada muestra de estado de red se compone de N características de *lightpath*. Las características describen todos los *lightpaths* activos en una instancia de red capturada por enlace l y por frecuencia f.

Entonces, cada muestra del estado de la red se define de la siguiente manera: si un *lightpath* g está activo en los enlaces L_g y en la frecuencia f, entonces la la entrada correspondiente a la frecuencia f y a los enlaces L_g se completa con las matriz de características del *lightpath*. Las entradas de una muestra que se refieren a frecuencias no ocupadas contienen ceros para todo el conjunto de características de *lightpath*. Por otra parte, las entradas del LUT contiene la matriz de características del *lightpath* que llega en todos los recursos que este utilice pero con la diferencia de que los valores de las *features* OSNR, SNR y BER del LUT valen -1.

A su vez, se anexa un descriptor de topología de red $\Theta \in \mathbb{R}^{L \times M}$ que contiene M características de topología de red. Cada característica de topología de red describe un aspecto de los L enlaces en la red. La primera columna de Θ lista la longitud de los enlaces de la red. La segunda columna lista el número de tramos por enlace. El grado del nodo fuente y destino de un enlace se lista en Θ_3 y Θ_4, respectivamente.

En las Tablas 4.14 y 4.15 se describen las variables de los *lightpaths* y la topología de red respectivamente.

x_n	Unidad	Nombre	Dominio de definición
x_1	1	Id de conexión	$x \in \mathbb{N} \mid 1001 < x < 100,000$
x_2	1	Tasa de datos de la línea de conexión	$\{112, 224, 448\}$
x_3	1	Cardinalidad del formato de modulación	$\{0, 4, 8, 16, 32, 64\}$
x_4	dB	OSNR	$x \in \mathbb{R} \mid 12.47 \leq x \leq 33.49$
x_5	dB	SNR	$x \in \mathbb{R} \mid 8.96 \leq x \leq 29.98$
x_6	1	BER	$x \in \mathbb{R} \mid 1.70 \cdot 10^{-12} \leq x \leq 3.8 \cdot 10^{-3}$
x_7	1	Id de nodo fuente	$x \in \mathbb{N} \mid 1 \leq x \leq 75$
x_8	1	Id de nodo destino	$x \in \mathbb{N} \mid 1 \leq x \leq 75$
x_9	km	Longitud del *lightpath*	$x \in \mathbb{R} \mid 24,214 \leq x \leq 7,834,746$
x_{10}	1	Número de tramos	$x \in \mathbb{N} \mid 1 \leq x \leq 106$
x_{11}	1	Número de enlaces	$x \in \mathbb{N} \mid 1 < x \leq 25$
x_{12}	km	Longitud mínima de enlace	$x \in \mathbb{R} \mid 24,214 < x \leq 1,221,189$
x_{13}	km	Longitud máxima de enlace	$x \in \mathbb{R} \mid 24,214 < x \leq 1,221,189$
x_{14}	THz	Frecuencia	$x \in \mathbb{R} \mid x = 192.2 + 0.0375n, n \in \mathbb{N} \mid n < 96$
x_{15}	Gb/s	Tasa de datos del *lightpath*	$\{56, 112, 168, 224, 280, 336\}$

Tabla 4.14: Variables de los *lightpaths* en el conjunto de datos basado en el estado de la red

θ_m	Unidad	Nombre	Dominio de definición
θ_1	km	Longitud del enlace	$x \in \mathbb{R} \mid 24,214 \leq x \leq 1,221,189$
θ_2	1	Número de tramos	$x \in \mathbb{N} \mid 1 \leq x \leq 106$
θ_3	1	Grado del nodo fuente	$x \in \mathbb{N} \mid 2 \leq x \leq 5$
θ_4	1	Grado del nodo destino	$x \in \mathbb{N} \mid 2 \leq x \leq 5$

Tabla 4.15: Variables de la topología de red en el conjunto de datos basado en el estado de la red

Cada muestra de estado de red está anexado al vector objetivo que contiene las métricas de error del LUT.

Es interesante analizar las características de la representación de los datos en el conjunto de datos basado en el estado de la red. Tomemos una muestra de estado de red $\mathbf{X} \in \mathbb{R}^{N \times L \times F}$. Esta muestra, contiene $|G|$ *lightpaths* activos en la red, donde G es el conjunto de *lightpaths* activos. Si consideramos el *lightpath* activo $g \in G$, su matriz de características se va a repetir por cada enlace $l \in L_g$ y por cada frecuencia $f \in F$ que atraviesa. El resto de entradas correspondientes a frecuencias no ocupadas se completan con ceros. Esto hace que las representación de datos tenga una fuerte estructura topológica ya que la matriz que describe a la frecuencia f va a variar dependiendo del *lightpath* que esté activo en esa frecuencia y de la matriz que describa al *lightpath* activo.

Debido a que se cuenta con dos representaciones distinta de datos, un conjunto de datos basado en *lightpaths* y otro basado en el estado de la red, se decidió hacer las secciones posteriores de manera separada para cada representación. Esto se debe a que los modelos de aprendizaje automático que se van

a utilizar para predecir la calidad de transmisión de un *lightpath* son distintos para cada representación de los datos.

Puesto que ambas representaciones de los datos están basadas sobre las mismas instancias de la red pero expresados de formas distintas en este caso y debido a la complejidad inherente de la estructura de la red decidimos enfocarnos en analizar y entender la estructura del conjunto de datos.

La salida de esta etapa del proyecto de grado nos proporcionó un entendimiento del conjunto de datos, un acercamiento a la estructura en la que los datos se almacenan y nos dio un entendimiento sobre las herramientas que se utilizan en el ámbito de la ciencia de datos a la hora de hacer análisis exploratorio de los mismos. El objetivo de facilitar la construcción de modelos de aprendizaje automático lo revisitaremos en las siguientes secciones cuando nos veamos enfrentados a analizar los resultados.

4.2. Metodologías aplicadas sobre el conjunto de datos basado en *lightpaths*

El objetivo de esta étapa del proyecto de grado es aplicar métodos estadísticos para predecir la calidad de transmisión de un *lightpath* que se establece (LUT) a partir de sus características. Utilizamos las siguientes metodologías:

- Exploración de los datos: se analizaron las *features* y las valor objetivo variables de los conjuntos de datos. Se observaron las distribuciones de las variables y se analizó la correlación entre las *features* y las valor objetivo variables. Esto fue presentado en la sección anterior.

- Modelos de aprendizaje automático: se entrenaron modelos de aprendizaje automático para predecir la calidad de transmisión de un *lightpath*. Se utilizó el primer conjunto de datos para entrenar los modelos y se evaluaron los modelos en los conjuntos de datos 1, 2, 3 y 4.

- Ingeniería de *features*: en base a los resultados de los modelos y con la ayuda de los valores de SHAP, se seleccionaron las *features* más importantes para la predicción de la calidad de transmisión.

Hasta ahora hemos visto como impactó la exploración de los datos en el proceso general del proyecto de grado. Esta etapa fue también exploratoria e iterativa que consistió en: construir modelos de aprendizaje automático en el problema de la estimación de la calidad de transmisión para los *lightpath* conjuntos de datos y seleccionar las *features* más importantes para la predicción de la calidad de transmisión. Luego, en base al entendimiento previo de los datos y las selección de features mediante herramientas especializadas construimos nuevos modelos, analizamos nuevamente las salidas de los modelos y refinamos todo el proceso como se puede observar ne la Figura 4.16.

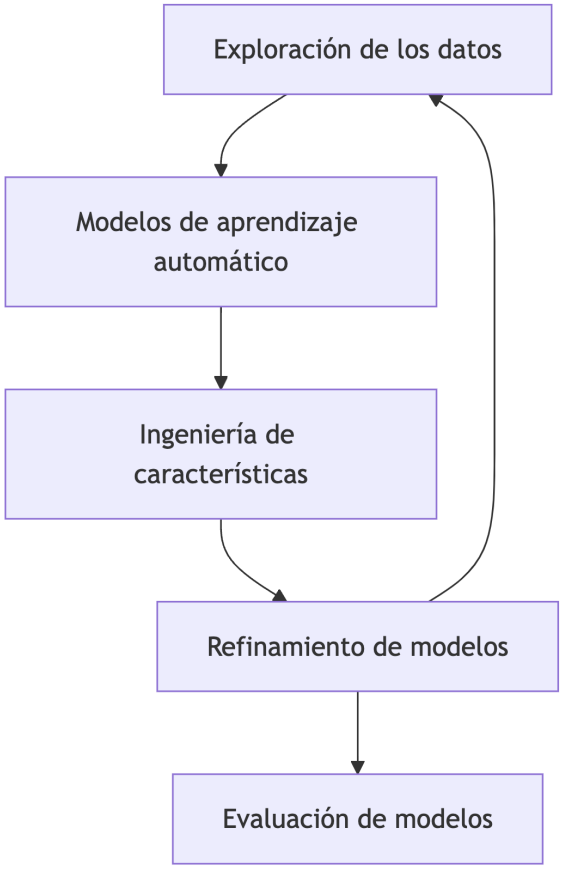

Figura 4.16: Flujo de trabajo para el conjunto de datos basado en *lightpaths*

A continuación presentamos los modelos de aprendizaje automático que usamos para resolver el problema de la estimación de la calidad de transmisión. La intención nuestra fue construir un modelo que actúe como clasificador de la clase y como regresor de las variables continuas.

4.2.1. Trabajo realizado por HHI

En el trabajo presentado en (Bergk y cols., 2022) además de presentar los datos y el análisis exploratorio de los mismos, se presentaron técnicas de aprendizaje automático para predecir la calidad de transmisión de un *lightpath*. Se utilizó un modelo de red neuronal artificial (ANN) con una capa oculta de 256 neuronas y la función de activación tanh. Se consideró un submuestreo de 100.000 muestras tomando en cuenta el desbalance de clases del conjunto de datos entero. El submuestreo es balanceado. Se dividió cada conjunto de datos preprocesado en 70 % de entrenamiento, 20 % de validación y 10 % de prueba. Se

estandarizaron los datos para que el clasificador sea menos sensible a los outliers removiendo la media y escalando a varianza unitaria. Se evaluó el modelo en cada uno de los conjuntos de datos para comparar el rendimiento de los modelos basados en conjuntos de datos que dependen de escenarios de simulación ligeramente diferentes. Se evaluó el modelo entrenado en cada conjunto de datos en los otros conjuntos de datos de la colección. Se observó que el modelo entrenado en el primer conjunto de datos alcanza una precisión por encima del 99 % en los conjuntos de datos dos y cuatro y 93.21 % en el tercer conjunto de datos. Según los puntajes de precisión presentados en la Tabla 4.16, los modelos entrenados en el primer conjunto de datos alcanzan mayores precisión en los otros conjuntos de datos que cualquier otro de los cuatro conjuntos de datos. Por lo tanto, el primer conjunto de datos se muestra como el más completo, versátil y exhaustivo.

Modelos entrenados en	**Conjuntos de datos**			
	1	**2**	**3**	**4**
Primer conjunto	-	99.05	93.21	99.4
Segundo conjunto	59.51	-	66.56	70.32
Tercer conjunto	55.41	57.39	-	57.47
Cuarto conjunto	73.3	99.12	57.59	-

Tabla 4.16: Precisión de los modelos entrenados en el conjunto de entrenamiento de cada conjunto de datos en los otros conjuntos de datos.

4.2.2. Modelos de aprendizaje automático

Luego de haber comprendido la estructura de los datos y sus características decidimos estudiar las metodologías aplicadas en otros trabajos sobre el mismo conjunto de datos. Una vez estudiado todo esto pasamos a la siguiente etapa del proyecto de grado: construir los modelo estadísticos.

Como mencionamos anteriormente, pusimos énfasis en comprender y explicar los resultados de las metodologías, no solo presentar resultados. Con este objetivo en mente elegimos el modelo *HistGradientBoostingRegressor* de la librería scikit-learn. La decisión de seleccionar un modelo regresor liviano de aprendizaje automático lo basamos la necesidad de evaluar su rendimiento en comparación con modelos más complejos utilizados por HHI, como se describió en la sección anterior. Decidimos utilizar ANNs porque priorizamos tener un modelo interpretable. Consideramos dos enfoques: boosting y bagging, optando finalmente por el primero.

Evaluación de los Modelos Para evaluar los modelos utilizamos las métricas F1 y precisión para la clase y R2 para las variables continuas. Estas métricas permiten medir el rendimiento del modelo en diferentes aspectos.

Regresión en múltiples variables Utilizamos *MultiOutputRegressor* para manejar la predicción de múltiples variables objetivo. Esta estrategia consiste

en ajustar un regresor independiente para cada variable objetivo, extendiendo así los regresores que no soportan nativamente la regresión multiobjetivo. Esto permite tratar cada variable objetivo de manera separada, optimizando el rendimiento de cada una según sus características específicas.

Análisis de SHAP Utilizamos los valores de SHAP para analizar la importancia de las *features* en la predicción de la calidad de transmisión. Esto fue una capa más de análisis sobre los mapas de calor y correlación entre variables hecho anteriormente. Nos permitió validar los resultados de etapas anteriores.

