UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN Y DISEÑO DIGITAL

CARRERA DE TELEMÁTICA

Trabajo de Integración Curricular previa la obtención del Grado Académico de Ingeniero en Telemática.

Título del proyecto de investigación:

TÉCNICAS DE APRENDIZAJE AUTOMÁTICO PARA REDUCIR LATENCIA EN REDES INALÁMBRICAS DE QUINTA GENERACIÓN

Autor

Jordy Steven Lucas Cantos

Director de proyecto:

Ing. Ángel Iván Torres Quijije, M.Sc.

Quevedo – Los Ríos – Ecuador

2025

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

Yo, **JORDY STEVEN LUCAS CANTOS,** declaro que la investigación aquí descrita es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Técnica Estatal de Quevedo, puede hacer uso de los derechos correspondientes a este documento, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

JORDY STEVEN LUCAS CANTOS

C.I: 1206591131

CERTIFICACIÓN DE CULMINACIÓN DEL PROYECTO DE INVESTIGACIÓN

El suscrito, **Ing. Ángel Iván Torres Quijije, M.Sc**. Docente de la Universidad Técnica Estatal de Quevedo, certifica que el estudiante **Jordy Steven Lucas Cantos**, realizó el Proyecto de Investigación de grado titulado **"Técnicas de Aprendizaje Automático para Reducir Latencia en Redes Inalámbricas de Quinta Generación"** previo a la obtención del título de **Ingeniero en Telemática**, bajo mi dirección, habiendo cumplido con las disposiciones reglamentarias establecidas para el efecto.

Ing. Ángel Iván Torres Quijije, M.Sc.
DIRECTOR DEL PROYECTO DE INVESTIGACIÓN

CERTIFICADO DEL REPORTE DE LA HERRAMIENTA DE PREVENCIÓN DE COINCIDENCIA Y/O PLAGIO ACADÉMICO

El suscrito, **Ing. Ángel Iván Torres Quijije, M.Sc**. Mediante el presente cumplo en presentar a usted, el informe de proyecto de Investigación titulado **"Técnicas de Aprendizaje Automático para Reducir Latencia en Redes Inalámbricas de Quinta Generación"** Presentado por el estudiante **Jordy Steven Lucas Cantos** egresado de la Carrera de telemática, que fue revisado bajo mi dirección según resolución del Consejo Directivo de la Facultad de Ciencias de la Computación y Diseño Digital, que se ha desarrollado de acuerdo al Reglamento de la Unidad de Integración Curricular de la Universidad Técnica Estatal de Quevedo y cumple con el requerimiento de análisis del sistema COMPILATIO el cual avala los niveles de originalidad en un 93% y similitud 7%, del trabajo investigativo. Valido este documento para que el estudiante siga con los trámites pertinentes, de acuerdo como lo establece el Reglamento.


CERTIFICADO DE ANÁLISIS
magister

Tesis JSLC-Compilatio

7%
Textos sospechosos

7% Similitudes

5% Idiomas no reconocidos (ignorado)

Nombre del documento: Tesis JSLC-Compilatio.pdf
ID del documento: 6bac27d584df709c739bbe51065190ff6643eea2
Tamaño del documento original: 1,2 MB

Depositante: ANGEL IVAN TORRES QUIJIJE
Fecha de depósito: 17/11/2025
Tipo de carga: interface
fecha de fin de análisis: 17/11/2025

Número de palabras: 12.038
Número de caracteres: 85.996

Ing. Ángel Iván Torres Quijije, M.Sc.

DIRECTOR DEL PROYECTO DE INVESTIGACIÓN

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN Y DISEÑO DIGITAL

CARRERA DE TELEMÁTICA

PROYECTO DE INVESTIGACION

TÍTULO:

Técnicas de Aprendizaje Automático para Reducir Latencia en Redes Inalámbricas de Quinta Generación

Presentado al Consejo Directivo de la Facultad de Ciencias de la Computación y Diseño Digital, como requisito previo a la obtención del título de Ingeniero en Telemática.

Aprobado por:

PRESIDENTE DEL TRIBUNAL

Ing. Diego Fernando Intriago Rodríguez, M.Sc.

MIEMBRO DEL TRIBUNAL

Ing. Santiago Meneses Narváez, M.Sc.

MIEMBRO DEL TRIBUNAL

Ing. Paola Benítez Navarrete, M.Sc.

QUEVEDO – LOS RÍOS – ECUADOR

2025

AGRADECIMIENTO

En primer lugar, deseo expresar mi más profundo agradecimiento a dios por haberme guiado con sabiduría, fortaleza y fe inquebrantable a lo largo de este camino, permitiéndome alcanzar esta meta académica.

A mis padres Bolívar Lucas y Ana Cantos quienes han sido el pilar fundamental de mi vida, les agradezco su apoyo incondicional, sus sacrificios y confianza constante. Sin su esfuerzo y amor la culminación de este proyecto no habría sido posible.

A mis hermanos Marlene, Viviana y Jeffrey Lucas por ser una fuente inagotable de inspiración, ánimo y comprensión en momentos complicados de mi vida.

A mi amada novia Julexy Mosquera por su paciencia infinita, motivación diaria y apoyo emocional. Tu presencia y compañía han sido mi refugio en momentos difíciles que llegaban a cuestionar mis capacidades, eres la mejor recompensa durante todo el proceso.

A mis cuñados Beto Palma y Luiggi Moreira por darme el ánimo y apoyo necesario para no perder el enfoque y cumplir con este logro profesional como ingeniero.

A la madre de mi novia por su inestimable, su hospitalidad al brindarme un lugar seguro en los días de largas jornadas y su aporte generoso con cualquier detalle que requerí. Su calidez ha sido esencial.

A mis abuelos, tíos, primos y toda la familia por su apoyo condicional y alentarme a cumplir con este proceso tan importante.

Un agradecimiento especial a mi grupo de trabajo el Equipo dinamita conformado por Julexy Mosquera, Karelys Pacheco, Dannar Delgado y Dayanna Castro, mujeres con grandes capacidades cuyo compromiso y talento fueron esenciales para completar con los proyectos académicos. Incluyo en mi gratitud a mis amigos Kenneth Rodríguez, Alexander Maigua, Christian Guerrero, Nilo Medina, Kimberly Mendoza, José Fabre, Kevin Zambrano, Francisco Vera, Wilmer Zambrano por su valioso apoyo y por compartir los momentos de estudio y esparcimiento que hicieron más llevadera esta etapa.

A mi tutor de proyecto el Ing. Ángel Torres Quijije, M. Sc por su dirección experta, tiempo y valiosos conocimientos. A los docentes de la carrera ingeniería en telemática por transmitir sus conocimientos y formar profesionales íntegros. Finalmente agradezco a la universidad técnica estatal de Quevedo por brindarme la oportunidad, lugar, plataforma y recursos necesarios para mi desarrollo y la ejecución de esta investigación.

Jordy Steven Lucas Cantos

DEDICATORIA

Dedico este proyecto que marca la culminación de mis estudios y el inicio de mi carrera profesional a las personas que lo hicieron posible con su amor, sacrificio y fe inquebrantable.

A mis padres Bolívar Lucas y Ana Cantos, arquitectos de mi vida y carácter. A ellos les debo firmeza de mis valores, visión de mi porvenir y convicción de que el esfuerzo siempre rinde sus frutos. Su amor y presencia constante fueron el faro que ilumino cada jornada de este exigente recorrido.

A mis hermanos, Marlene, Viviana y Jeffrey Lucas, compañeros de vida. A ellos les confío el testimonio de este logro; por haber compartido las alegrías y los silencios, y por ser una fuente de inspiración mutua que me motivó a avanzar incluso en los momentos más complejos.

A mi querida novia, Julexy Mosquera. A ti, compañera incondicional, dedico la energía final de este trayecto. Tu serenidad ante mis jornadas de desvelo, tu fe inquebrantable en mis capacidades y tu apoyo fueron el ancla emocional que me mantuvo firme. Cada triunfo plasmado en estas páginas está entrelazado con tu invaluable sacrificio y tu presencia constante, que fue mi remanso de paz. Este éxito lleva intrínsecamente tu huella, pues eres mi mayor motor y mi recompensa más hermosa.

Jordy Steven Lucas Cantos

RESUMEN

La reducción de latencia en redes inalámbricas de quinta generación representa un obstáculo crítico en el desempeño óptimo de aplicaciones sensibles al tiempo de respuesta como telemedicina, Iot y realidad aumentada que requieren de un sistema estable con el mejor rendimiento que evite perdidas en la información. El uso de algoritmos avanzados como el aprendizaje automático representa una innovación en la mejora de las redes actuales, estos algoritmos permiten optimizar la gestión de recursos y reducir significativamente los tiempos de respuesta que existe entre comunicaciones inalámbricas en entornos de quinta generación. Identificar modelos supervisados y no supervisados permiten una mejor adaptación de funcionamiento en el enlace de equipos que son evaluados en su rendimiento bajo distintas condiciones de tráfico, los resultados son una clave para la mejora del algoritmo que permita una adaptación automática en solución a mejores respuestas porque evidencian la viabilidad en el uso de algoritmos avanzados en la red que permitan una mejoría constante en el tráfico de redes al reducir el tiempo de retraso que se genera en el uso de equipos inalámbricos.

Palabras clave: Latencia, algoritmos avanzados, aprendizaje automático.

ABSTRACT

Reducing latency in fifth-generation wireless networks represents a critical bottleneck in the optimal performance of time-sensitive applications such as telemedicine, IoT and augmented reality that require a stable system with the best performance to avoid data loss. The use of advanced algorithms such as machine learning represents an innovation in the improvement of current networks, these algorithms allow optimizing Resource management and significantly reducing the response times that exist between Wireless communications in fifth generation environments. The identification of supervised and unsupervised models allows a better adaptation of the link performance of the equipment that are evaluated in their performance under different traffic conditions, these results are key to the improvement of the algorithm that allows an automatic adaptation in solution to better responses. These results show the feasibility in the use of advanced algorithms in the network that allow a constant improvement in the network traffic reducing the delay time generated in the use of wireless equipment.

Keywords: Latency, advanced algorithms, machine learning.

CÓDIGO DUBLÍN

Título:	Técnicas de Aprendizaje Automático para Reducir Latencia en Redes Inalámbricas de Quinta Generación		
Autor:	Jordy Steven Lucas Cantos		
Palabras claves:	Latencia	Algoritmos avanzados	Aprendizaje automático
Fecha de publicación:	25 de septiembre del 2025		
Editorial:	Quevedo- UTEQ "Campus Central", 2025		
Resumen: (hasta 300 palabras)	Latencia crítica en las redes móviles 5G es el principal obstáculo para el rendimiento de las aplicaciones sensibles al tiempo (telemedicina, IoT), lo que requiere una optimización constante de los recursos de la red. Este estudio propone el uso de algoritmos de aprendizaje automático (ML) como innovación para el enrutamiento dinámico, lo que reduce significativamente los tiempos de respuesta. La investigación evaluó la viabilidad y la eficacia de los modelos supervisados y no supervisados para el ajuste automático del tráfico, lo que garantiza una mejora constante en el flujo de datos y minimiza los retrasos asociados al uso de dispositivos en entornos 5G.		
Abstract: (hasta 300 palabras)	Critical latency in 5G mobile networks is the main obstacle to the performance of time-sensitive applications (telemedicine, IoT), requiring constant optimization of network resources. This study proposes the use of machine learning (ML) algorithms as an innovation for dynamic routing, which significantly reduces response times. The research evaluated the feasibility and effectiveness of supervised and unsupervised models for automatic traffic adjustment, ensuring constant improvement in data flow and minimizing delays associated with device use in 5G environments.		
Descripción:	121 hojas: dimensiones, 29 x 21 cm + CD-ROM 6162		
URI:			

Tabla de contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE FORMULAS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

La tecnología de Quinta generación(5G) representa un cambio en el ámbito de telecomunicaciones por su capacidad para incrementar grandes velocidades de transferencia de datos y reducir significativamente latencia, aumentando la fiabilidad en aplicaciones críticas en tiempo real. Con latencias teóricas inferiores a un milisegundo y velocidades de hasta 20 Gbps, están diseñadas para soportar una variedad de servicios como realidad virtual y aumentada (AR/VR), Internet de las Cosas (IoT), telemedicina y vehículos autónomos inteligentes [1].

Sin embargo, la implementación de 5G enfrenta retos técnicos significativos. Entre ellos, la gestión dinámica de recursos de red para adaptarse a condiciones cambiantes del entorno y demandas de usuarios es uno de los desafíos más apremiantes. Densificación de red, utilización de espectros de frecuencias como mmWave, integración de tecnologías avanzadas como MIMO masivo y beamforming añaden complejidad al panorama [2]. En este contexto, garantizar tiempos de respuesta ultra bajos con un rendimiento constante requiere soluciones innovadoras y altamente adaptativas.