Primer modelo - HistGradientBoostingRegressor sin ajustes en los hiperparámetros De manera exploratoria decidimos utilizar *HistGradientBoostingRegressor* sobre el primer conjunto de datos sin ajustes en los hiperparámetros y sin preprocesamiento especial de los datos. La división de datos fue de 70 % para entrenamiento y 30 % para pruebas.

Usamos *MultiOutputRegressor* que permite crear un modelo de regresión para cada valor objetivo variable. En este caso, se crearon 4 modelos de regresión, uno para cada valor objetivo variable (class, OSNR, SNR, BER). Para el caso de la clase, se hizo un posprocesamiento de lo que devolvió el regresor donde se redondearon los valores a 0 o 1.

Para evaluar el modelo entrenado se utilizó F1 y precisión para la clase y R2 para OSNR, SNR y BER.

Aparte, se hizo un análisis de SHAP para ver la importancia de cada *feature* en la predicción de cada valor objetivo variable.

Segundo modelo - HistGradientBoostingRegressor con cinco *features* seleccionadas Una vez que construimos, evaluamos y analizamos el primer modelo como comentamos al principio de esta sección tomamos todo esto en cuenta para ajustar el entrenamiento y las variables para construir un segundo modelo.

Esta iteración consistió en entrenar el modelo *HistGradientBoostingRegressor* con solo 5 *features* seleccionadas en base al análisis de shap del modelo anterior. Las *features* seleccionadas fueron: cardinalidad del formato de modulación, cantidad de tramos, longitud del *lightpath*, promedio de ocupación del enlace y frecuencia.

Tercer modelo - HistGradientBoostingRegressor con tres *features* seleccionadas La siguiente iteración de este experimento consistió en entrenar el modelo HistGradientBoostingRegressor con solo 3 *features* seleccionadas. Las *features* seleccionadas fueron: cardinalidad del formato de modulación, cantidad de tramos y longitud del *lightpath*.

Cuarto modelo - HistGradientBoostingRegressor con dos *features* seleccionadas Por último, se entrenó el modelo HistGradientBoostingRegressor

con solo 2 *features* seleccionadas en base a los analisis hechos. Las *features* seleccionadas fueron: cardinalidad del formato de modulación y cantidad de tramos.

Se hizo un trabajo de selección de *features* que permitió reducir la complejidad del modelo y sacar conclusiones sobre las *features* más relevantes para la predicción de la calidad de transmisión.

4.3. Resultados obtenidos sobre el conjunto de datos basado en *lightpaths*

Luego de haber hecho una breve descripción de cada modelo vamos a considerar el por qué de las decisiones y los resultados de cada modelo hecho en este proyecto de grado sobre el conjunto de datos basado en *lightpaths*.

4.3.1. Primer modelo - HistGradientBoostingRegressor sin ajustes en los hiperparámetros

Los valores que obtuvimos al evaluar el primer modelo se pueden observar en la tabla 4.17. Como vemos, tanto el F1 y la precisión para la clase como el R2 para OSNR, SNR y BER son muy altos. Esto indica que el modelo es muy bueno para predecir la calidad de transmisión de un *lightpath*.

Como se puede ver en la tabla 4.17, el modelo lo entrenamos sobre el 70 % del primer conjunto de datos y lo evaluamos sobre el 30 % restante así como sobre los otros conjuntos de datos. Esto nos dio una idea de cómo se comporta el modelo en diferentes escenarios de simulación y si es capaz de generalizar bien incluso ante distintas características topológicas (como el tercer conjunto de datos).

Conjunto de datos	F1	Precisión	R2
Primero (30 % datos para prueba)	0.9940	0.9909	0.9959
Segundo (100 % de los datos)	0.9984	0.9971	0.9936
Tercero (100 % de los datos)	0.9922	0.9890	0.9850
Cuarto (100 % de los datos)	0.9984	0.9972	0.9918

Tabla 4.17: Resultados de los conjuntos de datos

Además, graficamos los valores predichos en función de los valores reales para cada variable continua (OSNR, SNR, BER). Esto se hizo para los 4 conjuntos de datos. Para facilidad a la hora de graficar tomamos en cuenta 10000 valores aleatorios de cada subconjunto de datos sobre los que se evaluó.

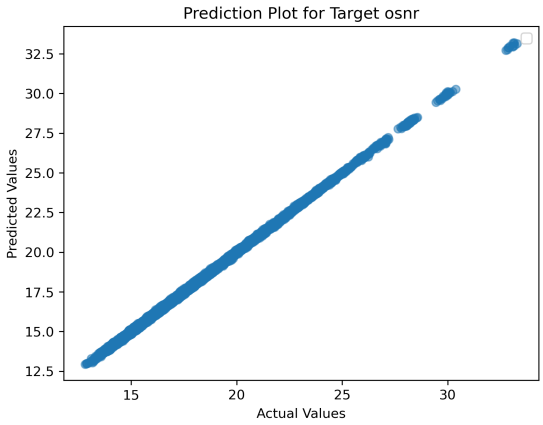

Figura 4.17: Valores predichos en función de valores reales para OSNR

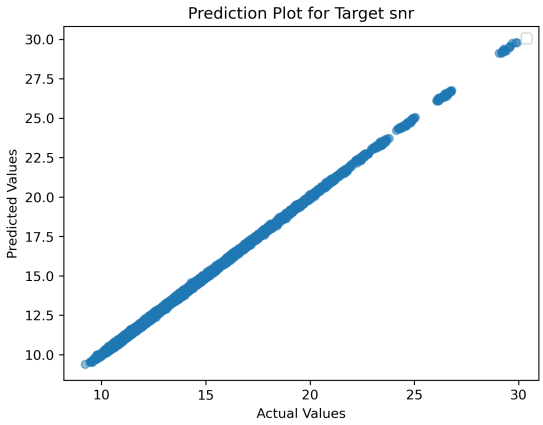

Figura 4.18: Valores predichos en función de valores reales para SNR

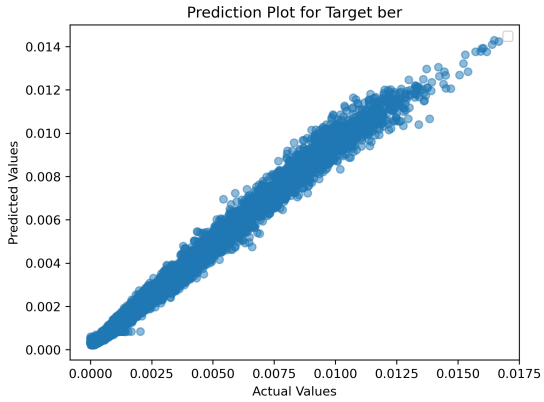

Figura 4.19: Valores predichos en función de valores reales para BER

Pudimos observar en las Figuras 4.17, 4.18 y 4.19 que los valores predichos son muy cercanos a los valores reales para OSNR y SNR. Para BER, se nota una mayor dispersión de los valores predichos respecto a los valores reales.

A continuación en las Figuras 4.20, 4.21, 4.22 y 4.23 se presentan los análisis de SHAP para cada valor objetivo variable.

Una vez obtenidos los resultados del modelo nos concentramos en la fase de análisis de los resultados y las metodologías para refinar el mismo. En las Figuras 4.20, 4.21, 4.22 y 4.23 podemos ver la parte central del estudio que son los valores de SHAP para cada valor objetivo.

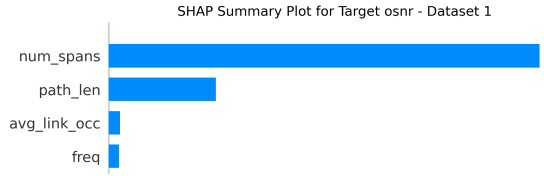

Figura 4.20: Análisis de SHAP para OSNR en el primer conjunto de datos

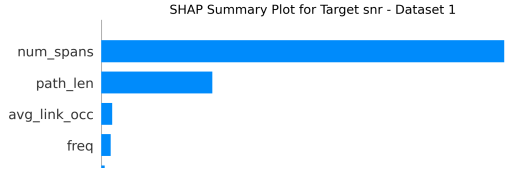

Figura 4.21: Análisis de SHAP para SNR en el primer conjunto de datos

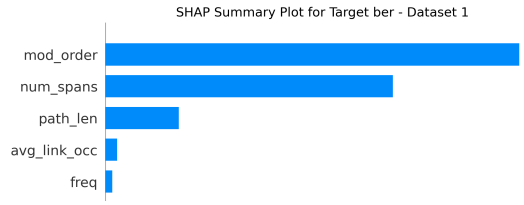

Figura 4.22: Análisis de SHAP para BER en el primer conjunto de datos

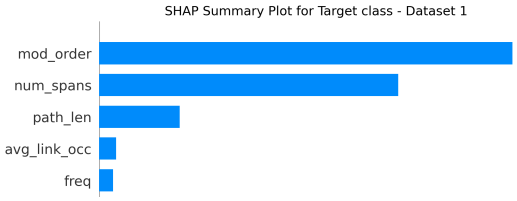

Figura 4.23: Análisis de SHAP para class en el primer conjunto de datos

Para BER, las *features* más importantes son: cantidad de tramos y longitud del *lightpath*. Para SNR, las *features* más importantes son: cantidad de tramos y longitud del *lightpath*. Para BER, las *features* más importantes son: cardinalidad del formato de modulación, cantidad de tramos y longitud del *lightpath*. Para la predicción de class son: cardinalidad del formato de modulación, cantidad de tramos y longitud del *lightpath*. Tiene sentido ya que la variable class se define en base al BER. Se puede ver que las *features* más importantes en general son cardinalidad del formato de modulación, cantidad de tramos y longitud del *lightpath* seguidas de promedio de ocupación del enlace y frecuencia y se resumen en la Tabla 4.18

Valor Objetivo	***Features* Más Importantes Según SHAP**
OSNR	Cantidad de tramos, longitud del *lightpath*
SNR	Cantidad de tramos, longitud del *lightpath*
BER	Formato de modulación, cantidad de tramos, longitud del *lightpath*
Clase	Formato de modulación, cantidad de tramos, longitud del *lightpath*
General	Formato de modulación, cantidad de tramos, longitud del *lightpath*, promedio de ocupación del enlace, frecuencia

Tabla 4.18: *Features* más importantes según los valores de SHAP para cada valor objetivo.

Los resultados que obtuvimos fueron muy buenos y sacamos conclusiones sobre el conjunto de datos basado en *lightpaths*. A priori, haciendo el análisis de correlación de las *features* con las valor objetivo variables, se pudo ver que las *features* más correlacionadas con las medidas de error eran la cardinalidad del formato de modulación, tasa de datos del *lightpath*, cantidad de tramos, longitud del *lightpath*, número de enlaces, suma de la ocupación del enlace, longitud del enlace más largo y tasa de datos de la conexión. El modelo de aprendizaje

automático construído arroja resultados previsibles y coherentes con el análisis de correlación de las *features*.

Podemos entonces ver que la fase de análisis exploratorio de datos proveyó herramientas importantes a la hora de toma de decisiones sobre los datos que fueron comprobadas en esta etapa.

A la hora de establecer un *lightpath*, las variables que más influyen en la calidad de transmisión son la modulación, el número de tramos, la longitud del *lightpath*, el promedio de la ocupación de los enlaces y la frecuencia.

4.3.2. Segundo modelo - HistGradientBoostingRegressor con cinco *features* seleccionadas

Con todas las observaciones hechas hasta el momento fue que analizamos los resultados obtenidos en el segundo modelo. Los valores obtenidos fueron:

Conjunto de datos	F1	Precisión	R2
Primero (30 % datos para prueba)	0.9927	0.9889	0.9954
Segundo (100 % de los datos)	0.9984	0.9970	0.9929
Tercero (100 % de los datos)	0.9912	0.9885	0.9851
Cuarto (100 % de los datos)	0.9985	0.9971	0.9916

Tabla 4.19: Resultados del Segundo modelo

Se puede ver que con las cinco *features* más importantes extraídas de la Tabla 4.18 se obtienen resultados muy buenos y comparables con el modelo anterior. Observamos que sin las restantes *features* igual se consigue un modelo con resultados comparables.

En este caso, también hicimos un análisis de SHAP que arrojó los mismos resultados que en el modelo anterior, es decir, las *features* más importantes se repitieron.

4.3.3. Tercer modelo - HistGradientBoostingRegressor con tres *features* seleccionadas

En esta tercera iteración de la construcción de un modelo estadístico para predecir las métricas de error tomamos como entrada los resultados del primer y segundo modelo. Los resultados obtenidos fueron:

Conjunto de datos	F1	Precisión	R2
Primero (30 % datos para prueba)	0.9880	0.9841	0.9927
Segundo (100 % de los datos)	0.9971	0.9955	0.9913
Tercero (100 % de los datos)	0.9838	0.9768	0.9797
Cuarto (100 % de los datos)	0.9975	0.9955	0.9901

Tabla 4.20: Resultados del Tercer modelo

El modelo nos siguió mostrando que con menos variables era igual de posible obtener resultados comparables de vuelta. Esta vez con las primeras tres *features* más importantes extraídas de la Tabla 4.18 obtuvimos buenos resultados para las variables continuas y la clase.

4.3.4. Cuarto modelo - HistGradientBoostingRegressor con dos *features* seleccionadas

Para el cuarto modelo aparte de tener en cuenta los resultados de las anteriores iteraciones del trabajo tuvimos en cuenta lo que arrojó el análisis exploratorio de datos. Si recordamos los valores obtenidos en la Tabla 4.15 podemos ver que el número de tramos y el largo del *lightpath* tienen una correlación de 1.0. Con esta premisa quitamos la segunda característica del conjunto de variables y con solo la cardinalidad del formato de modulación y cantidad de tramos obtuvimos estos resultados:

Conjunto de datos	F1	Precisión	R2
Primero (30 % datos para prueba)	0.9868	0.9801	0.9924
Segundo (100 % de los datos)	0.9969	0.9943	0.9912
Tercero (100 % de los datos)	0.9897	0.9853	0.9807
Cuarto (100 % de los datos)	0.9974	0.9951	0.9897

Tabla 4.21: Resultados del Cuarto modelo

Además, como otra herramienta para comprobar los resultados, graficamos valores predichos en función de los valores reales para cada variable continua (OSNR, SNR, BER) para el cuarto modelo en el primer conjunto de datos.

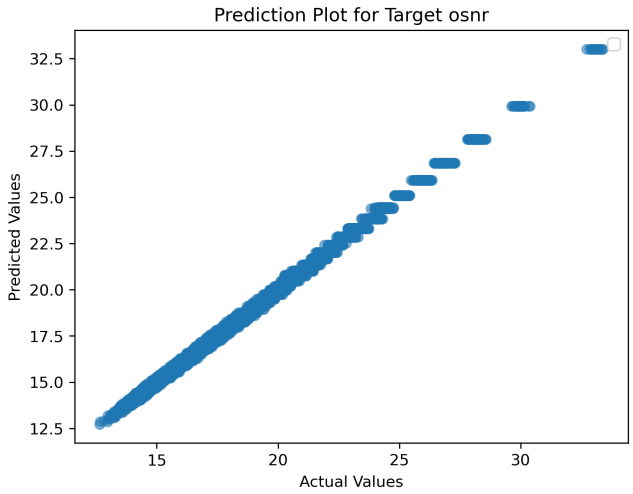

Figura 4.24: Valores predichos en función de valores reales para OSNR

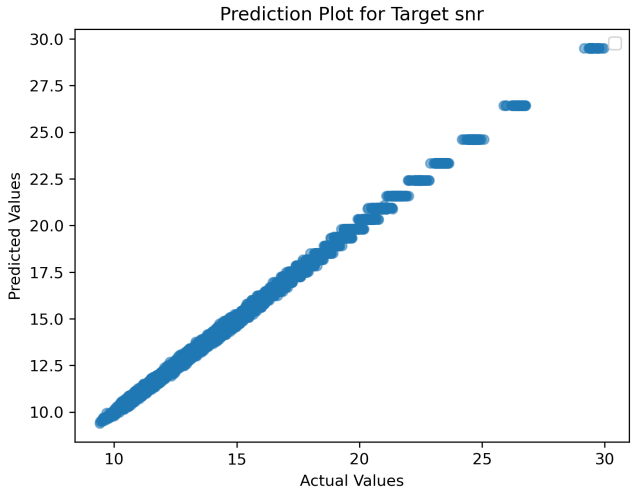

Figura 4.25: Valores predichos en función de valores reales para SNR

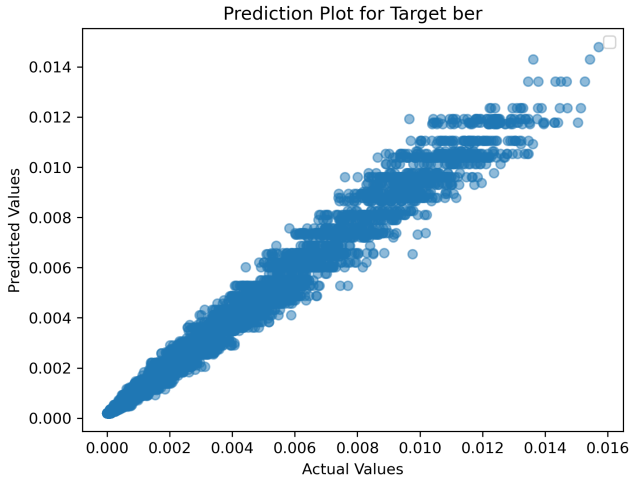

Figura 4.26: Valores predichos en función de valores reales para BER

En las Figuras 4.24, 4.25 y 4.26 se puede ver que los valores predichos son muy cercanos a los valores reales para OSNR y SNR y que para BER hay una mayor dispersión de los valores predichos respecto a los valores reales. Aún con 2 variables, el modelo sigue siendo muy bueno y nos demuestra que la cardinalidad del formato de modulación y la cantidad de tramos son las variables más importantes para la predicción de la calidad de transmisión.

Volviendo al análisis de correlación hecho a priori habíamos visto la fuerte

correlación entre el número de tramos y la longitud del *lightpath*, la longitud del enlace más largo y número de enlaces. También habíamos visto que la cardinalidad del formato de modulación tenía correlación el *lightpath* tasa de datos dado que el segundo depende del primero. Ambas correlaciones se ven reflejadas a la hora de entrenar el modelo y muestra que el formato de modulación y el número de tramos describen bien al resto de las variables.

En comparación con el modelo propuesto en (Bergk y cols., 2022), se puede ver que los resultados obtenidos son muy buenos y que se logró una buena predicción de la calidad de transmisión de un *lightpath* utilizando un modelo de aprendizaje automático simple e interpretable. Además, comprobamos que de igual manera que en (Bergk y cols., 2022), el primer conjunto de datos sirve como base de entrenamiento para evaluar los modelos en los otros conjuntos de datos.

4.3.5. Herramientas utilizadas

Para la realización de esta parte del trabajo utilizamos las siguientes herramientas de hardware con las siguientes características.

Modelo	Procesador	Memoria RAM	Tiempo de Entrenamiento (s)
Primer modelo	MacBook Pro Apple M1 Pro	16 GB	26.95
undo modelo	MacBook Pro Apple M1 Pro	16 GB	13.10
Tercer modelo	MacBook Pro Apple M1 Pro	16 GB	12.24
Cuarto modelo	MacBook Pro Apple M1 Pro	16 GB	12.17

Tabla 4.22: Herramientas utilizadas para la realización de los modelos

El trabajo realizado en esta etapa del proyecto de grado se encuentra en el repositorio de Gitlab (Facultad de Ingeniería - MINA, s.f.-b).

4.3.6. Comparación con el Trabajo de HHI

En el trabajo presentado por (Bergk y cols., 2022), se utilizó un modelo de red neuronal artificial (ANN) para predecir la calidad de transmisión de un *lightpath*. La ANN, con una capa oculta de 256 neuronas y función de activación tanh, logró precisiones superiores al 99 % en conjuntos de datos 02 y 04, y 93.21 % en el tercer conjunto de datos. El submuestreo balanceado y la estandarización de datos fueron puntos importantes de su metodología.

Por otro lado, en el estudio que nostros realizamos empleamos *HistGradientBoostingRegressor* debido a su capacidad de interpretabilidad y simplicidad en comparación con las ANN. A pesar de no utilizar técnicas de submuestreo ni estandarización especial, alcanzamos altos niveles de precisión y R2 consistentes en todos los conjuntos de datos evaluados.

Conjunto de datos	Precisión (HHI)	Precisión de modelos (%)
Segundo	99.05	Primer: 99.71, Segundo: 99.70, Tercero: 99.55, Cuarto: 99.43
Tercero	93.21	Primer: 98.90, Segundo: 98.85, Tercero: 97.68, Cuarto: 98.53
Cuarto	99.4	Primer: 99.72, Segundo: 99.71, Tercero: 99.55, Cuarto: 99.51

Tabla 4.23: Comparación de precisión entre modelos entrenados en el primer conjunto de datos y evaluados en los otros.

En términos de interpretabilidad, el uso de SHAP en el estudio permitió identificar claramente las características más importantes, tales como cardinalidad del formato de modulación, cantidad de tramos, y longitud del *lightpath*. Esto contrasta con el enfoque de ANN de (Bergk y cols., 2022), que, aunque efectivo, no proporciona la misma claridad en cuanto a la importancia de las características.

Los resultados obtenidos (Tabla 4.23) muestran que con una adecuada selección de características, es posible obtener modelos de alta precisión sin la necesidad de complejidades adicionales asociadas con las redes neuronales profundas. Además, el análisis de correlación y la reducción del número de *features* permitieron simplificar el modelo manteniendo una alta precisión y explicabilidad.

4.4. Metodologías aplicadas sobre el conjunto de datos basado en el estado de la red

Luego de haber estudiado y trabajado sobre la primera representación de los datos empezamos con el conjunto de datos basado en el estado de la red. El objetivo de esta sección de trabajo en el proyecto de grado fue estimar la calidad de transmisión del *lightpath* que se establece en una red extendiendo el conocimiento y los resultados de las primeras etapas hacia esta representación de datos y enfrentarnos a un conjunto de datos más rico y complejo. El flujo de trabajo fue el que se muestra en la figura 4.27 donde se aplicaron las siguientes metodologías:

- Procesamiento inicial de los datos crudos. Transformación de datos a una estructura conocida y familiar.
- Preprocesamiento para el modelo de aprendizaje automático.
- Creación de modelos de aprendizaje automático para estimar la QoT,
- Optimización y refinamiento de los modelos en base a los resultados obtenidos.

Debido a que ambas representaciones de datos se construyeron sobre las mismas instancias y simulaciones de red y lo que se cambia es el enfoque y la perspectiva de los datos trasladamos todo el conocimiento obtenido hacia esta nueva etapa con el mismo objetivo.

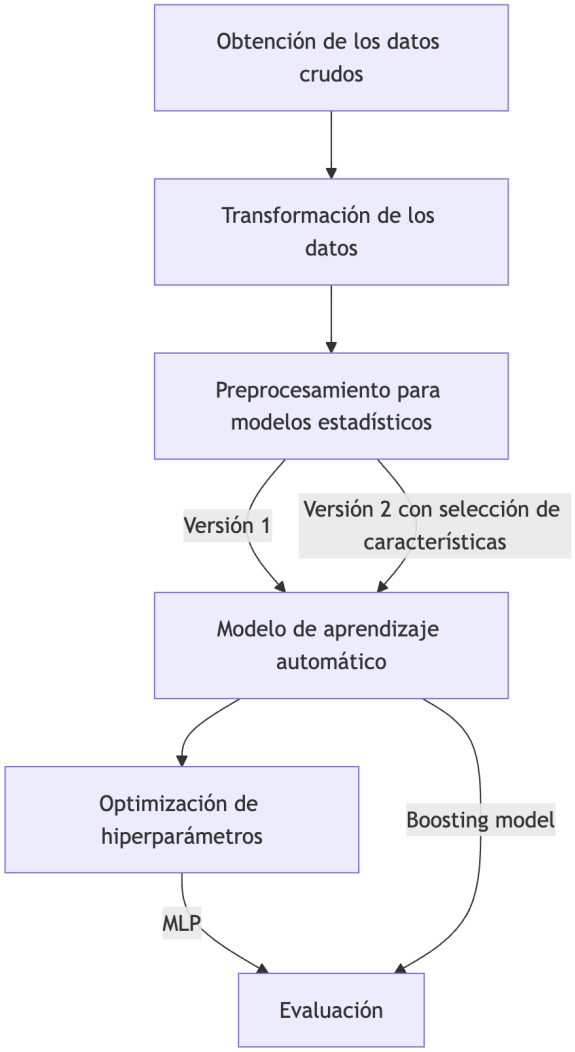

Figura 4.27: Flujo de trabajo para el conjunto de datos basado en el estado de la red

4.4.1. Trabajo realizado por HHI

En (Safari y cols., 2021) se proponen técnicas de aprendizaje automático para predecir la calidad de transmisión de un *lightpath* en una red óptica elástica con la información de toda la red en ese momento. Se propone una formulación de estimación de calidad de transmisión en toda la red que abstrae el estado de la red en un conjunto de matrices. Cada estado de red se describe en un conjunto de dos matrices $L \times (2 + F)$, donde L es el número de enlaces de la

topología de red y F es el número de bandas de frecuencia en cada enlace. Cada matriz de estado de red tiene 2 + F columnas, habiendo dos columnas para dos características topológicas; una para la longitud de cada enlace y otra para el número de tramos de cada enlace y el resto de las columnas corresponden a características de los *lightpaths* en esas bandas de frecuencia. Las matrices consideradas en la formulación se pueden ver en las Tablas 4.24 y 4.25 donde los primeros dos elementos de $D_i, 1$ y $D_i, 2$ son iguales, el resto de los elementos toman valores alternativos. Cada elemento de $D_i, 1$ y $D_i, 2$ tiene el valor 0 para los elementos que corresponden a bandas de frecuencia no ocupadas en cada enlace. $D_i, 1$ tiene valores de BER de todos los *lightpaths* activos en sus canales y enlaces correspondientes. $D_i, 1$ tiene el valor -1 en las bandas de frecuencia y enlaces correspondientes al LUT. $D_i, 2$ tiene la cardinalidad de modulación $mod\_order_{l,f}$ de todos los *lightpaths* activos y el LUT.