El aprendizaje automático o machine learning (ML), una rama de la inteligencia artificial surge como una herramienta clave para abordar estos desafíos. Mediante el análisis de grandes volúmenes de datos y capacidad de identificar patrones complejos, el ML ofrece enfoques novedosos para optimizar el rendimiento de redes al predecir condiciones de congestión, asignar dinámicamente recursos de la red y minimizar la latencia en tiempo real [3], [4]. Estas capacidades son esenciales para cumplir con los requisitos de aplicaciones críticas y servicios ultra confiables

Para la implementación y presentación de los modelos de ML se utilizará Google Colab. Esta plataforma permitirá ejecutar código en la nube, facilitando la demostración y visualización de las funciones desarrolladas para entrenar los modelos. Se espera contribuir al cuerpo de conocimiento existente sobre la aplicación de Aprendizaje automático en telecomunicaciones, abordando específicamente el problema crítico de la latencia en redes 5G y sentando las bases para futuros desarrollos en esta área.

CAPÍTULO I

CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN

1.1. Problema de investigación

1.1.1. Planteamiento del problema

En la actualidad, las redes de quinta generación (5g) representan un avance significativo en términos de conectividad, velocidad y capacidad para soportar aplicaciones críticas, como la realidad aumentada, vehículos autónomos y el Internet de las Cosas (IoT). Sin embargo, uno de los principales desafíos que enfrentan 5G es latencia, cuyo impacto puede comprometer la calidad del servicio y limitar el aprovechamiento pleno de las capacidades de la tecnología 5G.

La latencia se genera debido a factores como gestión ineficiente de recursos, el creciente volumen de tráfico y necesidad de procesar grandes cantidades de datos en tiempo real. Los enfoques tradicionales para gestión de recursos han demostrado ser insuficientes para abordar demandas dinámicas y complejas que requieren las redes 5G. Esto ha llevado a explorar técnicas de ML como una alternativa innovadora y prometedora.

Las técnicas de ML permiten analizar grandes volúmenes de datos y tomar decisiones optimizadas que mejoran la asignación de recursos y, en consecuencia, disminuyen latencia. Diversos estudios han demostrado que el uso de estas técnicas puede reducir significativamente los tiempos de respuesta en la red, lo que se traduce en una mejora del rendimiento global del sistema. Sin embargo, su efectividad depende de validaciones rigurosas que permitan simular y medir su funcionamiento en escenarios controlados antes de implementarlas en entornos reales.

1.1.2. Formulación del problema

¿Cómo la implementación de técnicas de aprendizaje automático puede contribuir a la reducción de latencia en redes inalámbricas de quinta generación mediante el uso de algoritmos avanzados?

1.1.3. Sistematización del problema

- ¿De qué manera la implementación de técnicas de aprendizaje automático puede contribuir a la reducción de latencia en redes inalámbricas de quinta generación a nivel global?

- ¿Cuáles son las técnicas de aprendizaje automático más efectivas para la reducción de latencia en redes de quinta generación y cómo pueden ser aplicadas en entornos simulados representativos?
- ¿Cómo las técnicas de aprendizaje automático pueden ser evaluadas en escenarios específicos de simulación?

1.2. Objetivos

1.2.1. Objetivo general

Aplicar técnicas de enrutamiento dinámico para la reducción de latencia en redes inalámbricas de quinta generación empleando algoritmos avanzados.

1.2.2. Objetivos específicos

- Seleccionar técnicas de aprendizaje automático adecuadas para la reducción de latencia en redes de quinta generación, con base en una revisión del estado del arte.
- Diseñar escenarios que simulen el desempeño de técnicas de aprendizaje automático representando entornos de quinta generación.
- Analizar la reducción de latencia del enrutamiento dinámico en redes de quinta generación.

1.3. Justificación

Uno de los problemas más críticos en las redes 5G es la latencia, que se refiere al tiempo que tarda un paquete de datos en viajar desde su origen hasta su destino. Aunque la tecnología 5G está diseñada para ofrecer latencias mucho menores que sus predecesoras (de aproximadamente 200 ms en 4G a menos de 1 ms en 5G) [6], existen factores como la congestión de red, interferencia y asignación ineficiente de recursos pueden provocar aumentos inesperados en latencia. Esto afecta a la calidad del servicio (QoS) y también la experiencia del usuario final [7].

La latencia en redes 5G es uno de los principales retos para lograr comunicaciones eficientes y confiables, especialmente en aplicaciones como telemedicina, realidad aumentada y vehículos autónomos. Aunque la tecnología 5G establece objetivos teóricos de latencia cercanos a 1 ms, en la práctica se producen pérdidas significativas debido a la congestión, interferencias y dificultades en la asignación dinámica de recursos, principalmente en conexiones inalámbricas. Esta situación afecta directamente la calidad del servicio y la experiencia del usuario.

Como estudiante de ingeniería telemática, me parece importante encontrar soluciones innovadoras a los problemas de latencia que afectan a las redes inalámbricas. La creciente demanda de servicios de conectividad y el número cada vez mayor de dispositivos requieren mecanismos inteligentes para optimizar el uso del espectro y mejorar la gestión de los recursos. El uso del ML representa una oportunidad de aplicar técnicas avanzadas para minimizar la pérdida de rendimiento en situaciones reales.

Este proyecto busca abordó la reducción de latencia mediante técnicas de aprendizaje automático utilizando algoritmos supervisados y no supervisados para reducir congestiones y gestionar el tráfico adaptativamente. Las pruebas se realizaron en un entorno simulado como Google Colab para implementar y mostrar funciones, análisis de datos y resultados de simulación. La finalidad es aportar soluciones prácticas para redes operativas, mejorando rendimiento y confiabilidad que ofrezcan contribuir al progreso académico impulsando una infraestructura eficiente para futuras tecnologías 5G.

CAPÍTULO II

FUNDAMENTACIÓN TEÓRICA DE LA INVESTIGACIÓN

2.1. Marco conceptual

2.1.1. Redes de quinta generación

Las redes de quinta generación (5G) representan un avance significativo en la tecnología de telecomunicaciones al ofrecer las velocidades más altas en transmisión de datos, menor latencia y mayor capacidad de conexión para varios dispositivos. Este sistema tiene muchos beneficios que pueden crear desafíos a la red en general, la alta demanda de las personas en el uso del internet presenta muchos inconvenientes como retraso, consumo energético, sostenibilidad ambiental e infraestructura[5].

2.1.2. Redes inalámbricas

Es una tecnología utilizada para la creación de redes de área local inalámbricas (WLAN), se rige bajo los estándares operativos por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE). La red inalámbrica permite una comunicación sin cableado mediante un Punto de Acceso (AP) que envía señales en forma de onda de radio a través de frecuencias específicas como 2.4 GHz y 5GHz que transportan estos de un dispositivo como teléfonos inteligentes, computadoras portátiles impresoras y cámaras[6].

2.1.2.1. Tipos de WIFI

- WI-FI-802.11ac es el último y mejor protocolo de wifi 5 lanzado en 2013 con el objetivo de favorecer aplicaciones de alto rendimiento, funciona en la frecuencia de 5 GHz con una tecnología MIMO (Múltiple entrada, Múltiple salida) y diversidad espacial para atender hasta 4 clientes simultáneamente con un rendimiento de datos hasta 1 Gigabit/s.
- WI-FI-802.11n lanzado en 2009 como wifi 4con una característica distintiva en el uso de múltiples antenas que aumentan las velocidades de datos, esta estrategia de diversidad de antenas o MIMO mejoras las velocidades de datos hasta 74 Mbit/s con datos transferidos mediante OFDM (Multiplexación por División de Frecuencia Ortogonal), permitiendo la transmisión simultanea de datos con mayor eficacia.

- WI-FI-802.11g o también conocido como wifi 3 funciona en la frecuencia de 4.2 GHz con velocidades de datos que oscilan entre 3 y 54 Mbit/s, utiliza tecnología como CSMA/CA (Acceso múltiple por detección de portadora y prevención de colisiones) y OFDM para codificar/distribuir los datos transmitidos a través de 50 frecuencias subportadoras.
- WI-FI-802.11b también conocido como wifi 1 lanzado en 1999 para redes inalámbricas con velocidades hasta 11 Mbits/s en la banda de 2.4 GHz. Utiliza CSMA/CA como protocolo para acceso a medios que permite a varios dispositivos transmitir paquetes de datos a través de la red sin colisiones.
- WI-FI-802.11a lanzado en 1999 con el nombre de wifi 2 era caracterizado por usar por primera vez la tecnología OFDM a diferencia de sus antiguas versiones que usaban el espectro ensanchado de secuencia directa (DSSS), Esta versión de wifi utiliza la banda de 5GHz con una ligera cobertura ligeramente menor a sus homólogos de 2.4 GHz debido a su reducida penetración en aquella frecuencia[6].

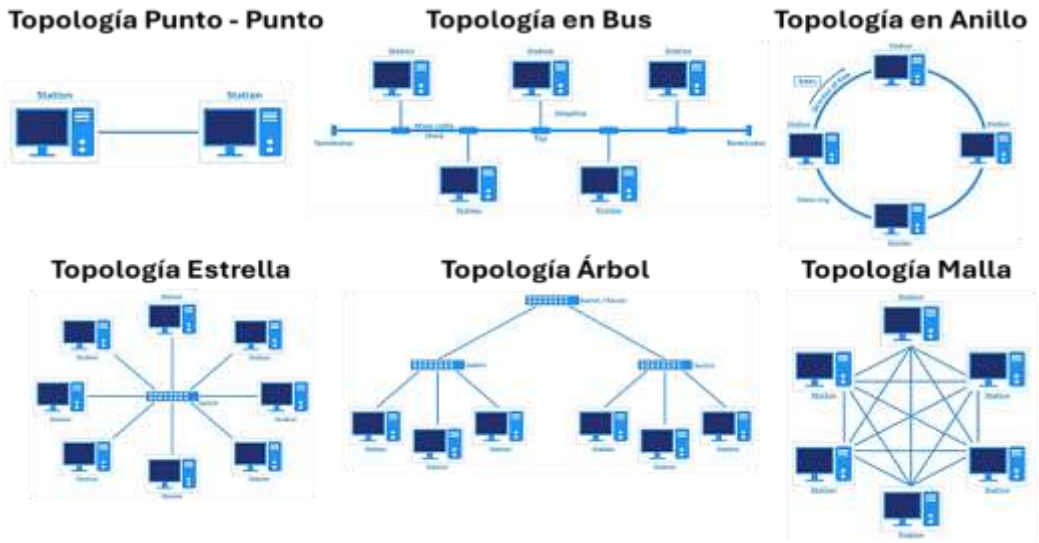
2.1.3. Topologías de red

Las topologías describen la conexión esquemática y rendimiento que alcanza una red entre sus dispositivos, cuando se construye una red de ordenadores es necesario definir qué topología de red se desea utilizar según Nakivo[7] .Las redes más tradicionales incluyen las topologías:

- **Topología punto a punto:** Es la más sencilla y se utiliza cuando sólo hay dos ordenadores u otros dispositivos de red conectados entre sí.
- **Topología en bus:** estándar Token Bus que se utiliza para crear un anillo token lógico en redes construidas utilizando la topología de bus. Se pasa una ficha de una estación a otra en una secuencia definida que representa el anillo lógico en el sentido de las agujas del reloj o en sentido contrario.
- **Topología red en anillo:** cada estación está conectada a otras dos estaciones a cada lado. Las otras dos estaciones son vecinas de esta estación. Los datos viajan secuencialmente en una dirección, por lo que la red funciona en modo semidúplex.

- **Topología en estrella:** es la topología de red más utilizada hoy en día por las numerosas ventajas que ofrece. Esta topología requiere una unidad centralizada, que se denomina conmutador, y todos los demás dispositivos de red se conectan a este conmutador con cable de red propio.
- **Topología en red árbol:** es una extensión de la topología en estrella, se puede conectar varias estrellas como ramas en una red compleja mediante conexiones entre conmutadores. Las estaciones se conectan a los puertos de estos conmutadores.
- **Topología en malla:** configuración en la que cada estación de la red está conectada a las demás estaciones. Todos los dispositivos están interconectados entre sí. Hay dos tipos de malla: la malla completa y la malla parcial. En una malla parcialmente conectada, al menos dos estaciones de la red están conectadas a otras múltiples estaciones de la red. En una malla completa, cada estación está conectada a todas las demás.

Ilustración 1. Topologías de red



Autor: Nakivo[7]

Fuente: https://www.nakivo.com/es/blog/types-of-network-topology-explained/

2.1.4. Aprendizaje automático

El aprendizaje automático o machine learning (ML) es una disciplina dentro de la inteligencia artificial que capacita a las máquinas para que aprender de los datos y mejorar su desempeño de manera autónoma, sin necesidad de programación explícita. aprenden de forma autónoma a realizar una tarea o hacer predicciones a partir de datos y mejorar su rendimiento con el tiempo. Una vez entrenado, el algoritmo podrá encontrar los patrones en nuevos datos[8].