Link	**Largo (len)**	**Número de tramos (nspan,l)**	**Bandas de Frecuencia (1 a F)**		
1	len_1	nspan_1	0 / BER / -1	...	0 / BER / -1
2	len_2	nspan_2	0 / BER / -1	...	0 / BER / -1
⋮	⋮	⋮	⋮	⋱	⋮
L	len_L	nspan_L	0 / BER / -1	...	0 / BER / -1

Tabla 4.24: Matriz $D_{i,1}$: Longitud, Número de tramos y BER para Cada Enlace y Slot de Frecuencia

Link	**Largo (len)**	**Número de tramos (nspan,l)**	**Bandas de Frecuencia (1 a F)**		
1	len_1	nspan_1	0 / $mod\_order_{l,1}$	...	0 / $mod\_order_{l,F}$
2	len_2	nspan_2	0 / $mod\_order_{l,1}$	...	0 / $mod\_order_{l,F}$
⋮	⋮	⋮	⋮	⋱	⋮
L	len_L	nspan_L	0 / $mod\_order_{l,1}$	...	0 / $mod\_order_{l,F}$

Tabla 4.25: Matriz $D_{i,2}$: Longitud, Número de tramos y Modulación para Cada Enlace y Slot de Frecuencia

Se transformaron los TEDs (*Traffic Engineering Database records*) obtenidos de las simulaciones a conjuntos de datos de calidad de transmisión en toda la red en un paso de preprocesamiento. Cada muestra describe un estado de red durante la fase que llega un *lightpath* e incluye un LUT y todos los *lightpaths* previamente establecidos en la red. Se etiquetaron las muestras por el BER del LUT.

Se dividió los datos en un 70 % de entrenamiento, 20 % de validación y 10 % de prueba y se escaló dividiendo sobre el valor máximo de cada característica correspondiente. Se estudiaron tres escenarios diferentes para cada topología de red. Primero se consideró solo $D_i, 1$ o $D_i, 2$ como entrada a DCNN y el mejor modelo obtenido no alcanzó una precisión de prueba superior al 74 %.

Se modificó la arquitectura y se consideró un tensor 3D para cada muestra, incluyendo tanto $D_i, 1$ como $D_i, 2$, como entrada a DCNN y la precisión resultante mejoró significativamente para ambas redes. La formulación propuesta

basada en DCNN logró una precisión del 98.79 % y 99.52 % sobre el conjunto de datos de prueba para las redes CORONET y TID (Telefónica, España), respectivamente. Los resultados que obtuvieron fueron muy buenos y se pueden ver en la Tabla 4.26. Se ponen solo los resultados sobre la topología CORONET ya que es la misma topología que se utiliza en el presente caso de estudio.

Conjunto de datos	Precisión de Entrenamiento	Precisión de Validación	Precisión de Prueba
Di,1	72.75 %	74.10 %	72.75 %
Di,2	65.66 %	72.35 %	65.66 %
Di,1 and Di,2	98.79 %	99.52 %	98.79 %

Tabla 4.26: Resultados en términos de precisión para la topología CORONET.

4.4.2. Transformación inicial de los datos crudos

La primera meta al realizar un estudio de la representación de datos fue transformar los datos a una forma familiar y conocida. Los datos crudos se encontraban en un formato multidimensional que contenía las siguientes dimensiones vistas en la Tabla 4.27. Cada una de las variables principales contenía una matriz de datos.

Categoría	Descripción
Dimensiones	muestra, *features* del lightpath, enlace, frecuencia, *features* de la red, métricas objetivo
Variables Principales	Datos: Información de estado de red con dimensiones para cada muestra, *lightpath*, enlace y frecuencia.
	Descriptor topológico: Características topológicas para cada enlace.
	Objetivo: Valores objetivo para cada muestra que comprenden OSNR, SNR, BER y clase.
	Otras variables: Identificadores y descripciones de muestras, características de *lightpaths*, topología de red y frecuencias.

Tabla 4.27: Descripción de las dimensiones y variables principales del conjunto de datos

Debido a la complejidad de los datos, lo primero que hicimos aplicando una técnica intuitiva al ver la multidimensionalidad en la estructura de datos fue aplanar las *features* de los *lightpaths* y anexar los datos descriptores de cada enlace. A esto se le agregó la incorporación de los valores objetivo y la creación de columnas temporales para ciertas *features* y valores objetivo. Por lo tanto, el conjunto de datos resultante adoptó la siguiente estructura:

1. **Representación de la muestra**: Cada fila del conjunto de datos representa una muestra de estado de la red.

2. **Aplanamiento de *features* de los *lightpaths***: Para cada fila, por cada *slot* de frecuencia dentro de un enlace, se añadió la matriz de características del *lightpath*. Esto resultó en 15 columnas por cada *slot* de frecuencia en la red, lo que produce un total de $15 \times 198 \times 96 = 284.160$ columnas, con el formato *feature_link_slot*.

3. **Anexo de características de la topología de red**: Se agregaron las características topológicas correspondientes a cada enlace, lo que resultó en $4 \times 198 = 792$ columnas, con el formato *nt_feat_link*.

4. **Incorporación de los valores objetivo**: Se anexaron los valores objetivo correspondientes, generando 4 columnas adicionales con el formato *valor_objetivo*.

5. **Creación de columnas temporales para ciertas *features***: Se generaron columnas temporales para las siguientes *features*: cardinalidad del formato de modulación, cantidad de tramos, OSNR, SNR y BER. Estas columnas temporales se calcularon utilizando un promedio con decaimiento exponencial basado en las últimas 10 filas.

   - La selección de estas 10 filas surgió a partir del análisis de las *features* más importantes para el modelo de *lightpath*, así como para OSNR, SNR y BER, bajo la hipótesis de que los errores particulares de las bandas de frecuencia contribuyen al error general.

   Esto dio lugar a $5 \times 198 \times 96 = 95.040$ columnas adicionales, con el formato *feature_temp*.

6. **Desplazamiento temporal de los valores objetivo**: Se desplazaron los valores objetivo tres veces, de forma que se crearon columnas para *valor_objetivo*, *valor_objetivo en t-1*, *valor_objetivo en t-2* y *valor_objetivo en t-3*.

7. **Mantenimiento del orden de las filas**: A lo largo del proceso de transformación, se mantuvo el orden original de las filas del conjunto de datos.

El conjunto de datos resultante cuenta con dos dimensiones: 80.000 filas y 399.976 columnas, y se almacenó en fragmentos de 500 filas, cada uno ocupando un espacio de 800 MB.

Para procesar los datos, se tomaron las primeras 80.000 filas del conjunto total y se dividieron en fragmentos de 500 filas. Cada fragmento se dividió en 12 procesos, que realizaron la transformación en paralelo. Esta división se llevó a cabo para evitar la carga total del conjunto de datos en memoria, dado su elevado costo, y para permitir la agregación de las variables de *lag*.

El procesamiento de cada fragmento de 500 filas tomó aproximadamente 30 minutos, lo que resultó en un tiempo total de procesamiento de 80 horas.

Finalmente, una de las características más relevantes de este conjunto de datos es la elevada cantidad de columnas generadas y lo esparso de los datos ya

que en el paso número dos de la transformación de datos si el *slot* de frecuencia estaba desocupado se colocó una matriz de ceros para indicar que el recurso no estaba ocupado.

4.4.3. Preprocesamiento para el modelo de aprendizaje automático - Primera versión

Una vez que contamos con un formato conocido de datos se genera un preprocesamiento de los datos con el objetivo de generar una entrada para un método de aprendizaje automático. Ya desde la transformación de los datos que enfrentamos nuevos desafíos por el gran volumen de datos a manipular y que impactó cada pequeña parte de esta fase del proyecto de grado. El preprocesamiento se pudo ejecutar solo sobre un fragmento de los datos crudos (50.000) filas debido a lo costoso del paso de aplicar reducción de dimensionalidad. El paso de aplicar PCA generó un cuello de botella que hizo que los modelos de aprendizaje automático ejecutados sobre este conjunto de datos tuvieran solo 50.000 filas como entrada. Este consistió en las siguientes etapas:

1. **Reducción de precisión numérica**: Se convirtió el tipo de dato de *float64* a *float32* con el fin de reducir el uso de memoria sin perder precisión significativa.

2. **Selección de variables**: Se seleccionaron las siguientes columnas relevantes para el modelo: cardinalidad del formato de modulación, número de tramos y las métricas OSNR, SNR y BER de cada lightapth en su forma temporal (teniendo en cuenta los últimos diez registros) creadas en el paso anterior. Esta selección redujo el conjunto de datos a un total de 95.040 columnas.

3. **División del conjunto de datos**: El conjunto de datos fue dividido en tres subconjuntos: entrenamiento (68 %), validación (12 %) y prueba (20 %). Esta división garantiza que el modelo pueda ser evaluado adecuadamente en diferentes etapas.

4. **Normalización de variables**: Se aplicó un proceso de normalización a las variables continuas de X (las características) y al vector Y (los valores objetivo). Es importante destacar que la clase no fue normalizada para preservar su integridad como variable categórica.

5. **Reducción de dimensionalidad con PCA**: Se aplicó un análisis de componentes principales (PCA, por sus siglas en inglés) al conjunto de *features*, reduciendo el número de dimensiones a 4500 componentes que mantienen el 87 % de la varianza original de los datos.

6. **Preservación del orden de las filas**: Durante el preprocesamiento, se garantizó la preservación del orden original de las filas. Esto se hizo debido a que para este preprocesamiento se tomó al conjunto de datos como una secuencia de datos.

La normalización y el análisis de componentes principales (PCA) se llevaron a cabo utilizando únicamente el subconjunto de entrenamiento. Posteriormente, se aplicaron los mismos valores de media y desviación estándar calculados durante la normalización a los conjuntos de validación y prueba, para asegurar la consistencia entre los diferentes conjuntos de datos.

Debido a la alta dimensionalidad de los datos, que contenían 50.000 filas y 95.040 columnas, la aplicación de PCA presentó desafíos importantes. Inicialmente, se intentó aplicar el método tradicional de PCA, pero surgieron problemas de memoria, ya que este enfoque requiere cargar todo el conjunto de datos en la memoria RAM.

Se exploraron alternativas como bibliotecas que manejan grandes conjuntos de datos que no entran en memoria RAM y *IncrementalPCA* que es una técnica que va aplicando PCA a fragmentos de los datos incrementalmente pero ninguna de estas opciones logró procesar el conjunto de datos de manera efectiva. Finalmente, se optó por utilizar *randomized PCA*, un método estocástico de PCA que no requiere cargar todo el conjunto de datos en memoria. Este enfoque utiliza métodos probabilísticos para estimar los componentes principales de manera eficiente, lo que permitió la reducción de dimensionalidad sin comprometer los recursos de memoria.

4.4.4. Preprocesamiento para el modelo de aprendizaje automático - Segunda versión

La segunda versión del preprocesamiento fue similar a la primera versión explicada en 4.4.3, aunque presentó algunas diferencias importantes. Debido al entendimiento de que el conjunto de datos presentaba un fuerte carácter temporal y que los modelos de aprendizaje automático sobre la primera versión de la transformación no tuvieron buen desempeño este conjunto de datos solo tiene datos que toman en cuenta los registros anteriores. Este preprocesamiento se aplicó a las primeras 52.500 filas obtenidas en el primer paso de la transformación de datos 4.4.2. A continuación, se describen las etapas del proceso:

1. **Reducción de precisión numérica**: Se convirtió el tipo de dato de *float64* a *float32* con el objetivo de reducir el uso de memoria, manteniendo una precisión aceptable.

2. **Selección de variables**: Se seleccionaron las columnas correspondientes a las versiones temporales de las siguientes variables: cardinalidad del formato de modulación, número de tramos y las métricas OSNR, SNR y BER de cada lightapth. Esta selección mantuvo un total de 95.040 columnas.

3. **División del conjunto de datos**: El conjunto de datos fue dividido en tres subconjuntos: entrenamiento (68 %), validación (12 %) y prueba (20 %).

4. **Normalización de las variables**: Se aplicó el mismo proceso de normalización que en 4.4.3

5. **Reducción de dimensionalidad con PCA**: Se utilizó un análisis de componentes principales (PCA) sobre las *features*, reduciendo el número de dimensiones a 6000 componentes. Estos componentes retuvieron el 97 % de la varianza original del conjunto de datos.

6. **Preservación del orden de las filas**: Al igual que en la primera versión, se mantuvo el orden de las filas para asegurar la coherencia temporal y secuencial de los datos.

Al aplicar PCA, se enfrentaron las mismas dificultades observadas en la primera versión del preprocesamiento, debido a la alta dimensionalidad del conjunto de datos y los problemas de memoria asociados. Para superar estas dificultades, se optó nuevamente por el uso de *randomized PCA*, un método estocástico que permitió reducir la dimensionalidad sin necesidad de cargar todo el conjunto de datos en la memoria RAM.

4.4.5. Preprocesamiento para el modelo de aprendizaje automático - Tercera versión

La tercera versión del preprocesamiento tomó un encare un tanto diferente. A partir de los resultados obtenidos en los modelos sobre el conjunto de datos basado en *lightpaths*, se concluyó que las *features* más importantes para la predicción de la calidad de transmisión de un *lightpath* que se establece en la red son la cardinalidad del formato de modulación y la cantidad de tramos. Por lo tanto, se puede suponer que estas dos *features* son las más importantes para la predicción de la calidad de transmisión a nivel de la red entera. Se decidió hacer una versión de los datos que utilice solo estas dos *features* para el modelo de aprendizaje automático. El preprocesamiento consistió en:

1. **Reducción de precisión numérica**: Se convirtió el tipo de dato de *float64* a *float32*.

2. **Selección de variables**: Se seleccionaron las columnas cardinalidad del formato de modulación y número de tramos. Esto resulta en 38.016 columnas.

3. **Transformación de los datos de temporales a supervisados**: Se transformaron los datos de un formato de series temporales a uno de aprendizaje supervisado. Este proceso consiste en eliminar la dependencia temporal entre las filas consecutivas, de modo que las predicciones se basen en datos independientes.

    - Para implementar esto, se tomó cada fila del conjunto de datos y se calculó el promedio de las últimas 25 filas (incluyendo la fila actual). Este promedio reemplazó los valores originales de la fila.
    - De manera similar, para el vector objetivo, se tomó el promedio de las últimas 25 filas, excluyendo la fila actual. Este valor se utilizó para

crear una nueva *feature*, reflejando la evolución histórica del valor objetivo.

4. **División del conjunto de datos**: El conjunto de datos fue dividido en tres subconjuntos: entrenamiento (68 %), validación (12 %) y prueba (20 %).

5. **Normalización de las variables**: Se aplicó el mismo proceso de normalización que en 4.4.3

6. **Reducción de dimensionalidad con PCA**: Se utilizó un análisis de componentes principales (PCA) sobre las *features*, reduciendo el número de dimensiones a 4500 componentes. Estos componentes retuvieron el 97 % de la varianza original del conjunto de datos.

En todas las versiones del preprocesamiento, se abordó el problema como un problema de *forecasting* (pronóstico), donde el objetivo es predecir el valor futuro de una variable basándose en valores observados en el pasado. Debido a este enfoque, se decidió conservar el orden de las filas y construir columnas temporales para las *features*, lo que permitió capturar la evolución temporal de las variables y mejorar las predicciones basadas en series de tiempo.

4.4.6. Modelo de aprendizaje automático - MLP

Se implementó una red neuronal de tipo MLP (Perceptrón Multicapa) con las siguientes características, resumidas en la Tabla 4.28:

Características	Descripción
Capa de entrada	Todas las *features* de cada muestra
Capas ocultas	4 capas ocultas, tamaños variables desde 128, 64, 32 y 16 nodos hasta 1024, 512, 256 y 128 nodos
Funciones de activación	ReLU (usualmente)
Dropout	Se aplicaron tasas de *dropout* para evitar el sobreajuste
Batch Normalization	Se normalizaron las activaciones de cada capa oculta
Salidas	2 salidas: una para regresión (3 nodos) y una para clasificación binaria

Tabla 4.28: Resumen de las características del modelo MLP

El modelo MLP fue entrenado utilizando el optimizador AdamW, con una tasa de aprendizaje inicial de 1×10^{-4} y un decaimiento de peso de 1×10^{-5}. La tasa de aprendizaje fue ajustada mediante un esquema escalonado, reduciéndose en un factor de 0.5 cada 20 etapas de entrenamiento.

Para la clasificación binaria, se utilizó una función de pérdida de entropía cruzada binaria, mientras que para la regresión se empleó la función de pérdida de error cuadrático medio. Además, se aplicó *early stopping* con un criterio de paciencia definido antes del entrenamiento.

La evaluación del modelo se llevó a cabo calculando el coeficiente R^2 para las métricas continuas y los valores de precisión y F1 para la clasificación binaria. Finalmente, se realizaron gráficos para visualizar las pérdidas de entrenamiento y validación, los valores reales frente a los predichos, y la distribución de errores y residuos.

4.4.7. Optimización de hiperparámetros

Para la optimización de hiperparámetros, se utilizó la librería *Optuna*, una herramienta diseñada para automatizar y acelerar este proceso. Los hiperparámetros definidos para la optimización fueron los siguientes:

Hiperparámetro	Descripción
layer_sizes	Tamaños de las capas ocultas del modelo.
dropout_rates	Tasas de *dropout* aplicadas para cada capa oculta, con el objetivo de evitar sobreajuste.
activation_funcs	Funciones de activación utilizadas en las capas ocultas (e.g., ReLU, *elu*, *selu*).
lr	Tasa de aprendizaje utilizada por el optimizador.
weight_decay	Decaimiento de peso aplicado para regularización.
step_size	Tamaño del paso para el *scheduler*, que define cuándo reducir la tasa de aprendizaje.
gamma	Factor de reducción de la tasa de aprendizaje aplicado por el *scheduler*.
optimizer_choice	Elección del optimizador utilizado (e.g., Adam, RMSProp).
batch_size	Tamaño del lote de datos (*batch*) utilizado en el entrenamiento.

Tabla 4.29: Hiperparámetros optimizados durante el proceso de optimización

Para realizar la búsqueda del mejor conjunto de hiperparámetros, se llevaron a cabo un total de 500 pruebas. A fin de acelerar el proceso de optimización, se implementó un *pruner*, que eliminó las pruebas no prometedoras después de 250 pruebas. Este mecanismo compara el mejor resultado de una prueba con la mediana de los resultados anteriores en el mismo paso. Si el resultado de la prueba actual es peor que la mediana, se descarta la prueba, reduciendo así el tiempo total de la optimización.

El objetivo principal del modelo era optimizar tanto la clasificación como la regresión, priorizando el coeficiente R^2 debido a las dificultades que presentaba el modelo en la predicción de las métricas continuas. Para lograr esto, se definió

la siguiente función objetivo:

$$objetivo = 0.3 \times f1 + 0.7 \times r2$$

Dado que el proceso de optimización es costoso en términos de tiempo y recursos, se decidió realizar la optimización únicamente sobre la primera versión del preprocesamiento de los datos 4.4.3.

Resultados de la optimización Después de 500 pruebas, se encontró el siguiente conjunto de hiperparámetros como el que maximiza la función objetivo en el conjunto de validación:

Hiperparámetro	Valor óptimo
layer_sizes	[64, 32, 16, 8]
dropout_rates	[0.053, 0.44, 0.30, 0.35]
activation_funcs	[*relu*, *elu*, *selu*, *leaky_relu*]
lr	0.0398
weight_decay	0.0017
step_size	23
gamma	0.857
optimizer_choice	Adam
batch_size	2048

Tabla 4.30: Resultados del conjunto óptimo de hiperparámetros encontrados

Con este conjunto de hiperparámetros, se entrenó el modelo para evaluar su desempeño sobre el conjunto de prueba.

Resultados del modelo Los resultados obtenidos en el conjunto de prueba fueron los siguientes:

Métrica	Valor obtenido
R2s	OSNR: -0.042, SNR: -0.041, BER: -0.009
Precisión	0.75
F1	0.86

Tabla 4.31: Resultados de desempeño del modelo en el conjunto de prueba

Como se puede observar en la Tabla 4.31, los resultados del modelo continúan siendo deficientes en lo que respecta a la predicción de las métricas continuas, obteniendo valores negativos de R^2. Aunque la precisión y el *F1* para la clasificación binaria fueron aceptables, el modelo no logró capturar correctamente las relaciones subyacentes en las métricas continuas como OSNR, SNR y BER.

4.4.8. Ingeniería de *features* aplicada al conjunto de datos basado en el estado de la red

Debido a los malos resultados que obtuvimos con las metodologías anteriores sobre este conjunto de datos tomamos un un enfoque diferente. En las tres versiones anteriores se pensó al conjunto de datos como un formato de series temporales y aunque cada uno de las tres presentó particularidades el enfoque fue parecido.

Repasando la representación de datos que usamos anteriormente, se podía observar que cada columna de la matriz de *features* representaba un recurso de la red (en este caso, un *slot* de frecuencia en un enlace óptico) y que dentro de esa columna se encontraba la matriz de características del *lightpath* que pasaba por ese recurso en ese momento. Cada fila de la matriz de características representaba el momento en el que se estableció un nuevo lightpath. Por lo tanto, si en la fila i de la matriz de características había un *lightpath* que pasaba por los recursos r_1, r_2, r_3, r_4 y r_5, entonces la misma matriz de características se repetía en las columnas r_1, r_2, r_3, r_4 y r_5. Por más que es una manera muy intuitiva de verlo, hace que el modelo tenga que aprender a identificar patrones en las columnas que contienen a la misma matriz de características y que no aportan información nueva.

Uno de los problemas que generó fue la gran cantidad de *features* que se utilizaron. Creímos que ninguno de los modelos que construímos logró capturar las relaciones entre los datos ni procesar la cantidad de información que había. A su vez todas las representaciones que construímos tenían muchos valores nulos debido a la naturaleza de los datos (cada recurso no ocupado tenía valores nulos en todas las columnas). Se decidió un enfoque diferente donde se utilizaron agregadores estadísticos para reducir la cantidad de *features*.

Para evitar repetir el mismo enfoque, procedimos de la siguiente manera para cada uno de los 198 enlaces de la red:

1. Seleccionamos las siguientes *features* asociadas a los *lightpaths* activos en el enlace:
   - Cardinalidad del formato de modulación,
   - Frecuencia,
   - Longitud del *lightpath*,
   - Cantidad de tramos del *lightpath*.
2. Para cada una de las *features* seleccionadas, calculamos los siguientes valores estadísticos descriptivos: media, desviación estándar y valor máximo de cada *feature* de los *lightpaths* activos en el enlace correspondiente.

   Por ejemplo, si en el enlace i hay tres bandas de frecuencia ocupadas por tres *lightpaths* activos y noventa y seas bandas de frecuencia desocupadas, la media de una *feature* j se calcula de la siguiente forma:

$$\text{feature\_i\_mean} = \frac{\text{feature\_i\_lp\_1} + \text{feature\_i\_lp\_2} + \text{feature\_i\_lp\_3}}{3}$$

sin tener en cuenta el resto de banda de frecuencias desocupadas.

De manera similar, calculamos la desviación estándar y el valor máximo de cada *feature* para los *lightpaths* activos en ese enlace.

3. Este procedimiento se repitió para todas las *features* seleccionadas en cada uno de los 198 enlaces de la red.

Para seleccionar las *features* consideramos 15 *features*: identificador de la conexión, tasa de datos de la conexión, cardinalidad del formato de modulación, OSNR, SNR, BER, grado del nodo fuente , grado del nodo destino, longitud del *lightpath*, cantidad de tramos, número de enlaces, longitud del enlace más corto, longitud del enlace más largo, frecuencia y tasa de datos del *lightpath*.

- Eliminamos el identificador de la conexión porque no aporta información relevante.
- Eliminamos OSNR, SNR y BER de cada enlace en particular ya que se agregan después de otra manera.
- Removimos grado del nodo fuente y grado del nodo destino porque al estar dentro del enlace se repiten para todos los *lightpaths* y la noción de estar en el mismo enlace ya está integrada en la naturaleza de los datos.
- De longitud del *lightpath*, cantidad de tramos, número de enlaces, longitud del enlace más corto y longitud del enlace más largo, se seleccionaron longitud del *lightpath* y cantidad de tramos que consideramos debido a los anteriores análisis que representaban a las *features* restantes.
- De análisis anteriores se observó que de las restantes *features* que quedan cardinalidad del formato de modulación y frecuencia son las más importantes, por lo que se decidió que cubrían la información necesaria.

Por lo tanto, se terminaron seleccionando cuatro *features* para cada *lightpath* activo en cada link: cardinalidad del formato de modulación, frecuencia, longitud del *lightpath* y cantidad de tramos.

El flujo de preprocesamiento que implementamos fue el siguiente:

1. Por cada enlace de los 198:
   a) Seleccionamos las *features* cardinalidad del formato de modulación, frecuencia, longitud del *lightpath* y cantidad de tramos de los *lightpaths* activos en ese link.
   b) Calculamos la media, desviación estándar y máximo de cada *feature* seleccionada y agragamos esas columnas al conjunto de datos.
   c) Agregamos una columna que indica la cantidad de *lightpaths* activos en ese link.
   d) Agregamos una columna que indica si el *lightpath* que se establece en esa muestra está activo en ese link.

2. Agregamos las columnas clase, OSNR, SNR y BER que son las métricas del LUT que se quieren predecir.

3. Agregamos una columna que indica la cantidad de *lightpaths* activos en la red en esa muestra.

4. Agregamos una columna que indica el porcentaje de ocupación de la red en esa muestra.

5. Calculamos la media, desviación estándar y máximo del OSNR, SNR y BER de todos los *lightpaths* activos en la red y agregamos esas columnas al conjunto de datos.

6. Calculamos la media, desviación estándar y máximo de longitud del *lightpath* de todos los *lightpaths* activos en la red y agregamos esas columnas al conjunto de datos.

7. Para la cardinalidad del formato de modulación de todos los *lightpaths* activos calculamos cuántos tienen una cardinalidad de 4, 8, 16, 32 y 64 y agregamos las columnas mod_order_4 que indica cuántos *lightpaths* tienen una cardinalidad de 4, y mod_order_4.0_percentage que indica el porcentaje de *lightpaths* activos que tienen una cardinalidad de 4.

La salida del flujo dio un total de 2796 *features* que fue la entrada del modelo de aprendizaje automático. Debido a que el proceso del flujo de datos es costoso en términos de tiempo y recursos, realizamos el flujo para las primeras 914.850 muestras de la red y guardamos el conjunto de datos preprocesado en 15 fragmentos de 60.990 muestras cada uno.