Ilustración 2. Proceso de construcción del aprendizaje automático

Autor: Martín[9]

Fuente: https://www.ideati.net/blog/ia-predictivo-construccion/

- **Obtener datos:** El proceso de aprendizaje depende de calidad y cantidad disponibles que puede ser proporcionado por el usuario que entrene el modelo.
- **Limpiar y preparar datos:** Paso fundamental que ocupa un gran trabajo debido a la calidad de datos que debe ser proporcionado al modelo, en este paso se realizan tareas como remoción de duplicados e información incorrecta, incompleta o corrupta que alteren el resultado final.
- **Entrenar modelo:** Aquí se elige el modelo adecuado para nuestro problema a resolver y entrenar con los datos que han sido procesados anteriormente para que aprenda de ellos.
- **Evaluar resultados:** se trata de medir los resultados obtenidos para determinar su calidad, esta evaluación nos determina si debemos volver atrás en alguna etapa o podemos continuar.
- **Publicar para uso:** La solución está lista para trabajar con nuevos datos y casos reales por lo que debe estar listo disponible en un ambiente de producción para ser consultada por los usuarios y las herramientas informáticas que lo requieran.

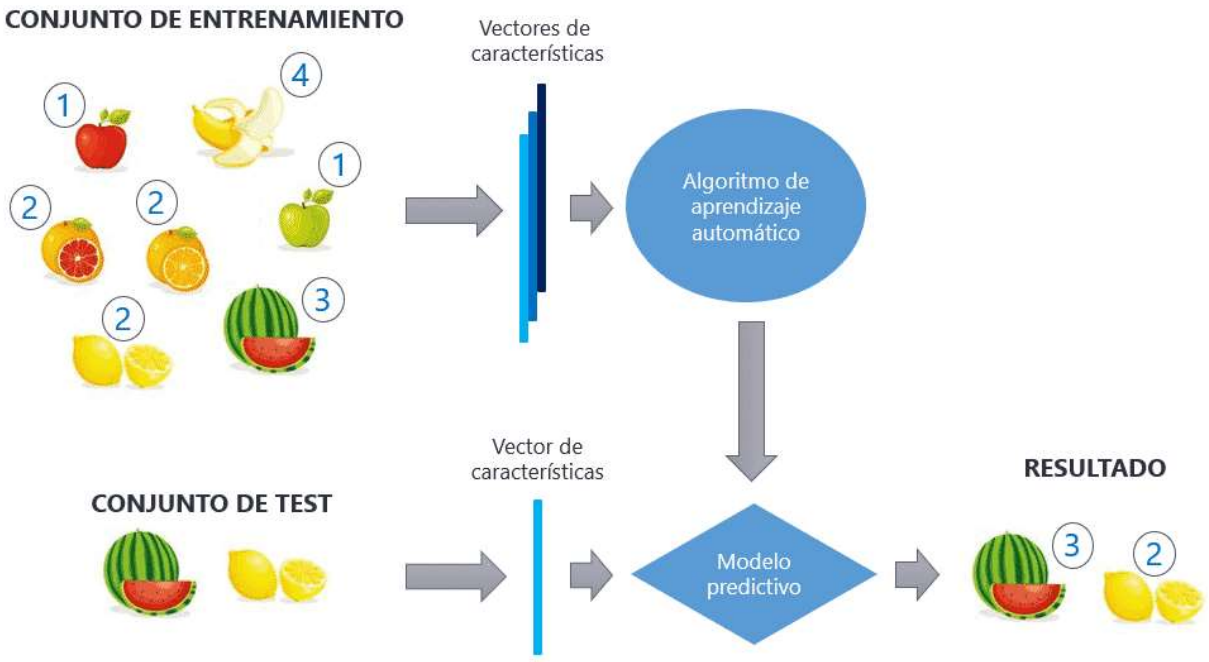
- **Seguimiento y mejora:** No debemos desatender las herramientas una vez iniciado su uso, sino que es importante darle seguimiento para evaluar resultados, en caso de alguna falla se debe realizar ajustes al modelo predictivo elegido con el objetivo de tener siempre la mayor calidad posible [9].

2.1.4.1. Tipos de aprendizaje automático

2.1.4.1.1. Aprendizaje supervisado

Utiliza un conjunto de datos etiquetados para entrenar el modelo con el objetivo de predecir las salidas correctas a partir de las entradas proporcionadas, busca aprender de la relación entre las variables proporcionadas. Los datos para el entrenamiento incluyen la solución deseada, llamada "etiquetas" (labels). Un claro ejemplo es al clasificar correo entrante entre Spam o no. Entre las diversas características que queremos entrenar deberemos incluir si es correo basura o no con un 1 o un 0 [10]

Ilustración 3. Funcionamiento de modelo supervisado



Autor: Diego Calvo[11]

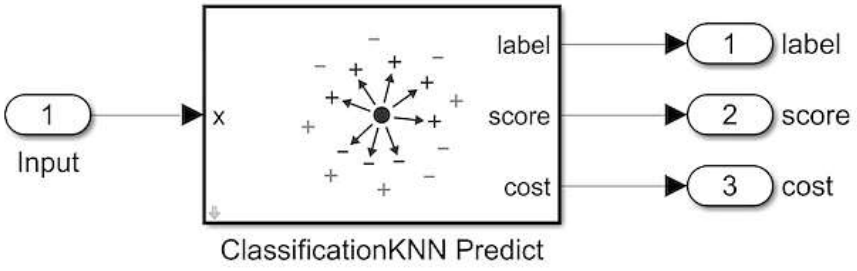
Fuente: https://www.diegocalvo.es/aprendizaje-supervisado/

2.1.4.1.2. Tipos de aprendizaje supervisado

2.1.4.1.2.1 Vecinos más cercanos (KNN)

Método de clasificación extremadamente sencillo, pero increíblemente eficaz, su atractiva radica en sus superficies de decisión no lineales con un único parámetro entero ajustable con validación cruzada que ofrece una calidad esperada en predicciones automáticas. El principio detrás de los métodos del vecino más cercano es encontrar un número predefinido de muestras de entrenamiento más cercanas en distancia al nuevo punto, y predice la etiqueta a partir de estos. El número de muestras puede ser definido por el usuario constante (aprendizaje de k vecinos más cercanos) o basado en variaciones sobre la densidad local de puntos (aprendizaje de vecinos basado en radio) [12].

Ilustración 4. modelo de clasificación KNN



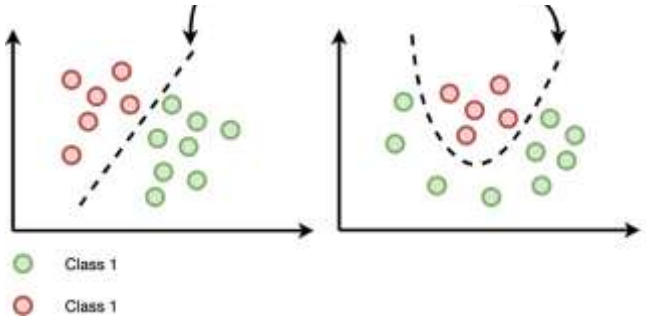
Autor: MathWorks[13]

Fuente: https://la.mathworks.com/help/stats/predict-class-labels-using-classification-knn-predict-block.html

2.1.4.1.2.2 Regresión logística (LOGR)

Modelo estadístico fundamental para la clasificación siendo su función principal modelar la probabilidad de un resultado para clasificar a una clase determinada. Utiliza funciones logísticas (o sigmoide) para obtener predecir resultados binarios basado en observaciones previas de un conjunto de datos analizados en la relación entre una o más variables independientes en la función lógica[14].

Ilustración 5. Clasificación de LOGR



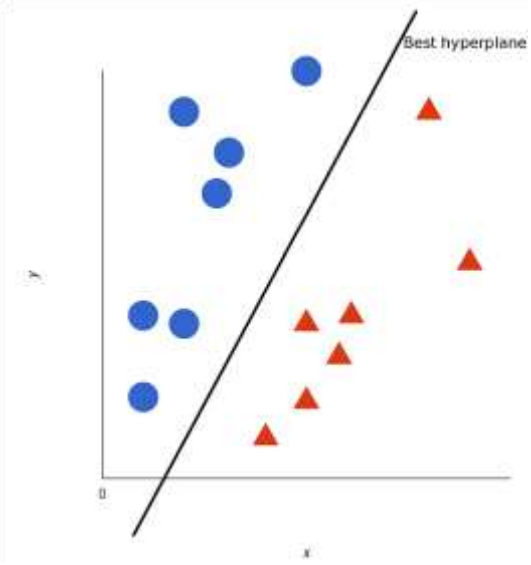
Autor: Luis Torres[15]

Fuente: https://www.themachinelearners.com/regresion-logistica-regularizacion/

2.1.4.1.2.3 Máquina vectorial de soporte (SVM)

Algoritmo de aprendizaje automático utilizado para problemas de clasificación o regresión que clasifica los datos mediante la búsqueda de una línea o hiperplano optimo que maximiza la distancia entre cada clase en un espacio. Toma estos puntos de datos y genera el hiperplano (que en dos dimensiones es simplemente una línea) que separa mejor las etiquetas. Esta línea es el límite de decisión: todo lo que caiga a un lado lo clasificaremos como azul y todo lo que caiga al otro lado como rojo.

Ilustración 6. Clasificación de SVM



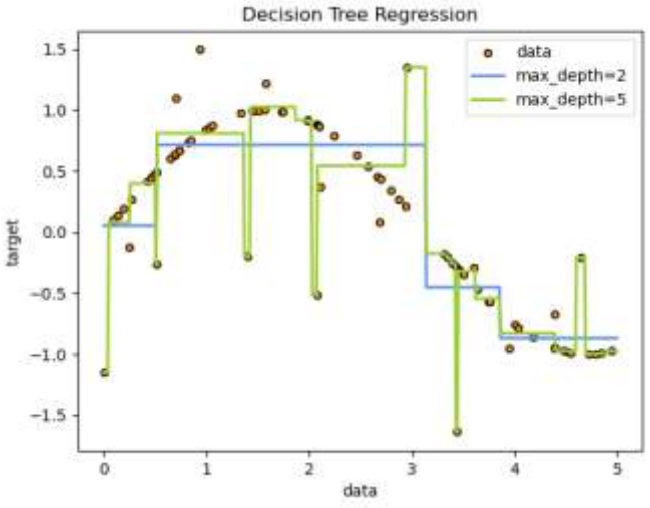
Autor: César Rodríguez[16]

Fuente: https://medium.com/@csarchiquerodriguez/maquina-de-soporte-vectorial-svm-92e9f1b1b1ac

2.1.4.1.2.4 Árbol de decisión/Random Forest (RF)

Un diagrama de árbol de decisiones permite evaluar mediante el aprendizaje de reglas de decisión simples inferidas a partir de los datos característicos, un árbol puede verse como una aproximación constante por partes que aprenden de los datos para aproximar una curva sinusoidal con un conjunto de reglas de decisión if-then-else (sí, entonces, Sino) para determinar las mejores decisiones[17].

Ilustración 7. Ejemplo de DTs



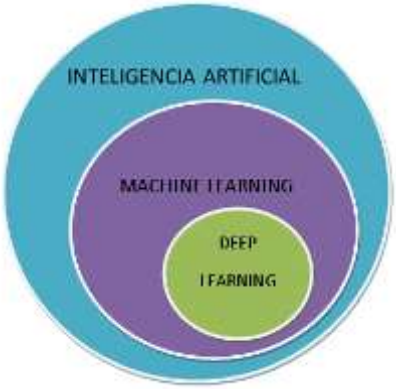
Autor: Scikit Learn[17]

Fuente: https://scikit-learn.org/stable/modules/tree.html

2.1.4.2. Aprendizaje Profundo

El aprendizaje profundo o conocido globalmente como Deep Learning es un subconjunto de varios modelos de aprendizaje automático que utiliza redes neuronales artificiales (ANN) como sistema de multicapas apiladas de forma jerárquica al seleccionar características más importante capa por capas. La primera capa aprende de patrones muy simples, la siguiente capa combina esos patrones en formas más complejas, las capas más profundas continúan combinando la información hasta comprender los conceptos que mejoren la función basada en redes neuronales multicapa[18].

Ilustración 8. Estructura del Aprendizaje Profundo



Autor: Raúl López[19].