4.4.9. Metodologías aplicadas sobre el conjunto de datos luego de la ingeniería de *features*

Una vez que tuvimos el conjunto de datos procesado y guardado luego de la ingeniería de features seguimos un proceso muy similar a anteriores construcciones de métodos estadísticos:

- Se hizo un análisis exploratorio de datos para entender la distribución de las *features* y como se relacionan con las métricas que se quieren predecir. Los resultados se omiten porque no arrojaron información relevante.

- Utilizamos un algoritmo básico de boosting que ya fue utilizado (*HistGradientBoostingRegressor*) para ver cómo se comporta en el conjunto de datos. A su vez, hicimos un análisis de SHAP para entender qué *features* son más importantes para el modelo.

- Utilizamos un modelo de redes neuronales con la misma arquitectura que el MLP utilizado anteriormente pero con las *features* generadas en el flujo de selección de características.

4.5. Resultados sobre el conjunto de datos basado en el estado de la red

Luego de los cuatro preprocesamientos distintos hechos en 4.4.3, 4.4.4, 4.4.5 y 4.4.8 se evaluaron los modelos de aprendizaje automático usando los datos generados como entrada.

4.5.1. Evaluación sobre la primera versión de los datos

El modelo MLP se entrenó con los siguientes hiperparámetros, resumidos en la Tabla 4.32 y se entrenó y evaluó sobre el conjunto de datos generado en 4.4.3.

Hiperparámetro	Valor
batch_size	256
layer_sizes	[256, 128, 64, 32]
dropout_rates	[0, 0.2, 0.2, 0.4]
activation_funcs	[*F.relu*, *F.relu*, *F.relu*, *F.relu*]
lr	1×10^{-3}
weight_decay	1×10^{-5}
step_size	10
gamma	0.5
optimizer_choice	AdamW

Tabla 4.32: Hiperparámetros del modelo MLP - Primera versión

Implementamos un esquema de *early stopping* con una paciencia de 20 *epochs*, es decir, si después de 20 *epochs* no se observaba mejora en la pérdida de validación, el entrenamiento se detenía automáticamente. Aunque se permitió que el modelo se entrenara hasta 700 *epochs*, el entrenamiento se detuvo luego de 24 *epochs* debido a la falta de mejora en la validación.

Resultados del modelo Los resultados obtenidos en el conjunto de prueba fueron los siguientes:

Métrica	Valor obtenido
R2	OSNR: -0.2888, SNR: -0.2888, BER: -0.1155
Precisión	0.73
F1	0.84

Tabla 4.33: Resultados de desempeño del modelo MLP - Primera versión

Como se puede observar en la Tabla 4.33, los resultados no son satisfactorios en cuanto a las métricas continuas, ya que los valores de R^2 para OSNR, SNR y BER son negativos, lo que indica que el modelo no logró capturar correctamente las relaciones subyacentes entre las características y los valores objetivo.

Por otro lado, aunque la clasificación binaria obtuvo una precisión del 73 % y un *F1* de 0.84, es importante resaltar que el conjunto de datos tiene una distribución desbalanceada, con un 74.7 % de muestras correspondientes a la clase positiva y un 25.3 % a la clase negativa. Este desbalance indica que el modelo tiene un desempeño peor que un modelo que simplemente prediga siempre la clase mayoritaria.

Análisis gráfico de los resultados Las siguientes figuras muestran los valores predichos versus los valores reales para las métricas continuas.

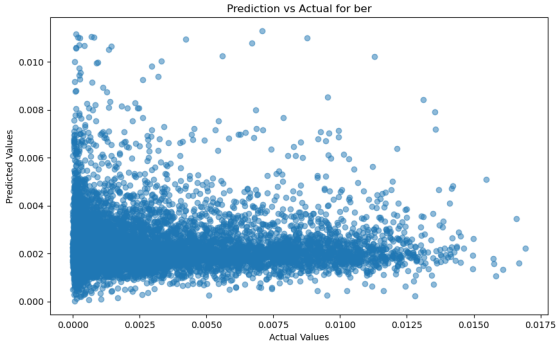

Figura 4.28: Valores reales vs. predichos para BER

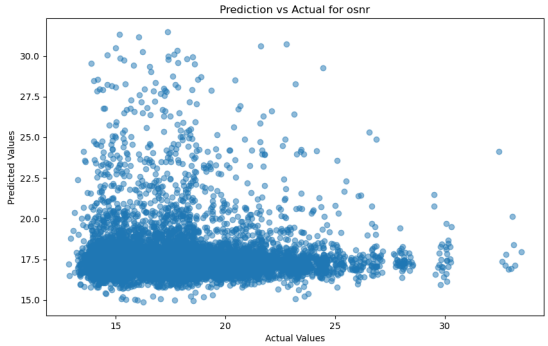

Figura 4.29: Valores reales vs. predichos para OSNR

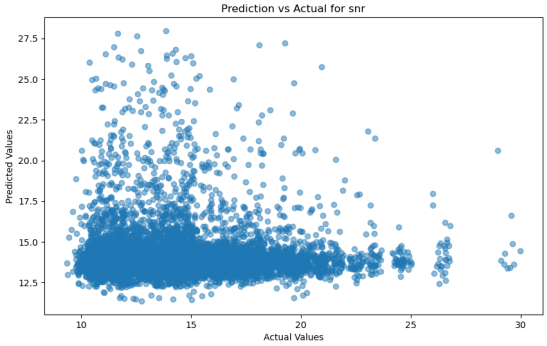

Figura 4.30: Valores reales vs. predichos para SNR

Como se observa en las Figuras 4.28, 4.29 y 4.30, los valores predichos no se alinean con los valores reales, lo que refleja la incapacidad del modelo para predecir de manera precisa las métricas continuas de calidad de transmisión.

Finalmente, en la Figura 4.31 se visualizan las pérdidas de entrenamiento y validación durante el entrenamiento del modelo. Se puede observar que, aunque las pérdidas de entrenamiento disminuyen como se esperaba, las pérdidas de validación comienzan a aumentar, lo que indica un sobreajuste del modelo al conjunto de entrenamiento y su poca capacidad de generalizar.

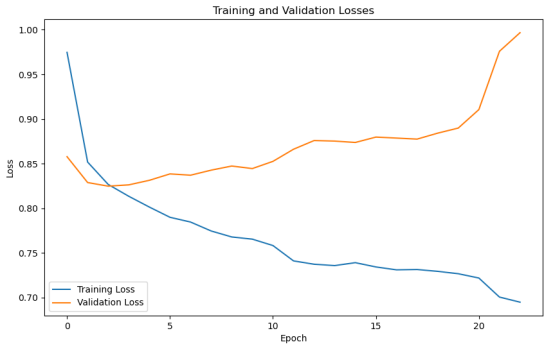

Figura 4.31: Pérdidas de entrenamiento y validación durante el entrenamiento del modelo

Conclusiones Los resultados indican que el modelo no logró predecir adecuadamente las métricas continuas (OSNR, SNR, BER) y, aunque el desempeño en la clasificación binaria fue aceptable, este desempeño es inferior al esperado debido al desbalance de clases. Se puede concluir que el modelo necesita mejoras significativas para ser útil en este contexto, incluyendo la posibilidad de ajustar

los hiperparámetros o explorar arquitecturas alternativas.

4.5.2. Evaluación sobre la segunda versión de los datos

Para poder comparar los resultados con la primera versión, se entrenó nuevamente el modelo MLP utilizando los mismos hiperparámetros, pero sobre el conjunto de datos generado en 4.4.4.

Los resultados obtenidos en el conjunto de prueba se presentan en la Tabla 4.34.

Métrica	Valor obtenido
R2	OSNR: -1.2093, SNR: -1.2092, BER: -0.2001
Precisión	0.71
F1	0.83

Tabla 4.34: Resultados de desempeño del modelo MLP sobre la segunda versión de los datos

Se puede ver que los resultados son peores que en la primera versión. Las gráficas de los valores reales en función de valores predichos y de las pérdidas de entrenamiento y validación son muy similares a las de la primera versión y por ello se omiten.

4.5.3. Evaluación sobre la tercera versión de los datos

La tercera versión del modelo MLP se entrenó con los mismos hiperparámetros sobre el conjunto de datos procesado en 4.4.5 y los resultados se resumen en la Tabla 4.35.

Métrica	Valor obtenido
R2	OSNR: -0.2789, SNR: -0.2789, BER: -1.3624
Precisión (clasificación binaria)	0.56
F1	0.68

Tabla 4.35: Resultados de desempeño del modelo MLP sobre la tercera versión de los datos

Como se observa en la Tabla 4.35, los resultados obtenidos son similares a los de las versiones anteriores en cuanto a las métricas continuas y un descenso en la precisión y F1. A continuación se muestran los gráficos de los valores reales en función de valores predichos y de las pérdidas de entrenamiento y validación que son similares a las versiones anteriores.

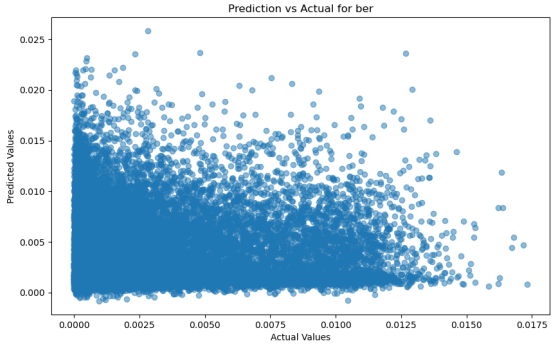

Figura 4.32: Valores reales en función de valores predichos para BER

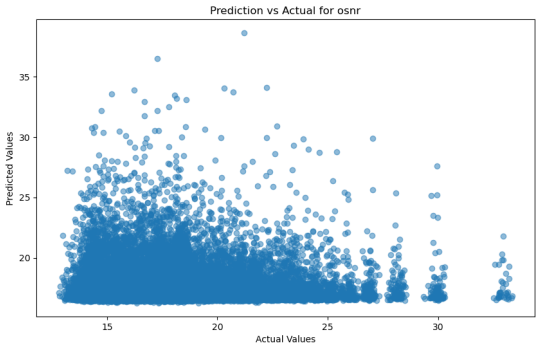

Figura 4.33: Valores reales en función de valores predichos para OSNR

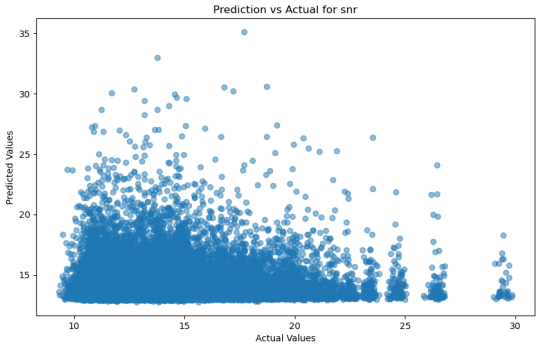

Figura 4.34: Valores reales en función de valores predichos para SNR

Figura 4.35: Pérdidas de entrenamiento y validación

4.5.4. Resultados del modelo de *boosting* sobre el conjunto de datos al que se le aplicó ingeniería de *features*

Con la misma premisa de los modelos construídos sobre la anterior representación de datos quisimos utilizar un modelo interpretable una vez depurado el conjunto de datos. Por eso elegimos *HistGradientBoostingRegressor*. Separamos al conjunto de datos en 70 % entrenamiento y 30 % para evaluación. Se utilizó el modelo con los hiperparámetros por defecto y se normalizaron las *features* eliminando la media y escalando a la varianza. Se obtuvieron los siguientes resultados que se observan en la Tabla 4.36.

Métrica	**Valor**
F1 para la clase	0.88
R2 para OSNR	0.96
R2 para SNR	0.96
R2 para BER	0.16

Tabla 4.36: Resultados de desempeño del modelo

Se obtuvieron buenos resultados para la clasificación binaria y para las métricas continuas. De todas maneras, se puede ver que el modelo no es capaz de predecir correctamente la métrica BER. Esto se puede deber a que la métrica BER tiene una distribución muy sesgada hacia valores muy pequeños y el modelo no es capaz de predecir correctamente estos valores.

A continuación en las Figuras 4.36, 4.37 y 4.38 se puede ver las gráficas de los valores reales en función de valores predichos para las métricas continuas.