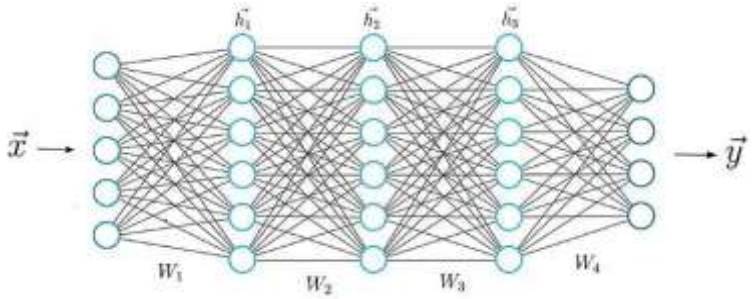
Fuente: https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

2.1.4.2.1. Tipos de aprendizaje Profundo

2.1.4.2.1.1 Redes Neuronales (ANN)

Una red neuronal es un reflejo del comportamiento del cerebro humano que permite a los programas informáticos reconocer patrones y resolver problemas en los campos del aprendizaje automático, estos sistemas también se conocen como redes neuronales artificiales (ANN) o redes neuronales simuladas (SNN). Su estructura está diseñada para parecerse al cerebro humano, lo que hace que las neuronas biológicas se envíen señales entre sí. Las ANN contienen capas de nodos que comprenden una entrada, una o más capas ocultas y una capa de salida[20].

Ilustración 9. Ejemplo de red neuronal



Autor: Scikit Learn[17]

Fuente: https://scikit-learn.org/stable/modules/tree.html

Cada neurona artificial está conectada a otra y tiene un umbral y un peso asociados. Cuando la salida de cualquier nodo está por encima del umbral, ese nodo se activará y enviará datos a la siguiente capa. Si no se supera el umbral, no se pasan datos al siguiente nodo. Las redes neuronales dependen de los datos de entrenamiento para aprender y mejorar su precisión con el tiempo. Una vez que estos algoritmos de aprendizaje se ajustan a la precisión, se convierten en herramientas poderosas en IA. Nos permiten clasificar y agrupar datos a alta velocidad. Las tareas de reconocimiento de imágenes tardan sólo unos minutos en procesarse en comparación con la identificación manual[20].

2.1.4.2.1.2 Memoria de corto y largo plazo (LSTM)

Es un modelo computacional basado en redes neuronales biológicas compuesta por capas de nodos (también conocido como neuronas), la información fluye desde una entrada (recibe todas las características) a través de una o más capas ocultas (entrenamiento y clasificación) hasta una capa de salida encargada de dar la predicción. La función de cada neurona en una capa oculta que calcula una suma moderada de sus entradas aplicando una función de activación para introducir no linealidad y pasar su resultado a la siguiente capa, el aprendizaje ocurre durante el entrenamiento mediante un proceso llamado retro propagación que ajusta los "pesos" de las conexiones para minimizar los errores de predicción. Las redes LSTM pueden utilizar sus conexiones de retroalimentación para almacenar representaciones de eventos de entrada en forma de activaciones (corto o largo plazo) para mejorar su resultado, este modelo es potencialmente significativo para muchas aplicaciones que incluyen el procesamiento de información y control[21].

Ilustración 10. Funcionamiento de LSTM

Autor: CMAX[22].

Fuente: https://ciberseguridadmax.com/lstm/

2.1.5. Métricas de Evaluación

2.1.5.1. Throughput (Rendimiento de la Red)

El throughput, o rendimiento, es la cantidad real de datos transmitidos con éxito a través de una conexión de red en un período de tiempo determinado. Se ve afectado por varios factores, incluida la latencia, el ancho de banda disponible y la congestión de la red. A diferencia del ancho de banda, que es una medida teórica de capacidad máxima, el throughput proporciona una visión más precisa de la eficiencia real de la red en condiciones de uso normales. el throughput puede ser menor que el ancho de banda teórico debido a la presencia de latencia y otros factores limitantes[23]. Se mide Kilobits por segundo (kbps) o Gigabits por segundo (Gbps).

2.1.5.2. Pérdida de Paquetes (Packet Loss)

La pérdida de paquetes ocurre cuando uno o más paquetes de datos que viajan a través de una red informática no logran llegar a su destino. Es un problema común tanto en entornos cableados como inalámbricos y puede afectar gravemente el rendimiento de la red, provocando retrasos, mala calidad de voz, vídeo y tiempos de respuesta lentos de las aplicaciones[24].

2.1.5.3. Jitter (Variabilidad de la Latencia)

El jitter es la variabilidad en el tiempo de llegada de los paquetes de datos en una red. Esta fluctuación puede causar interrupciones y retrasos en la transmisión de información, afectando negativamente la calidad de la conexión. Una conexión con alto jitter puede resultar en una experiencia de usuario frustrante. Las videollamadas pueden sufrir cortes, los juegos en línea pueden tener lag y los videos pueden pausarse inesperadamente[25].

2.1.5.4. Eficiencia Energética (Energy Efficiency)

La eficiencia energética se refiere a la capacidad de un recurso informático para convertir energía eléctrica en trabajo útil con un desperdicio o pérdida mínima. se mide en Joules por bit (J/bit) y es cada vez más importante para hacer frente a centros de datos con energía limitada y lograr una computación sostenible[26].

2.1.5.5. Ancho de banda (Bandwidth)

El ancho de banda se refiere a la cantidad máxima de datos que pueden transmitirse a través de una conexión de red en un período de tiempo dado. Se mide típicamente en bits por segundo (bps) o en sus múltiplos, como kilobits por segundo (kbps) o megabits por segundo (Mbps). Un alto ancho de banda significa que se pueden transferir grandes volúmenes de datos en poco tiempo, lo que permite una rápida transmisión de archivos y una navegación web fluida, especialmente en redes congestionadas. Sin embargo, es importante destacar que un alto ancho de banda no garantiza una baja latencia, ya que la latencia está influenciada por otros factores además del ancho de banda disponible[23].

2.1.5.6. Latencia

Latencia en las redes es el tiempo que tardan los datos en recorrer de un punto a otro en una red. Una red con latencia elevada tendrá tiempos de respuesta más lentos, mientras que una red con baja latencia tendrá tiempos de respuesta más rápidos, los paquetes de datos se mueven por la red a una velocidad ligeramente inferior debido a los retrasos causados por la distancia, la infraestructura de Internet y otras variables. La suma de estos retardos constituye la latencia de una red y se mide en Milisegundos (ms) [27].

2.2. Marco referencial

El articulo “Aprendizaje automático, diseño y optimización para comunicaciones inalámbricas 5g” usó técnicas de ML en el diseño y optimización de las comunicaciones inalámbricas, específicamente en el contexto de las redes 5G explora cómo el aprendizaje automático puede mejorar la eficiencia de los sistemas de comunicación mediante la predicción, el ajuste dinámico de parámetros y la automatización en la gestión de recursos. Además, se discuten las aplicaciones de estas tecnologías en la mejora de la calidad de servicio y el rendimiento en redes móviles avanzadas[28]

En “Algoritmos de aprendizaje automático para optimizar las redes 5G: Desarrollo y evaluación del rendimiento” los algoritmos de ML pueden optimizar el rendimiento de las redes 5G. Se repasan técnicas como las redes neuronales y el aprendizaje aumentado para la gestión de recursos. También se destacan las mejoras en la eficiencia del espectro, la

cobertura y la reducción del retardo. Los autores presentan pruebas experimentales del impacto positivo del aprendizaje aumentado en entornos 5G. Por último, concluyen que el uso del ML es fundamental para la evolución hacia redes más inteligentes y autónomas[29].

Explorando el uso de estrategias de asignación de recursos optimizadas mediante ML para redes 5G se desarrolló “Optimización de estrategias de asignación de recursos para redes 5G mediante Aprendizaje automático, diseño y optimización para comunicaciones inalámbricas 5g”. Se enfoca en cómo las técnicas de ML pueden mejorar la gestión de recursos en entornos de redes complejas, garantizando un rendimiento eficiente y la calidad del servicio. La investigación aborda desafíos técnicos y propone soluciones innovadoras para maximizar la eficiencia en la distribución de recursos, lo que es crucial para el éxito de las redes 5G[30].

El proyecto “Uso de Machine-Learning en el control de congestión sobre redes 5G” demuestra un aumento de usuarios en redes móviles y sus demandas impulsan la necesidad de soluciones como las redes 3G y futuras generaciones con frecuencias superiores a 24 GHz. Estas enfrentan desafíos como pérdidas de transmisión y variabilidad del canal, donde técnicas de ML ofrecen ventajas para predecir y gestionar la congestión. Este estudio utiliza simulaciones en ns-3 para entornos de ondas milimétricas, generando datos que permiten proponer soluciones predictivas eficientes.[31]

La investigación “Análisis del Rendimiento de Redes 5G utilizando Aprendizaje automático”, las redes 5G ofrecen mayor velocidad y eficiencia, pero su complejidad plantea desafíos en gestión. Este artículo analiza cómo el ML aborda problemas como congestión y latencia, implementando modelos con herramientas como Scikit-learn y Tensor Flow para optimizar el rendimiento[32].

2.3. Marco Legal

El marco legal regulatorio de las telecomunicaciones y las redes inalámbricas en quinta generación (5G) es fundamental para garantizar un desarrollo eficiente y sostenible en la gestión de recursos de red. Las políticas y regulaciones establecidas por organismos nacionales e internacionales que promueven la investigación y la implementación de tecnologías que optimicen la gestión de recursos en los equipos proveedores de internet.

2.3.1. Regulaciones Internacionales

Las regulaciones de telecomunicaciones a nivel internacional son realizadas por la UIT (Unión Internacional de Telecomunicaciones) establece directrices, normas y recomendaciones sobre el uso del espectro y las tecnologías de telecomunicaciones. La UIT promueve el desarrollo de estándares que faciliten la interoperabilidad y eficiencia en redes 5G que es necesaria para abordar desafíos de control de recursos[5].

2.3.2. Ley Orgánica de las telecomunicaciones

En Ecuador la ley orgánica de telecomunicaciones (LOT) promulgada en 2015 es el ente principal legal que regula el uso de espectro radioeléctrico y las telecomunicaciones. La LOT establece que el espectro es un recurso natural de propiedad del estado, lo que implica que su gestión debe ser realizada de manera eficiente y equitativa constitucionalmente establecidos[33].

2.3.3. Agencia de Regulación y Control de las Telecomunicaciones

La Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL), fue creada mediante la Ley Orgánica de Telecomunicaciones publicada en el Registro Oficial 493, de 18 de febrero del 2015; encargada de la administración, regulación y control de las telecomunicaciones, así como de los aspectos técnicos de la gestión de medios de comunicación social que usen frecuencias del espectro radioeléctrico o que instalen y operen redes[34]. La ARCOTEL es una institución adscrita al Ministerio rector de las Telecomunicaciones y de la Sociedad de la Información (MINTEL).

CAPÍTULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1. Localización

La investigación se realizó a cabo en la Universidad Técnica Estatal de Quevedo situado en la Av. Carlos J. Arosemena 38, Quevedo - Los Ríos – Ecuador. con coordenadas geográficas de latitud -1.01248 y longitud -79.4692, a una altitud de 74 metros sobre el nivel del mar.

Ilustración 11. UTEQ en Google maps

Autor: Google Maps

3.2. Tipo de investigación

3.2.1. Investigación bibliográfica

Con una exhaustiva investigación de estudios y fuentes bibliográficas relacionadas al uso de aprendizaje automático en redes de telecomunicaciones para análisis y optimización de la red que generen un impacto en redes actuales. Esto incluye artículos científicos, libros, tesis, proyectos de investigación y conferencias que aborden estos temas dentro del marco referencial como un aporte a mi investigación.

3.2.2. Investigación aplicada

Aplicar conceptos e información existente para resolver problemas específicos dentro del funcionamiento principal de una red inalámbrica para determinar los factores que generen retraso en su funcionamiento general, mediante el uso de ML se reducirá esos elementos que influyen en la pérdida de datos.

3.2.3. Investigación descriptiva

Permite caracterizar el comportamiento de las redes 5G respecto a latencia bajo diferentes condiciones operativas. Se documentará las métricas de desempeño obtenidas en las simulaciones, describiendo cómo varían según la congestión de tráfico, asignación de recursos y respuesta a situaciones críticas. Este tipo de investigación ayuda a visualizar de manera estructurada el funcionamiento de red sin intervención.

Este tipo de investigación se centrará en describir las características de modelos de aprendizaje automático elegidos para realizar pruebas donde se consideran factores como:

- **Tiempos de respuesta:** se observará el tiempo que posee cada modelo en procesar y ejecutar las funciones primordiales de una red inalámbrica.
- **Comportamiento bajo niveles altos de tráfico:** se analizará la estabilidad y control de cada modelo al estar sometido con una gran capacidad de demanda en su red.
- **Adaptabilidad en la red:** Cada modelo llevara un porcentaje de éxito en su desempeño para determinar su eficiencia.
- **Eficiencia energética:** basado en el uso de recursos que genere el modelo se llega a conclusiones de cual es más eficiente con la red sin sobrecargar los equipos.

3.2.4. Investigación exploratoria

Obtener un panorama general sobre el uso del aprendizaje automático en redes 5G a través de la revisión bibliográfica de artículos científicos, libros y reportes técnicos, se identifican tendencias, técnicas relevantes y vacíos de conocimiento que justifican el desarrollo del proyecto.

3.3. Métodos de investigación

3.3.1. Método cuasi experimental

Aplicando el efecto de los modelos de aprendizaje automático sobre la latencia en redes 5G, sin necesidad de intervenir en redes reales. Desarrollando escenarios simulados donde al implementar algoritmos supervisados y no supervisados compararen resultados con redes sin optimización. Aunque no tenga participación en un entorno físico controlado con

el uso de Google colab para variables y observación del modelo de ML sobre las métricas de red, especialmente latencia, throughput y uso eficiente de recursos.

3.3.2. Método descriptivo

Utilizado para caracterizar el comportamiento de las redes 5G en condiciones con y sin modelos de aprendizaje automático. Permitirá documentar cómo varían indicadores como latencia, congestión o eficiencia energética bajo distintos niveles de tráfico, densidad de usuarios o configuración de red. A través del análisis de resultados obtenidos en las simulaciones, logrando describir de manera detallada cómo influyen los algoritmos en el rendimiento general de la red.

3.3.3. Método bibliográfico

Enfocado en la revisión y análisis de fuentes científicas, artículos, libros y estudios previos relacionados con la aplicación del aprendizaje automático en telecomunicaciones, especialmente en redes 5G. Esta revisión permitirá establecer el marco teórico, identificar los enfoques más utilizados, seleccionar los algoritmos más adecuados y justificar la importancia de aplicar técnicas de ML para optimizar la latencia y la gestión de recursos en entornos inalámbricos avanzados.

3.4. Fuentes de recopilación de información

La recopilación de información es un proceso crucial para la construcción de cualquier investigación ya que permite obtener datos relevantes y verificados que respaldan las hipótesis y objetivos en el proyecto. En este trabajo se utilizaron dos tipos de fuentes de investigación que son las fuentes primarias y fuentes secundarias, cada una con un propósito en específico para aportar en la simulación de la red 5G y el uso de los modelos de aprendizaje automático.

3.4.1. Fuentes primarias

Uno de los recursos fundamentales en esta tesis fue la simulación de red 5G realizada completamente en MATLAB. Los datos obtenidos de la topología malla de routers, junto con las métricas de rendimiento como latencia, throughput, pérdida de paquetes y consumo energético, fueron la base para evaluar el comportamiento de la red y probar la eficacia de las técnicas de aprendizaje automático. Estos datos de simulación primaria permitieron realizar un análisis exhaustivo de las condiciones de congestión en la red, la asignación de recursos y la optimización de tráfico.

3.4.2. Fuentes secundarias

Se revisaron artículos científicos de renombradas revistas y conferencias sobre redes 5G y el uso de Machine Learning (ML) en telecomunicaciones, tales como los publicados en la IEEE Communications Magazine y IEEE Transactions on Networking. Estos artículos fueron fundamentales para conocer las últimas tendencias tecnológicas y los avances en redes 5G, además de proporcionar ejemplos de aplicaciones de aprendizaje automático en la gestión de redes inalámbricas. Además, se consultaron informes técnicos emitidos por instituciones de telecomunicaciones de prestigio como la ITU (International Telecommunication Union) y la 3GPP (3rd Generation Partnership Project), que proporcionaron información sobre los requisitos técnicos de las redes 5G, los estándares de latencia y las especificaciones de rendimiento que se utilizaron para establecer los objetivos de la simulación.

3.5. Fase de simulación

La presente investigación se organiza en cuatros fases que desarrollan y evalúan la aplicación de técnicas de ML en redes 5G. Para abordar este desafío se han aplicado modelos de machine learning en el enrutamiento dinámico de las redes optimizando la asignación de recursos y mejorando la eficiencia de las comunicaciones en tiempo real. Este proceso se lleva a cabo en cuatro fases principales: la recopilación de información, la simulación del escenario de red 5G, el entrenamiento de los modelos de ML y la validación y presentación de resultados.

3.5.1. Fase 1: Recopilación de información

Es crucial recopilar información relevante sobre las tecnologías 5G actuales, sus topologías de red y las técnicas de aprendizaje automático aplicadas en redes de telecomunicaciones. Esta fase inicial es clave para establecer la base teórica y asegurar que la investigación esté alineada con los enfoques más efectivos y recientes.

- **Fuentes de Información:** La recopilación de información se llevó a cabo mediante una exhaustiva revisión de artículos científicos, libros, tesis y publicaciones especializadas que abordan:

  Latencia en redes 5G: Se identificaron estudios previos sobre la reducción de latencia en redes de quinta generación, como el trabajo de Zhang (2019) y Rodríguez (2023).

  Topologías de Red: Se exploraron diferentes tipos de topologías de red (punto a punto, en estrella, malla, etc.) y se compararon sus ventajas y desventajas, con especial énfasis en la topología malla debido a su resiliencia y redundancia.

  Técnicas de Aprendizaje Automático: Se revisaron las principales técnicas de ML, como Redes Neuronales Artificiales (ANN), Máquinas de Vectores de Soporte (SVM) y Random Forest, y su aplicación en la gestión de tráfico en redes 5G.
- **Justificación de la Topología Malla:** La topología malla fue seleccionada como la estructura base para la simulación de la red debido a su alta fiabilidad, capacidad para soportar congestiones de tráfico y su resiliencia ante fallos. En redes 5G densas, la redundancia proporcionada por la malla es crucial, ya que permite que la red se recupere rápidamente de posibles fallos de enlace y mantiene un rendimiento estable.
- **Selección de Modelos de Aprendizaje Automático:** Para la reducción de latencia y la gestión dinámica de recursos, se seleccionaron los modelos supervisados más adecuados, como Máquina vectores de soporte y Redes Neuronales Artificiales, que permiten realizar predicciones de congestión y optimizar la asignación de recursos en tiempo real. La revisión de literatura reciente (Shukla (2024)) permitió validar la efectividad de estos enfoques en redes 5G.

3.5.2. Fase 2: Fase de Simulación de Tráfico y Generación de Datos

Herramienta utilizada: MATLAB

- **Objetivo de la fase:** Desarrollar un escenario de red 5G realista con topología malla para simular tráfico y estudiar cómo la latencia y throughput se ven afectados por diferentes niveles de congestión.
- **Metodología:** Utilizando MATLAB, se creó una topología malla de 15 routers interconectados, con 100 dispositivos con valores variados entre teléfonos y computadores en los routers. Esto permitió simular las condiciones de una red densa y evaluar el comportamiento de la latencia, pérdida de paquetes y consumo energético en función de la congestión.
- **Detalles del código de simulación:** Generación de tráfico: Se utilizó un modelo Poisson para simular el tráfico entre dispositivos, y los dispositivos conectados aumentaron progresivamente durante la simulación.
- **Cálculo de métricas:** Se evaluaron métricas clave como latencia y throughput, basadas en el tráfico generado en cada dispositivo y la capacidad de los routers para manejar la carga.
- **Por qué se eligió la topología malla:** La topología malla fue seleccionada por su capacidad de redundancia y resiliencia, características esenciales para redes 5G que requieren de alta disponibilidad y rendimiento continuo. La malla permite múltiples rutas entre los dispositivos, lo que mitiga la congestión en entornos densos y optimiza la distribución del tráfico.

3.5.3. Fase 3: de Entrenamiento de Modelos de Aprendizaje Automático

Herramienta utilizada: MATLAB

- **Objetivo de la fase:** Entrenar modelos de aprendizaje automático para predecir y gestionar la congestión en la red 5G simulada utilizando Random Forest para medir correctamente las métricas, Redes Neuronales Artificiales y SVM para optimizar la red.
- **Metodología:** Entrenamiento supervisado: Los modelos de ML fueron entrenados con un conjunto de datos de tráfico generado en MATLAB. Se utilizó implementación de los modelos con Scikit-learn.

- **Validación cruzada:** Se aplicó validación cruzada para evaluar la precisión y el rendimiento de los modelos.
- **Selección de algoritmos:** Random Forest y ANN fueron seleccionados debido a su capacidad para predecir congestionamientos y optimizar la asignación de recursos en tiempo real. Ambos algoritmos fueron entrenados para clasificar condiciones de congestión en la red (normal, alerta, crítico).
  SVM se usó para clasificación binaria en condiciones específicas de congestión.

3.5.4. Fase 4: Evaluación de Resultados y Optimización

- **Objetivo de la fase:** se evalúan los resultados generados por la simulación. Los modelos de aprendizaje automático predicen la condición de congestión para cada router, lo que permite tomar decisiones dinámicas sobre la asignación de recursos y la optimización de la latencia.
- **Predicción de Congestión:** Utilizando los modelos de aprendizaje automático entrenados, se predicen las condiciones de la red (normal, alerta, crítico) y se ajustan los recursos de la red en consecuencia.
- **Visualización de Resultados:** Se generan gráficos para evaluar el rendimiento de la red, comparando métricas clave como latencia y throughput bajo diferentes condiciones de congestión.

3.6. Recursos

Tabla 1. recursos utilizados en simulación

Recurso	Uso en la Simulación	Librerías / Toolbox
MATLAB	Utilizado para la creación de la topología malla, la simulación de tráfico y el entrenamiento de modelos de aprendizaje automático (ML).	MATLAB base software
MATLAB 5G Toolbox	Se usa para simular escenarios 5G realistas, como la gestión del tráfico, el cálculo de latencia y la implementación de modelos de enrutamiento.	5G Toolbox
Simulink	Se utilizaría para modelar sistemas complejos, pero no se emplea directamente en esta simulación, ya que se realiza toda la simulación en el código MATLAB.	Simulink (si es necesario en futuras simulaciones)
Random Forest	Se usa para clasificar el estado de los routers (Normal, Alerta, Crítico) basado en métricas como latencia, throughput, y pérdida de paquetes.	Statistics and Machine Learning Toolbox
Poisson Distribution	Se usa para modelar el tráfico aleatorio generado por los dispositivos (PCs y teléfonos) en la simulación de la red 5G.	- Statistics and Machine Learning Toolbox
Graph Theory Functions	Se utiliza para crear y manipular la topología malla de la red 5G y realizar análisis de conectividad y rutas entre los routers y dispositivos.	- MATLAB base software

Autor: Jordy Lucas.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. Modelos según estado del arte

Latencia en redes de quinta generación es uno de los principales desafíos porque afecta al rendimiento general de la red, la evaluación de los modelos de aprendizaje automático (ML) requiere de métricas específicas que reflejen el desempeño en términos de optimización de recursos y calidad de servicio. Cada métrica tiene un propósito en específico que representa un factor muy importante para comprender la viabilidad y éxito de cada modelo de ML. Las métricas seleccionadas son esenciales para evaluar de manera objetiva en el contexto de las redes 5G, la latencia media y tasa de congestión son fundamentales para medir la mejora directa en el rendimiento de la red, mientras que el throughput y eficiencia energética permiten realizar la optimización global y el uso de recursos. La tasa de error de predicción asegura que los modelos sean efectivos y confiables en su tarea, al aplicar estas métricas seleccionar el modelo de ML más adecuado para reducir latencia en redes 5G al contribuir y mejorar el rendimiento de la red.

Tabla 2. Métricas de evaluación para ML

Métrica	Descripción	Unidad de medida	Objetivo
Latencia media	Tiempo promedio de transmisión de datos desde el origen hasta el destino	Milisegundos (ms)	Medir el impacto de los modelos de aprendizaje automático en la reducción de latencia
Tasa de congestión	Porcentaje de tiempo que la red está congestionada o sobrecargada	Porcentaje (%)	Evaluar cómo los modelos predicen y mitigan la congestión de red
Throughput	Cantidad total de datos transmitidos correctamente en un período determinado	Mbps o Gbps	Medir el rendimiento global de la red y la eficiencia de los modelos de ML
Eficiencia energética	Cantidad de energía consumida por bit transmitido en la red	Joules por bit (J/bit)	Evaluar la eficiencia de los modelos ML en términos de uso energético
Tasa de error de predicción	Precisión de las predicciones del modelo en términos de congestión o latencia	Porcentaje (%)	Evaluar la exactitud de las predicciones generadas por el modelo

Autor: Jordy Lucas

Para evaluar el desempeño de los algoritmos aplicados al enrutamiento dinámico se considera como métricas claves la tasa de congestión y tasa de error de predicción para determinar qué modelo es el más eficiente.

La tasa de congestión permite medir el nivel de saturación presentado en los routers en relación con su capacidad máxima, en la simulación cada router soporta hasta 70 dispositivos estableciéndose un umbral de congestión cuando el número de conexiones supera el 85% de dicha capacidad que se expresa:

Fórmula 1. Cálculo tasa de congestión

$$Tasa_{cong}(\%) = \frac{N_{rc}}{Nrt} \text{ x } 100$$

N_{rc} corresponde al número de routers congestionados, Nrt representa el total de routers en la red. Una menor tasa de congestión indica una mejor distribución del tráfico entre los nodos para una mejor eficiencia en el enrutamiento.