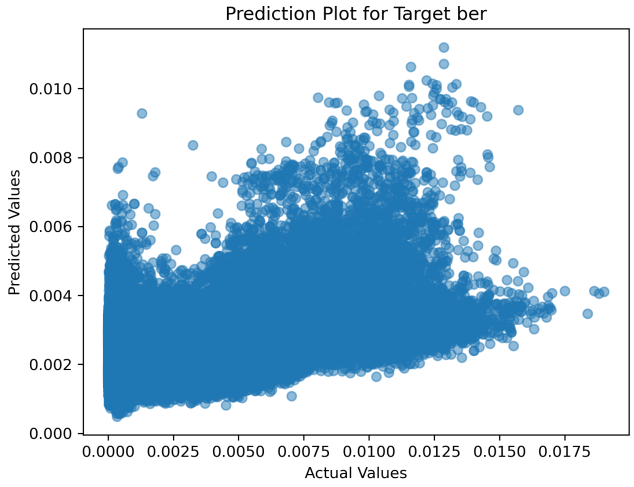

Figura 4.36: Valores reales en función de valores predichos para BER

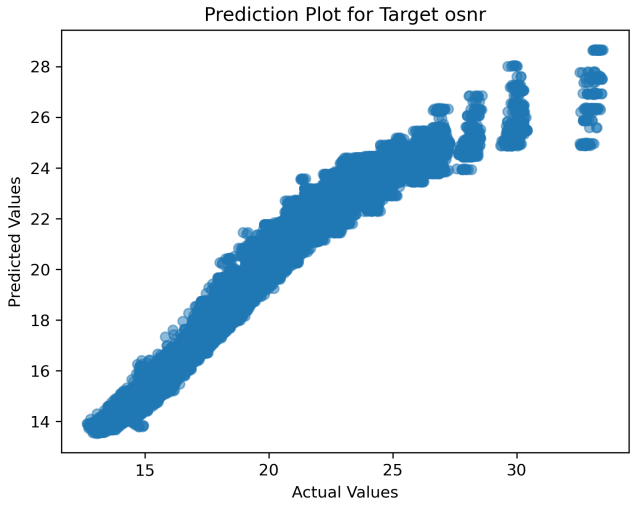

Figura 4.37: Valores reales en función de valores predichos para OSNR

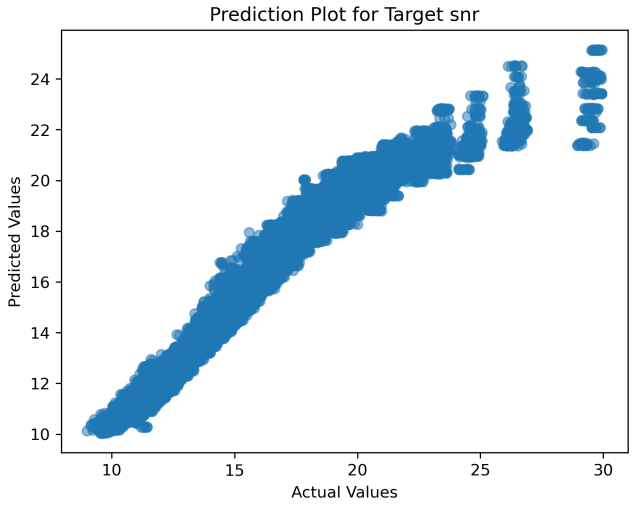

Figura 4.38: Valores reales en función de valores predichos para SNR

De la misma manera que se hizo en el conjunto de datos basado en *lightpaths*, se utilizó SHAP para entender qué *features* son más importantes para el modelo. Se puede ver en las Figuras 4.39 4.40, 4.41 y 4.42 las *features* más importantes para cada métrica.

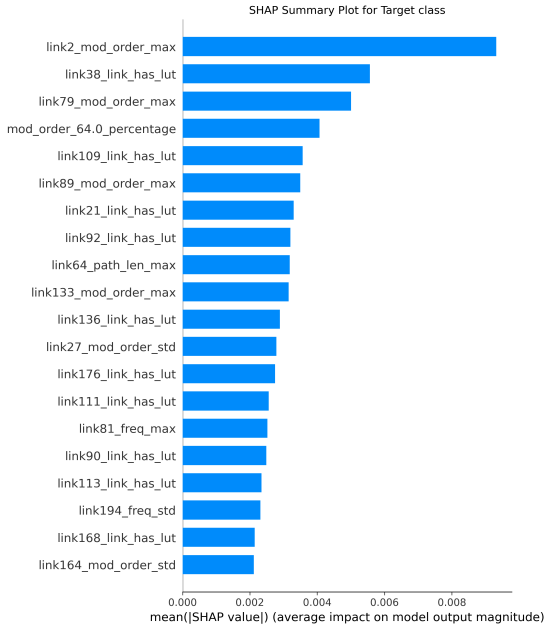

Figura 4.39: SHAP para la clase

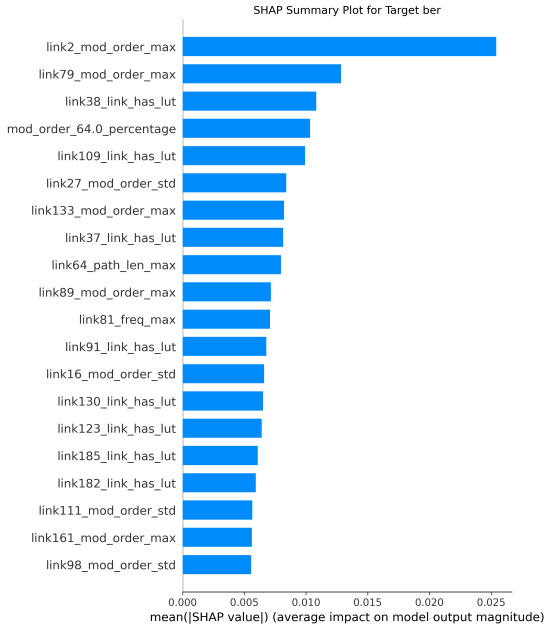

Figura 4.40: SHAP para BER

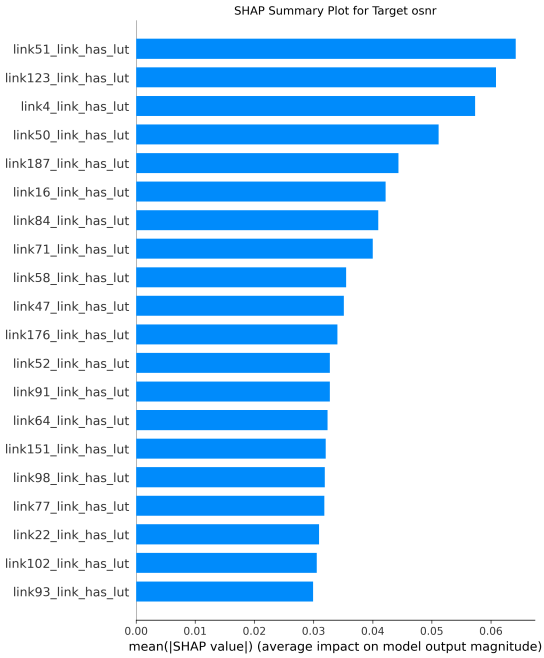

Figura 4.41: SHAP para OSNR

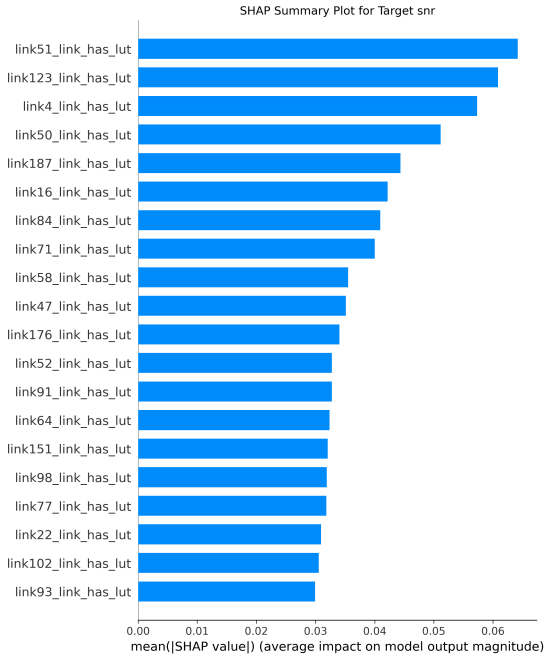

Figura 4.42: SHAP para SNR

4.5.5. Resultados del modelo de redes neuronales sobre el conjunto de datos al que se le aplicó ingeniería de *features*

Utilizamos un modelo de MLP al igual que en las versiones anteriores pero con el conjunto de datos generado en 4.4.8. Separamos a los datos en 70 % entrenamiento, 15 % validación y 15 % prueba. Estandarizamos los datos eliminando la media y escalando a la varianza. A su vez, redujumos la dimensionalidad del conjunto de datos utilizando PCA con 837 componentes que mantienen el 95 % de la varianza.

Los resultados se muestran en la Tabla 4.37.

Métrica	Valor
R2 para OSNR	0.98
R2 para SNR	0.98
R2 para BER	0.27
Precisión	0.81
F1	0.88

Tabla 4.37: Resultados del modelo MLP

Se obtuvieron buenos resultados para las métricas continuas y para la clasi-

ficación de la clase. De la misma manera que el modelo anterior de *boosting*, se puede ver que el modelo no es capaz de predecir correctamente la métrica BER.

A continuación en las Figuras 4.43, 4.44 y 4.45 se puede ver las gráficas de los valores reales en función de valores predichos para las métricas continuas.

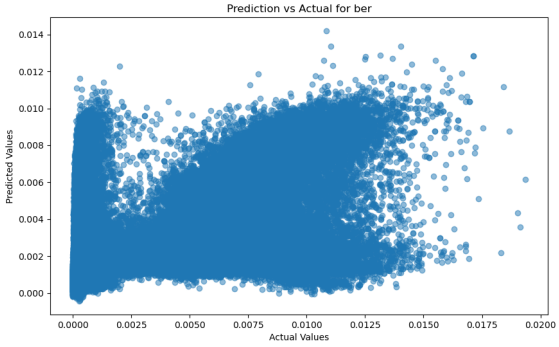

Figura 4.43: Valores reales en función de valores predichos para BER

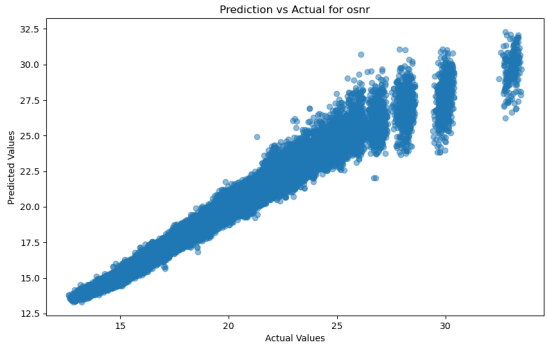

Figura 4.44: Valores reales en función de valores predichos para OSNR

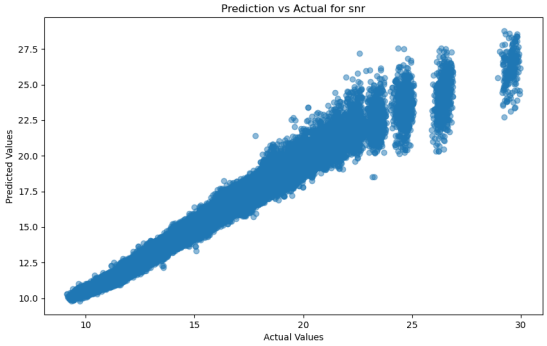

Figura 4.45: Valores reales en función de valores predichos para SNR

4.5.6. Herramientas utilizadas

En la Tabla 4.38 se resumen las herramientas utilizadas en esta etapa del proyecto de grado. Debido a las necesidades de hardware utilizamos el ClusterUY (Nesmachnow y Iturriaga, 2019) para realizar los cálculos necesarios. El código fuente de los modelos de aprendizaje automático se encuentra en el repositorio de Gitlab (Facultad de Ingeniería - MINA, s.f.-b).

Etapa	Procesador	Memoria RAM utilizada	GPU	Tiempo de ejecución	Número de procesos
Transformación de los datos	Intel Xeon Gold 6138	72 GB	-	80.2 horas	12
Preprocesamiento de los datos (1)	Intel Xeon Gold 6138	128 GB	-	10 horas	1
Preprocesamiento de los datos (2)	Intel Xeon Gold 6138	128 GB	-	9 horas	1
Preprocesamiento de los datos (3)	Intel Xeon Gold 6138	128 GB	-	9 horas y 20 minutos	1
Optimización del entrenamiento (1)	Intel Xeon Gold 6138	20 GB	2 x NVIDIA P100	15 horas	1
Ingeniería de *features*	Intel Xeon Gold 6138	100 GB	-	41 horas	1
Análisis exploratorio de datos	Intel Xeon Gold 6138	70 GB	-	11 horas	1
Entrenamiento MLP	AMD EPYC 7763	16 GB	2 x NVIDIA A40	2 horas y 30 minutos	1

Tabla 4.38: Herramientas utilizadas en las distintas etapas del proyecto

4.6. Conclusiones

En esta etapa del proyecto, con el objetivo de estimar la QoT de una red de fibra óptica a partir de un conjunto de datos establecido, generamos un proceso de trabajo entero sobre los datos. El trabajo incluyó análisis exploratorio de datos, construcción y refinamiento de modelos de aprendizaje automático y estudio de los resultados obtenidos.

Se trabajó con la meta de mejorar los métodos estadísticos presentados en (Bergk y cols., 2022) y (Safari y cols., 2021). Para el conjunto de datos basados en *lightpaths* se lograron resultados buenos y comparables con el trabajo existente. Para el conjunto de datos basados en estados de la red se generó una base de conocimiento solida sobre las herramientas a utilizar para gestionar grandes volúmenes de datos y se generaron ANNs para estimar las variables continuas que pueden ser tomadas en futuros trabajos como punto de comparación.

Capítulo 5

Conclusiones sobre el proyecto

En el proyecto de grado buscamos construir un flujo de trabajo entero para resolver algún problema inherente a las complejidades de las fibras ópticas. Nos planteamos resolver la obtención, almacenamiento y análisis de los datos. También la curación y transformación de los datos para elaborar un método estadístico de aprendizaje automático que nos sirva como herramienta ante la toma de decisiones.