La tasa de error de predicción cuantifica la capacidad del modelo de clasificación de los modelos para identificar correctamente los estados operativos de los routers sea normal, alerta o crítica para determinar desde el número de predicciones incorrectas durante el proceso de validación cruzada y su fórmula se presenta como:

Fórmula 2. Cálculo Error de predicción

$$Error_{pred}(\%) = \frac{N_{pi}}{N_{pt}}\ x\ 100$$

N_{pi} representa las predicciones incorrectas y N_{pt} el total de predicciones a partir, a partir de esos valores se define la eficacia del modelo con:

Fórmula 3. Cálculo eficacia del modelo

$$Eficacia(\%) = 100 - \ Error_{pred}(\%)$$

4.1.1. Modelos de aprendizaje automático

Para reducir la latencia en la red se puede aplicar diversas técnicas de aprendizaje automático (ML), cada modelo posee características idóneas para tareas específicas como la predicción de congestión, optimización de la asignación de recursos o la mejora en la eficiencia de la red. Es crucial seleccionar el modelo adecuado para cada tarea basada en la naturaleza de los datos y características de la red de quinta generación, entre los modelos usados se encuentran:

- **Máquina de Vectores de Soporte (SVM):** Utilizado para clasificación en espacios de alta dimensión, preciso para la detección de congestión y destaca en encontrar el límite (hiperplano) para separar clases. Requiere de un mayor ajuste de parámetros en su entrenamiento lo que permite realizar mejores funciones una vez aprendido lo esencial en su desarrollo.
- **Redes Neuronales Artificiales (ANN):** Modelo de aprendizaje profundo con alta capacidad para modelar datos no lineales y predecir en redes 5G, muestran un mayor rendimiento con mayores volúmenes de datos y son esenciales en generalización con entornos de múltiples variables.
- **Memoria a Corto Plazo (LSTM):** Modelo de aprendizaje profundo ideal para tareas que involucren dependencias temporales, lo que hace perfecto para predecir latencia o rendimiento a lo largo del tiempo. Diseñado para comprender tendencias de tráfico en incremento gracias a las secuencias de tiempo establecidos en su entorno.
- **Arboles de Decisión (RF):** Modelo de aprendizaje por conjunto que ofrece alta precisión y robusticidad al sobreajuste (exceso de parámetros) que permite un mejor control de las relaciones no lineales.
- **Vecinos más Cercanos (KNN):** Método de fácil comprensión e implementación gracias a su función de "Memorizar" los datos basados en instancia, clasifica basándose en la cercanía de otros puntos en el espacio de características.
- **Regresión Logística (LOGR):** Modelo estadístico rápido en entrenamiento y predicción por su detección de variables con mayor requerimiento.

Tabla 3. Modelos más recomendados según el estado del arte

Modelo	Tipo de aprendizaje	Ventajas	Desventajas
Árboles de Decisión (RF)	Supervisado (Aprendizaje por conjunto)	Alta precisión, robusto al sobreajuste y mejor control de relaciones no lineales	Lento para predecir y ocupa mayor memoria al almacenar mayor cantidad de arboles
Máquina vectorial de soporte (SVM)	Supervisado (Clasificador)	Alta precisión al encontrar mejor límite (hiperplano) para separar las clases, el enrutamiento es rápido una vez entrenado.	El entrenamiento es lento y consume muchos recursos que requiere de mayor ajuste de en parámetros
Memoria a corto plazo (LSTM)	Supervisado (Aprendizaje profundo)	Diseñado para entender tendencias de tráfico en incremento gracias a las secuencias de tiempo.	Bajo valor de predicción en escenarios con bajas secuencias y parámetros.
Vecinos más cercanos (KNN)	Supervisado (Basado en instancia)	Fácil entendimiento e implementación porque 'memoriza' los datos	Predicción lenta porque debe comprar el estado actual con toda la base de datos lo que provoca retraso.
Regresión Logística (LOGR)	Supervisado (Estadístico/Lineal)	Modelo rápido en entrenamiento y predicción gracias su detección de variables con mayor requerimiento.	Modelo lineal que puede provocar retraso en caso de operaciones simultaneas complejas.
Red neuronal (ANN)	Supervisado (Aprendizaje profundo)	Mayor capacidad de modelar datos por su entrenamiento neuronal	Entrenamiento lento debido a la toma de decisiones que debe realizar

Autor: Jordy Lucas

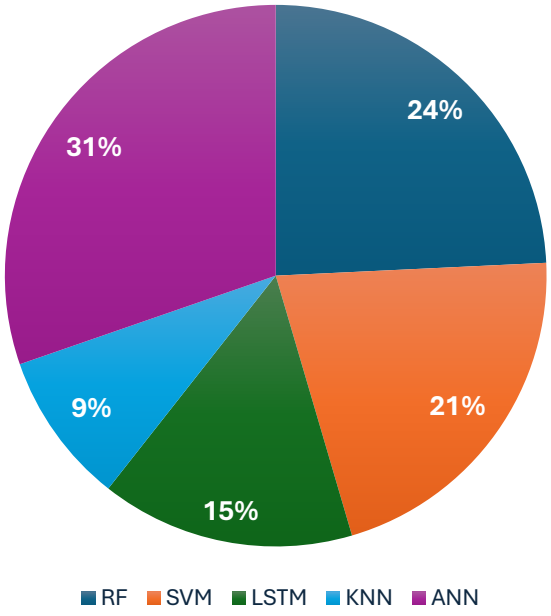
4.1.2. Discusión resultados en base al estado del arte

El análisis realizado sobre las métricas de latencia, throughput, pérdida de paquetes y consumo energético en redes 5G muestra que, a medida que se implementan modelos de aprendizaje automático, la optimización de latencia y gestión del tráfico mejoran considerablemente. Estos resultados están en línea con investigaciones previas como Shukla Neha (2024) que evidencian la capacidad de Random Forest para predecir la congestión y gestionar el tráfico en redes densas con alta eficiencia.

En la simulación realizada en este proyecto de investigación, los modelos de aprendizaje automático demostraron que la reducción de la latencia es posible gracias a su capacidad para ajustar dinámicamente los recursos de la red en función de las condiciones cambiantes del tráfico.

- El throughput es otro parámetro clave en redes 5G, ya que mide la capacidad de transmisión de datos. En este estudio, los modelos Random Forest y Redes Neuronales Artificiales (ANN) mostraron un aumento significativo en el throughput de la red al optimizar la asignación de recursos y la gestión del tráfico. Estos hallazgos respaldan estudios previos que destacan la importancia de maximizar el throughput para garantizar un alto rendimiento en redes 5G.
- La pérdida de paquetes es una métrica fundamental en redes 5G, ya que puede afectar gravemente la calidad de las comunicaciones. En esta investigación, los modelos de aprendizaje automático fueron efectivos para predecir y minimizar la pérdida de paquetes mediante ajustes dinámicos en la asignación de recursos. Aunque se observó un incremento en la pérdida de paquetes en redes con alto número de dispositivos conectados, el uso de Random Forest ayudó a mitigar este problema al hacer ajustes precisos en la gestión del tráfico.
- El consumo energético es un aspecto crucial, ya que las redes 5G deben ser eficientes en términos de energía. Los resultados de esta tesis muestran que los modelos de aprendizaje automático contribuyen a reducir el consumo energético al optimizar el uso de recursos de manera dinámica. Los modelos ajustaron el consumo energético principalmente en routers con menos tráfico, lo que mejora la sostenibilidad de la red sin comprometer su rendimiento.

Ilustración 12. Eficiencia de ML según estado del arte



Autor: Jordy Lucas

El gráfico muestra la eficacia de los modelos de aprendizaje automático utilizados en la optimización de redes 5G. ANN (Redes Neuronales Artificiales) es el modelo más eficaz con un 31%, destacándose por su capacidad para manejar tráfico dinámico en redes densas, como se confirma en estudios previos (Shukla, 2024). Le sigue Random Forest (RF) con 24%, eficaz en la predicción de congestión, aunque ANN mostró mayor rendimiento. SVM (21%), LSTM (15%) y KNN (9%) mostraron una eficacia menor, con SVM adecuado para clasificación, pero limitado en redes complejas. ANN demuestra ser el candidato modelo más prometedor para la gestión dinámica de tráfico en redes 5G, superando a los otros modelos según los resultados obtenidos en diferentes estudios.

4.2. Diseño de escenarios simulados

El diseño y simulación de escenarios en Matlab es fundamental para la fase de investigación, orientada a validar la hipótesis de que técnicas de aprendizaje automático en particular puede optimizar el rendimiento y reducir latencia en los entornos preparados. La estrategia metodológica se basa en dos pilares fundamentales los cuales son la evaluación comparativa de topologías para seleccionar la estructura base y validación del ML como enrutador dinámico sobre la topología elegida.

4.2.1. Análisis comparativo de topologías

El objetivo de esta fase consiste en determinar la estructura de red más eficiente para operar en un entorno de alta densidad previo a la implementación del ML. Se compararon tres topologías (Árbol, estrella y malla) utilizando un modelo base de 15 routers y distribución equilibrada de dispositivos para generar tráfico variable que permitan evaluar las métricas de rendimiento.

4.2.1.1. Elección de la topología

La elección de la topología de red es un factor clave para simular correctamente las condiciones de tráfico y latencia en las redes, mediante la comparación entre 3 topologías que son conocidas como malla, estrella y árbol, pero se toma mayor relevancia en la topología malla debido a sus características específicas que proporcionan redundancia y resiliencia en la red.

La topología malla fue elegida porque ofrece una alta fiabilidad y redundancia que permite establecer múltiples caminos entre routers, esta topología permite una mejor distribución y conexión ante posibles congestiones en los enlaces o celdas.

Tabla 4. comparativa entre topologías

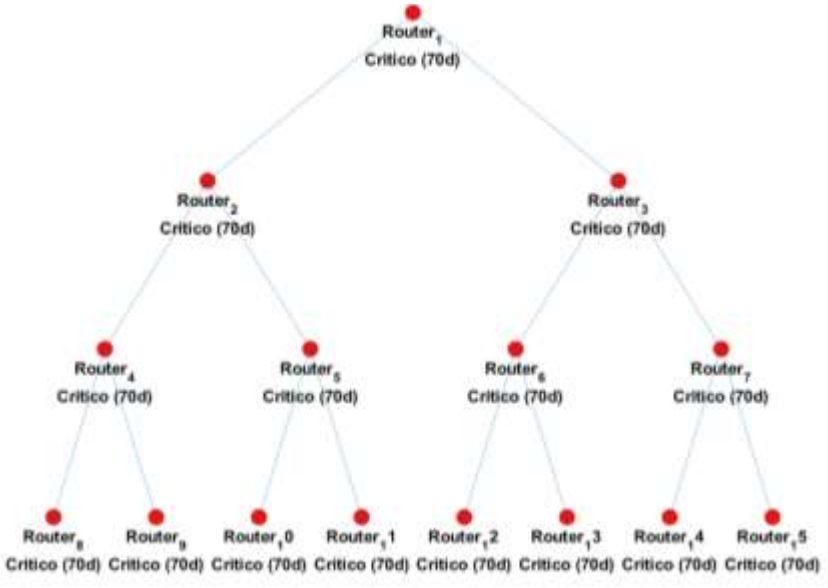
Topología	Descripción	Ventajas	Desventajas
Topología en Malla	Todos los routers están interconectados entre sí, proporcionando alta redundancia y resiliencia.	Alta fiabilidad, si un enlace falla, otros caminos toman el relevo, ideal para redes densas y críticas.	Mayor complejidad y coste debido al gran número de conexiones entre nodos.
Topología en Estrella	Un nodo central conecta todos los demás nodos, simplificando la gestión y estructura.	Fácil de gestionar, sencilla de implementar, adecuada para redes con dispositivos concentrados.	Dependencia del nodo central, si falla, toda la red se ve afectada.
Topología en Árbol	Estructura jerárquica donde un nodo raíz conecta otros nodos en una estructura jerárquica.	Eficiencia en la organización y escalabilidad.	La sobrecarga en el nodo raíz puede generar cuellos de botella.