El trabajo se marcó en dos etapas distintas. La primera etapa consistió en la creación de un flujo de datos para consumir de la red de fibra óptica de Antel. Nos propusimos obtener datos de distintas fuentes con la idea de crear nuestro propio modelo de datos para generar herramientas beneficiosas para Antel. Logramos construir un modelo parcial de datos que consumió de una interfaz que proveyó Antel. A su vez, creamos herramientas gráficas para analizar los datos almacenados en nuestra base de datos.

La segunda etapa del proyecto se encapsuló en analizar y crear modelos de aprendizaje automático una vez que se cuenta con un conjunto de datos de una red de fibra óptica. Conseguimos resultados comparables a los trabajos previamente hechos sobre el mismo conjunto de datos y establecimos modelos base para poder comparar futuros trabajos.

Ambas etapas junto al estudio previo del marco teórico dieron un basto conocimiento en el área de redes ópticas y en el proceso de construcción y evaluación de métodos estadísticos para resolver alguno de los desafíos que presentan las redes ópticas actuales.

5.1. Trabajo futuro

Los resultados obtenidos presentan oportunidades de mejora. En la primera etapa del proyecto, no tuvimos acceso a todas las fuentes de datos posibles y generamos un flujo con un solo origen de datos. Consumir información de

múltiples fuentes, obtener los datos de manera dinámica y generar un conjunto de datos similar al utilizado en la segunda etapa podría resultar muy valioso para generar herramientas que contribuyan a la toma de decisiones en una empresa como Antel.

Así como se lograron resultados y conclusiones significativas utilizando el conjunto de datos de Fraunhofer HHI, es esperable obtener resultados similares si se dispone de la información necesaria proveniente de Antel.

En relación con las metodologías y resultados obtenidos en los modelos de aprendizaje automático, es posible mejorar las herramientas utilizadas para gestionar el gran volumen de datos. En varias etapas del proyecto, nos vimos limitados por el tiempo y los recursos de hardware, lo que nos permitió realizar un análisis exhaustivo solo sobre uno de los cuatro conjuntos de datos en el caso de estudio de HHI. Replicar el trabajo en los restantes conjuntos y potenciar las distintas etapas mediante el uso de herramientas especializadas podría mejorar significativamente los resultados.

Sería de gran interés continuar trabajando con conjuntos de datos basados en el estado de la red, aprovechando las características topológicas de los datos y explorando distintas metodologías. El conjunto de datos posee un carácter temporal y secuencial que no se pudo aprovechar para mejorar el entrenamiento de los modelos. Investigar otros enfoques, como redes neuronales basadas en grafos, modelos de series temporales o una combinación de ambos, podría ser interesante para evaluar el comportamiento de estos métodos.

Referencias

Akiba, T., Sano, S., Yanase, T., Ohta, T., y Koyama, M. (2019). *Optuna: A next-generation hyperparameter optimization framework.* https://optuna.org/. (Accessed: 2024-11-21)

Aladin, S., Tran, A. V. S., Allogba, S., y Tremblay, C. (2020). Quality of transmission estimation and short-term performance forecast of lightpaths. *Journal of Lightwave Technology*. Descargado de https://ieeexplore.ieee.org/abstract/document/9004503/ doi: 10.1109/JLT.2020.9004503

Allogba, S., Aladin, S., y Tremblay, C. (2022). Machine-learning-based lightpath qot estimation and forecasting. *Journal of Lightwave Technology*, *40*(10), 3115–3123. Descargado de https://opg.optica.org/abstract.cfm?uri=jlt-40-10-3115 doi: 10.1109/JLT.2022.3160379

Bayer, M. (2012). *Sqlalchemy documentation.* https://docs.sqlalchemy.org/. (Accessed: 2024-11-21)

Bergk, G., Shariati, B., Safari, P., y Fischer, J. K. (2022). Ml-assisted qot estimation: a dataset collection and data visualization for dataset quality evaluation. *Journal of Optical Communications and Networking*, *14*(3), 43–50.

Bergstra, J., Bardenet, R., Bengio, Y., y Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems*, *24*, 2546–2554.

Bianchi, I., y Donnangelo, F. (2022). *Ciencia de datos en el dominio de las redes ópticas.* https://hdl.handle.net/20.500.12008/30947. (Tesis de grado, Universidad de la República, Facultad de Ingeniería, Instituto de Computación, Montevideo. También disponible en: https://repositorioslatinoamericanos.uchile.cl/handle/2250/4984845)

Bishop, C. M. (2006). *Pattern recognition and machine learning.* New York: Springer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. doi: 10.1007/BF00058655

Calvo, D. (2024). *Apache Kafka.* Descargado de https://www.diegocalvo.es/kafka/ (Accessed: 2024-11-30)

Charlet, G. (2008). Coherent detection associated with digital signal processing for fiber optics communication. *Comptes Rendus Physique*, *9*(8), 1012–1030. (Available online 4 December 2008)

Corporation, C. (2024). *What is optical transport networking (otn)?* https://www.ciena.com/insights/what-is/What-is-Optical-Transport-Networking-OTN.html. (Accessed: 2024-11-23)

Facultad de Ingeniería - MINA. (s.f.-a). *ANTEL Optical Network.* https://gitlab.com/fing-mina/optical-networks/antel-optical-network. (Accessed: 2024-11-30)

Facultad de Ingeniería - MINA. (s.f.-b). *HHI Optical Network.* https://gitlab.com/fing-mina/optical-networks/hhi-optical-network. (Accessed: 2024-11-30)

FiberOptics4Sale Blog. (2024). *What is wavelength selective switch (wss)?* https://www.fiberoptics4sale.com/blogs/archive-posts/95046534-what-is-wavelength-selective-switch-wss. (Accessed: 2024-08-24)

Fraunhofer HHI. (2024). *Qot dataset collection.* https://www.hhi.fraunhofer.de/en/departments/pn/communication-and-measurement-instruments/qot-dataset-collection.html. (Accessed: 2024-08-24)

Freund, Y., Schapire, R., y Abe, N. (1999). A short introduction to boosting. *Journal of the Japanese Society for Artificial Intelligence*. Descargado de http://www.yorku.ca/gisweb/eats4400/boost.pdf

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*(5), 1189–1232.

García, R., Pirri, M., Castro, A., y Grampín, E. (2023). *Optimización, monitorización y análisis de datos en la red Óptica de antel, entregable 1.* Universidad de la República, 31 de marzo de 2023.

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow* (2nd ed.). Sebastopol, CA: O'Reilly Media.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... Oliphant, T. E. (2020, septiembre). Array programming with NumPy. *Nature*, *585*(7825), 357–362. Descargado de https://doi.org/10.1038/s41586-020-2649-2 doi: 10.1038/s41586-020-2649-2

Hunter, y D., J. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

ITU-T. (2015). Characteristics of optical transport network functionality. *Recommendation G.798*. https://www.itu.int/rec/T-REC-G.798/en. (Accessed: 2024-11-23)

ITU-T. (2016). Interfaces for the optical transport network (otn). *Recommendation G.709*. https://www.itu.int/rec/T-REC-G.709/en. (Accessed: 2024-11-23)

ITU-T. (2020). *Spectral grids for wdm applications: Dwdm frequency grid.* https://www.itu.int/rec/T-REC-G.694.1-202010-I/en. (Recommendation ITU-T G.694.1)

Jurafsky, D., y Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* Upper Saddle River, N.J.: Pearson Pren-

tice Hall. (Descargado de http://www.amazon.com/Speech-Language-Processing-2nd-Edition/dp/0131873210/ref=pd_bxgy_b_img_y)

Kreps, J., Narkhede, N., y Rao, J. (2011). Kafka: a Distributed Messaging System for Log Processing. *Proceedings of the NetDB*, 1–7.

Lehmen, A. V., Doverspike, R., y Clapp, G. (2015). Coronet: Testbeds, demonstration, and lessons learned. *IEEE Journal of Optical Communications and Networking*. Descargado de https://ieeexplore.ieee.org/abstract/document/7063688/ doi: 10.1109/JOCN.2015.7063688

Lundberg, S. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*. Descargado de https://arxiv.org/abs/1705.07874

Lundberg, S. M., y Lee, S.-I. (2017). A unified approach to interpreting model predictions. En I. Guyon y cols. (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. Descargado de http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

López, V., y Velasco, L. (2016). *Elastic optical networks*. Cham, Switzerland: Springer.

Mannie, E. (2004, octubre). *Generalized Multi-Protocol Label Switching (GMPLS) Architecture.* RFC 3945. (Available: https://www.rfc-editor.org/info/rfc3945)

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

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... Turner, J. (2008). OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, *38*(2), 69–74. doi: 10.1145/1355734.1355746

McKinney, W. (2010). Data structures for statistical computing in python. En S. van der Walt y J. Millman (Eds.), *Proceedings of the 9th python in science conference* (pp. 56–61). doi: 10.25080/Majora-92bf1922-00a

Mitchell, T. M. (1997). *Machine learning* (Vol. 1) (n.º 9). New York: McGraw-Hill.

Monarch Network Architects. (2020). *Darpa core optical networks (coronet) continental united states (conus) topology.* http://monarchna.com/CORONET_CONUS_Topology.xls. (Accessed: 2024-08-17)

Multilayer perceptron. (s.f.). https://www.researchgate.net/figure/Multi-Layer-Perceptron-MLP-diagram-with-four-hidden-layers-and-a-collection-of-single_fig1_334609713. (Accessed: 2024-08-17)

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

Nesmachnow, S., y Iturriaga, S. (2019). Cluster-uy: Collaborative scientific high performance computing in uruguay. En M. Torres y J. Klapp (Eds.), *Supercomputing. isum 2019. communications in computer and information science* (Vol. 1151). Springer, Cham.

Norouzi, A., Zaim, A. H., y Ustundag, B. B. (2011). An integrated survey in optical networks: Concepts, components and problems. *International Journal of Computer Science and Network Security (IJCSNS)*, *11*(1), 10–15.

Open Networking Foundation (ONF). (2012). *Software-defined networking: The new norm for networks.* https://www.opennetworking.org/sdn-resources/sdn-library/whitepapers. (Accessed: 2014-06-14)

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. En *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Descargado de http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Patri, S. K., Autenrieth, A., Rafique, D., Elbers, J.-P., y Machuca, C. M. (2020). Hecson: Heuristic for configuration selection in optical network planning. En *Optical fiber communication conference* (pp. Th2A–32). Optical Society of America.

Poggiolini, P., Bosco, G., Carena, A., Curri, V., Jiang, Y., y Forghieri, F. (2014). The gn-model of fiber non-linear propagation and its applications. *Journal of Lightwave Technology*, *32*(4), 694–721.

Project, S. (2023). *Alembic documentation.* https://alembic.sqlalchemy.org/. (Accessed: 2024-11-21)

Rafique, D., y Velasco, L. (2018). Machine learning for network automation: overview, architecture, and applications [invited tutorial]. *IEEE/OSA Journal of Optical Communications and Networking*, *10*(10), D126–D143.

Ramaswami, R., Sivarajan, K., y Sasaki, G. (2009). *Optical networks: A practical perspective.* San Francisco, CA: Morgan Kaufmann.

Rosen, E., Viswanathan, A., y Callon, R. (2001, enero). *Multiprotocol Label Switching Architecture.* RFC 3031. (Available: https://www.rfc-editor.org/info/rfc3031)

Rottondi, C., Barletta, L., Giusti, A., y Musumeci, F. (2018). Machine-learning method for quality of transmission prediction of unestablished lightpaths. *IEEE/OSA Journal of Optical Communications and Networking*, *10*(2), A286–A297. doi: 10.1364/JOCN.10.00A286

Safari, P., Shariati, B., Bergk, G., y Fischer, J. K. (2021). Deep convolutional neural network for network-wide qot estimation. *IEEE Journal of Selected Topics in Quantum Electronics*, *27*(6), 1–10.

Shao, J., Liang, X., y Kumar, S. (2014, August). Comparison of split-step fourier schemes for simulating fiber optic communication systems. *IEEE Photonics Journal*, *6*(4), 1–15.

Shariati, B., Zervas, G., Nejabati, R., y Simeonidou, D. (2017). Impact of spatial and spectral granularity on the performance of sdm networks based on

spatial superchannel switching. *Journal of Lightwave Technology*, *35*(13), 2559–2568. doi: 10.1109/JLT.2017.2692301

Team, S. (2023). *Streamlit documentation.* https://docs.streamlit.io/. (Accessed: 2024-11-21)

Union, I. T. (s.f.). *Itu-t recommendations on optical transport networks.* https://www.itu.int/en/ITU-T/Pages/default.aspx. (Accessed: 2024-11-23)

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*. (Received 11 August 2023, accepted 30 August 2023, published 6 September 2023, current version 14 September 2023) doi: 10.1109/ACCESS.2023.3312387

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. doi: 10.1038/s41592-019-0686-2

Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. Descargado de https://doi.org/10.21105/joss.03021 doi: 10.21105/joss.03021

YData. (2023). *Ydata: Data science tools for data quality and synthetic data generation.* Descargado de https://github.com/ydataai/ydata-synthetic (GitHub repository, accessed: 2023-09-07)