Autor: Jordy Lucas

4.2.1.2. Resultados topología Árbol

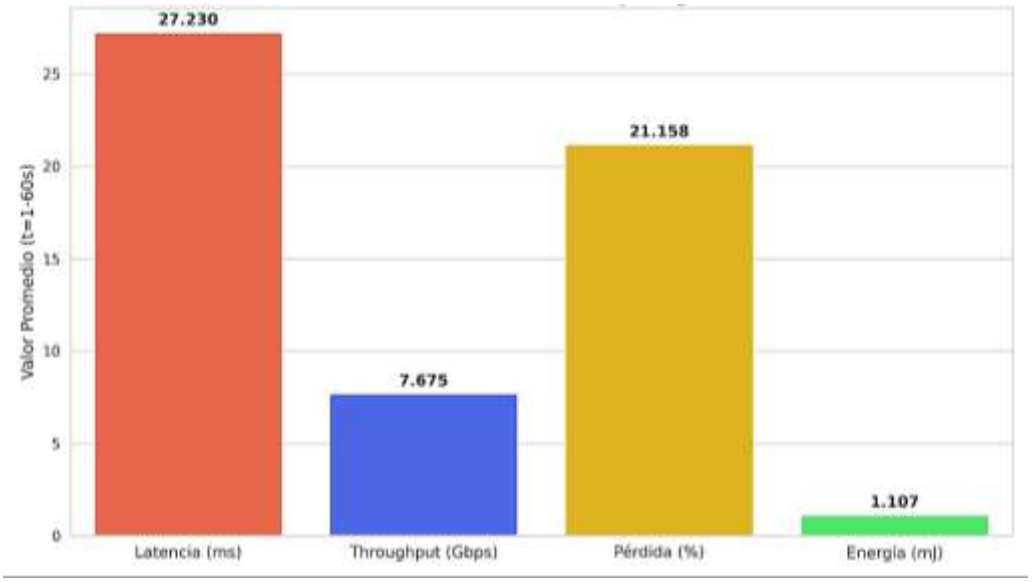
En la topología árbol la latencia promedio varió entre 274.78 ms y 325.52 ms que favorecen a la gestión del tráfico y mantienen un throughput alto (25.75 Gbps a 30.71 Gbps), se observó que latencia y consumo energético aumentaron en los routers con mayor cantidad de dispositivos conectados, resaltando la necesidad de optimización en la gestión de recursos.

Ilustración 13. Diseño de topología Árbol



Autor: Jordy Lucas

Ilustración 14. Resultados Generales de la topología árbol



Autor: Jordy Lucas

Tabla 5.Métricas del escenario árbol

Router	Laptops	Teléfonos	Dispositivos finales	Latencia_ms	Throughput_Gbps	Pérdida de paquetes	Energía_mJ
"Router_1"	37	33	70	21.404	6.2583	21.184	1.4209
"Router_2"	29	41	70	42.353	6.3998	42.277	1.4084
"Router_3"	35	35	70	43.505	6.3784	42.561	1.489
"Router_4"	30	40	70	41.865	6.339	42.269	1.4486
"Router_5"	29	41	70	42.967	6.2092	42.513	1.4051
"Router_6"	32	38	70	42.772	6.3503	42.688	1.4458
"Router_7"	33	37	70	42.913	6.2808	42.57	1.4518
"Router_8"	35	35	70	42.161	6.3427	41.346	1.4674
"Router_9"	37	33	70	42.006	6.3212	42.342	1.4619
"Router_10"	34	36	70	43.571	6.3197	42.682	1.4195
"Router_11"	35	35	70	44.101	6.3515	42.02	1.4152
"Router_12"	30	40	70	41.987	6.3733	41.559	1.4239
"Router_13"	32	38	70	42.596	6.3253	41.346	1.4484
"Router_14"	31	39	70	41.694	6.2177	41.377	1.4282
"Router_15"	38	32	70	42.258	6.3655	41.398	1.4989

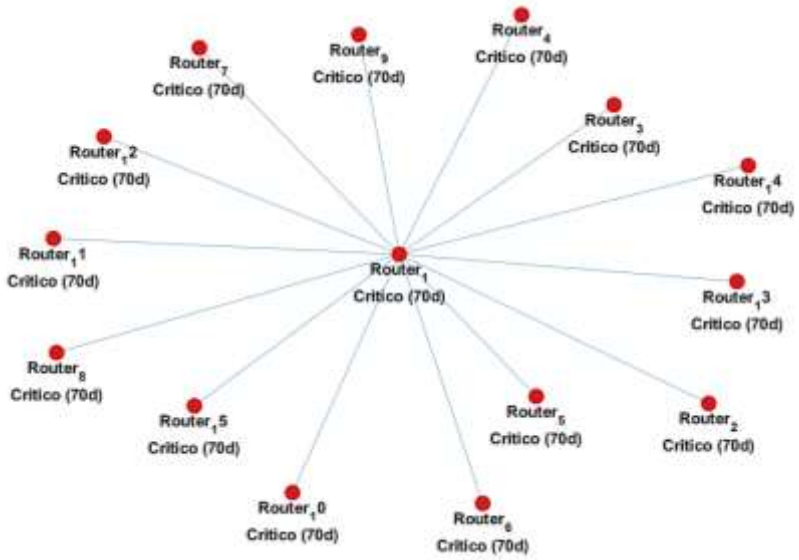
Autor: Jordy Lucas

4.2.1.3. Resultados topología Estrella

La topología estrella mostró el menor rango de latencia mínima con 225.64 ms, pero presentó un problema de cuello de botella en el router central debido a su diseño que concentra la carga, lo que resulta en mayor consumo energético y pérdida de paquetes,

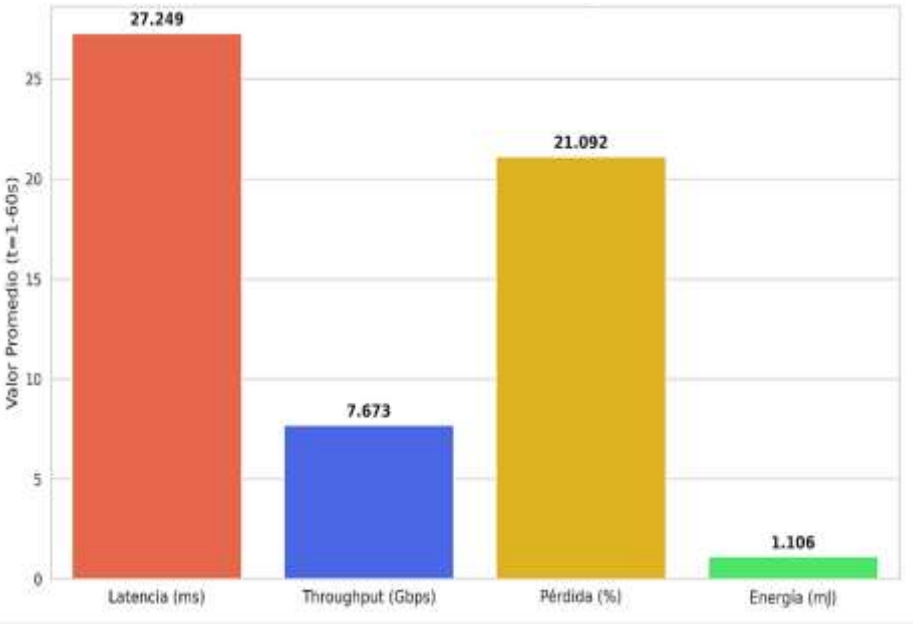
haciendo que la latencia y el rendimiento dependieran fuertemente de su rendimiento (router central).

Ilustración 15. Diseño de topología Estrella



Autor: Jordy Lucas

Ilustración 16. Resultados Generales de la topología estrella



Autor: Jordy Lucas

Tabla 6. Métricas del escenario estrella

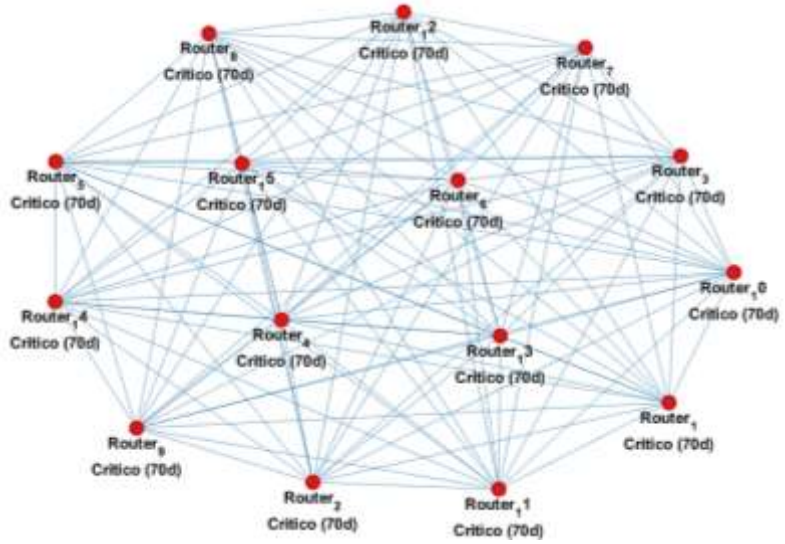
Router	Laptops	Teléfonos	Dispositivos Finales	Latencia_ms	Throughput_Gbps	Pérdida_paquetes	Energía_mJ
"Router_1"	31	39	70	21.13	6.2871	20.286	1.4903
"Router_2"	33	37	70	42.263	6.2392	40.599	1.4514
"Router_3"	39	31	70	42.421	6.3006	40.915	1.4525
"Router_4"	36	34	70	42.189	6.2182	40.954	1.4339
"Router_5"	33	37	70	41.943	6.3929	40.376	1.4667
"Router_6"	36	34	70	42.16	6.3209	40.382	1.4431
"Router_7"	35	35	70	43.202	6.3784	40.871	1.4393
"Router_8"	36	34	70	42.811	6.2735	40.322	1.4866
"Router_9"	35	35	70	42.751	6.2743	40.317	1.4383
"Router_10"	28	42	70	42.346	6.203	41.13	1.4455
"Router_11"	34	36	70	41.973	6.2992	40.895	1.4255
"Router_12"	30	40	70	42.035	6.3959	41.305	1.4908
"Router_13"	33	37	70	42.87	6.2392	41.486	1.4272
"Router_14"	30	40	70	42.947	6.33	41.017	1.4156
"Router_15"	34	36	70	42.279	6.2618	40.894	1.462

Autor: Jordy Lucas

4.2.1.4. Resultados topología Malla

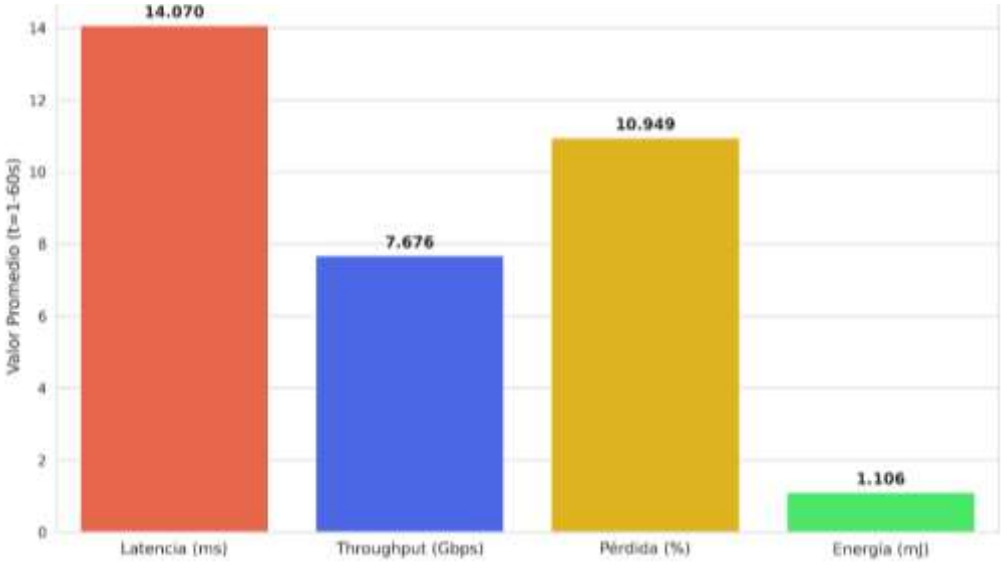
La topología malla con latencias entre 259.08 ms y 325.51 ms demostró una alta resiliencia(adaptación) y capacidad para manejar grandes volúmenes de datos con un rendimiento de 24.22 Gbps a 30.70 Gbps. El principal desafío fue la variación de latencia ante la congestión de tráfico en routers con muchos dispositivos que logró manejar hasta cierto punto debido a la complejidad de múltiples rutas.

Ilustración 17. Diseño de topología Malla



Autor: Jordy Lucas

Ilustración 18. Resultados generales de la topología Malla



Autor: Jordy Lucas

Tabla 7. Métricas del escenario malla

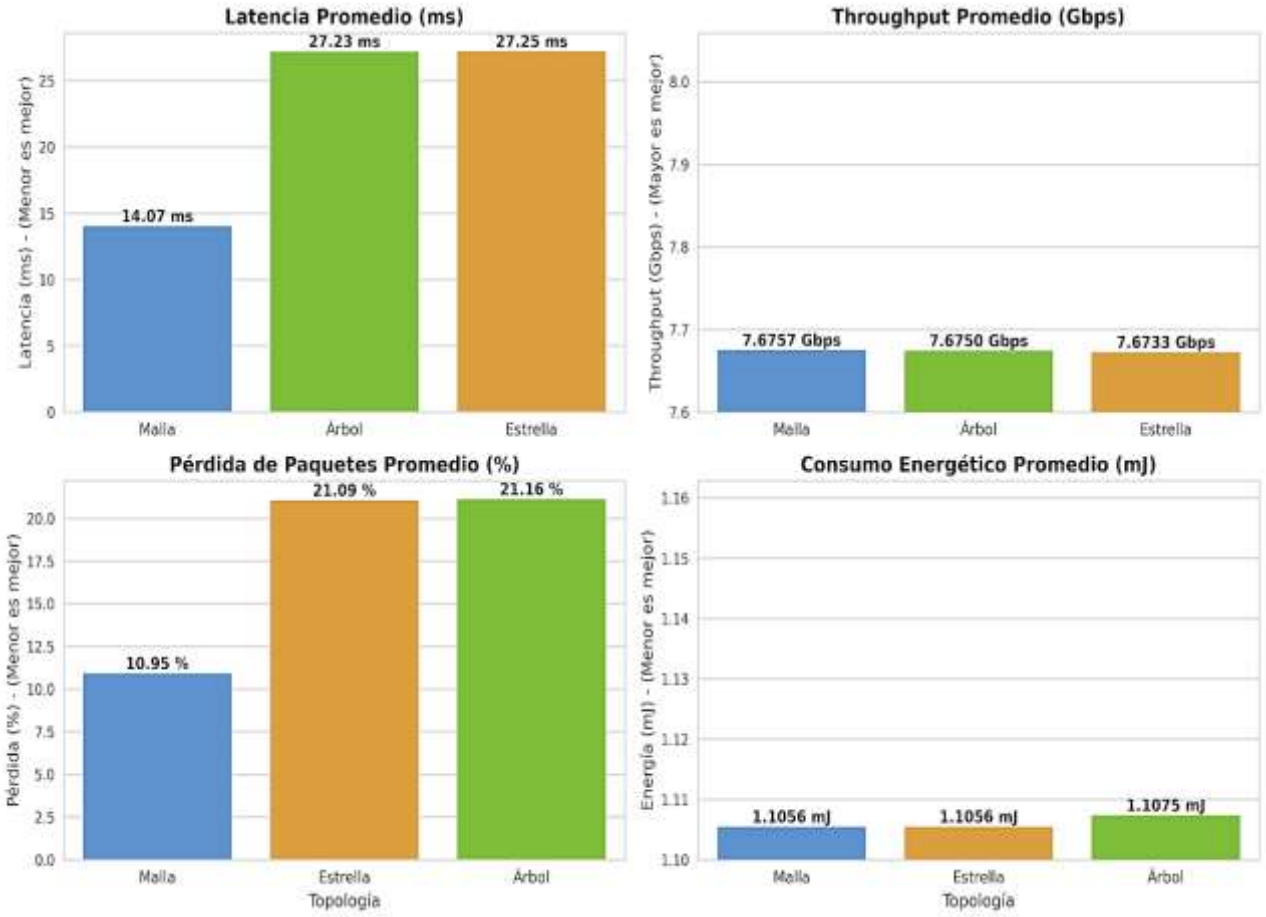
Router	Laptops	Teléfonos	Dispositivos Finales	Latencia_ms	Throughput_Gbps	Pérdida_paquetes	Energía_mJ
"Router_1"	38	32	70	21.251	6.2332	20.223	1.46
"Router_2"	38	32	70	20.747	6.2839	21.035	1.492
"Router_3"	34	36	70	21.923	6.3939	20.467	1.4869
"Router_4"	32	38	70	21.492	6.3908	20.794	1.475
"Router_5"	40	30	70	20.953	6.2424	20.46	1.4995
"Router_6"	37	33	70	21.112	6.2392	20.861	1.4327
"Router_7"	39	31	70	20.805	6.2251	21.441	1.4045
"Router_8"	37	33	70	21.794	6.2571	21.253	1.4124
"Router_9"	35	35	70	20.849	6.224	20.559	1.4645
"Router_10"	36	34	70	21.39	6.2764	21.167	1.4768
"Router_11"	41	29	70	21.011	6.3406	20.536	1.4114
"Router_12"	30	40	70	21.189	6.3782	20.211	1.4381
"Router_13"	34	36	70	20.855	6.3809	21.344	1.4587
"Router_14"	36	34	70	21.065	6.3986	21.366	1.4132
"Router_15"	40	30	70	21.025	6.2453	21.298	1.4305

Autor: Jordy Lucas

4.2.1.5. Resultados comparativo y elección de topología

La topología Malla fue seleccionada como la estructura base para la fase de optimización con ML, esta decisión fue en base a su alta fiabilidad, redundancia y adaptación. Latencia representa un gran reto debido a la congestión que se produce por la cantidad de dispositivos conectados simultáneamente, las redes 5G densas que requieren alta disponibilidad y rendimiento continuo.

Tabla 8. Gráficos comparativos generales de topologías



Autor: Jordy Lucas

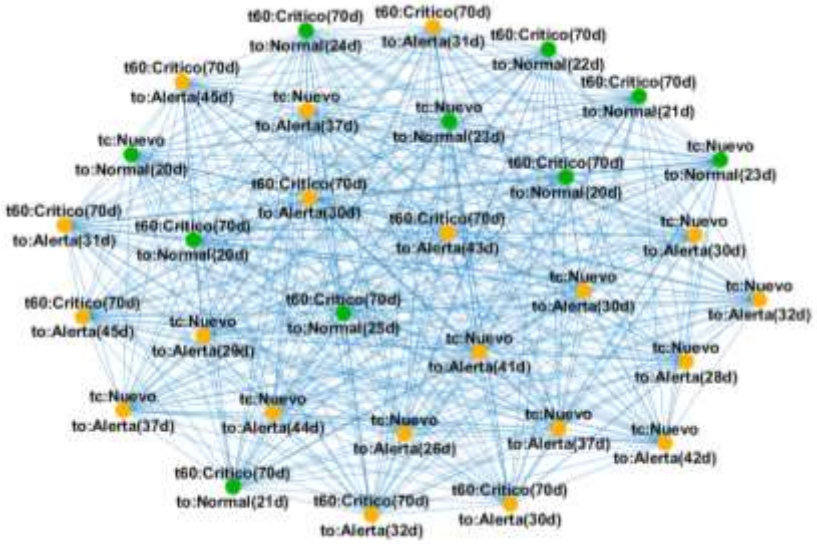
En base al gráfico se puede determinar que la topología malla como base de la simulación de optimización porque demostró ser la más eficiente alcanzando la menor latencia promedio con 14.07 ms y menor perdida de paquetes con 10.95% en comparación con otras topologías como el árbol y estrella que dieron el doble de estos valores. Latencia y perdida de paquetes son métricas críticas en 5G que reflejan la resiliencia y fiabilidad de la red, aunque estas las tres topologías mantuvieron un rendimiento alto y un consumo energético similar, la superioridad de la malla respecto a esas 2 métricas justifican su elección.

4.2.2. Resultados de escenarios con modelos de aprendizaje automático

4.2.2.1. Resultado simulación de red malla con ANN

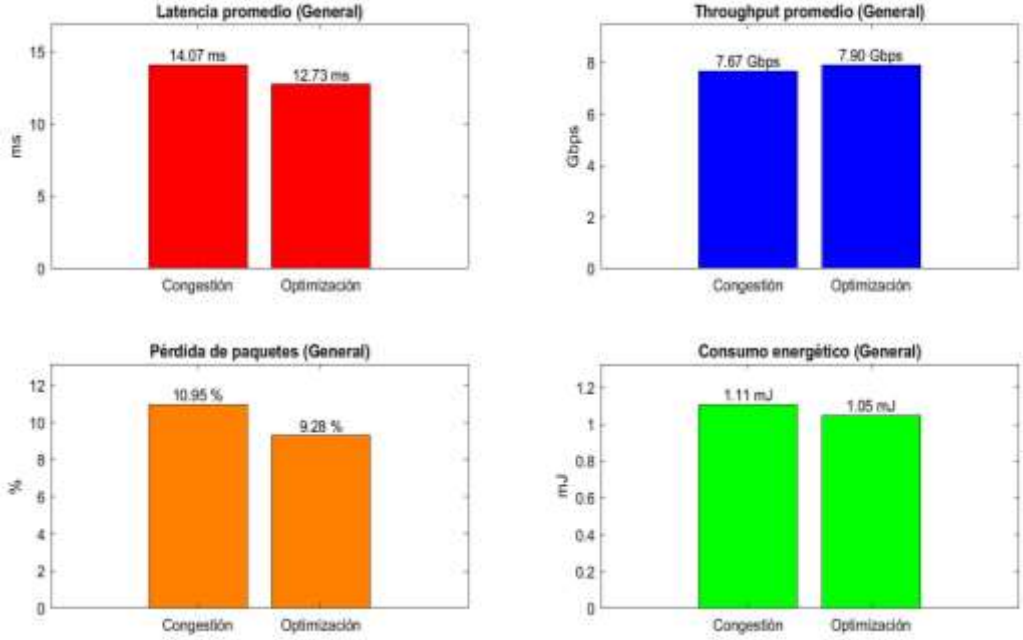
La red neuronal artificial (ANN) demostró una alta capacidad de clasificación con una eficacia de validación del 98.36% y un error mínimo del 1.64% en su toma de decisiones como enrutamiento dinámico.

Ilustración 19. Estado final de topología malla con ANN



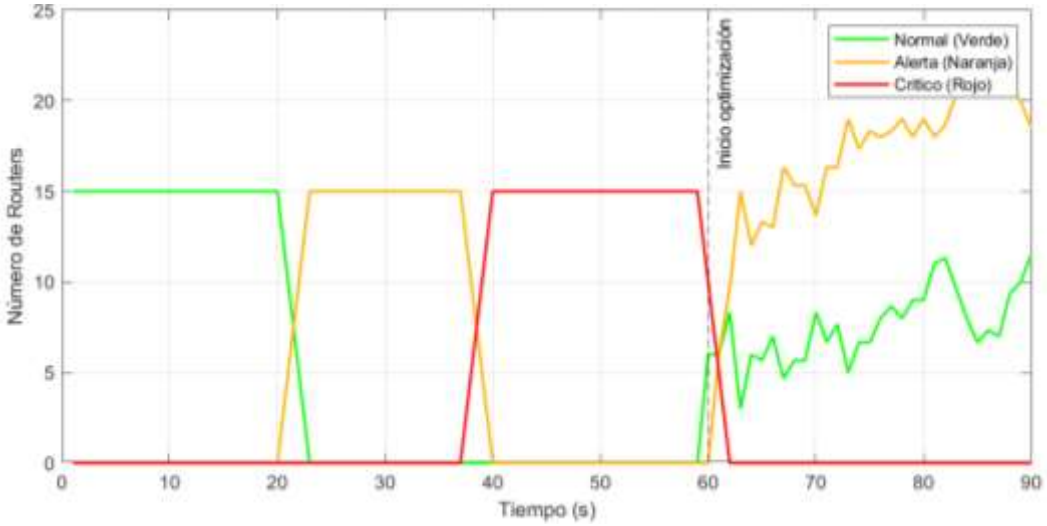
Autor: Jordy Lucas

Ilustración 20. Resultados de simulación ANN



Autor: Jordy Lucas

Ilustración 21. Evolución de estados en los routers con ANN



Autor: Jordy Lucas

Tabla 9. Métricas de eficiencia ANN

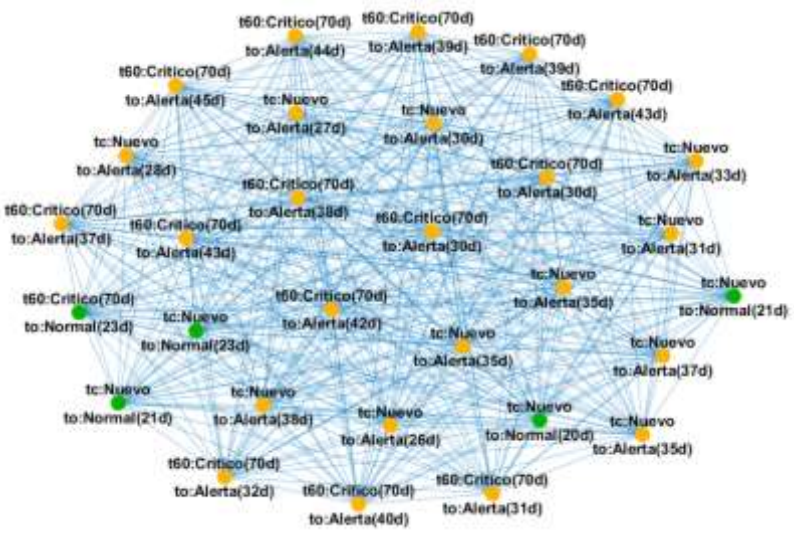
Métrica de eficiencia	Valor
Reducción de latencia (%)	9.5746
Reducción pérdida de paquetes (%)	15.229
Eficacia de validación (%)	98.361
Error de validación (%)	1.6393

Autor: Jordy Lucas

4.2.2.2. Resultado simulación de red malla con KNN

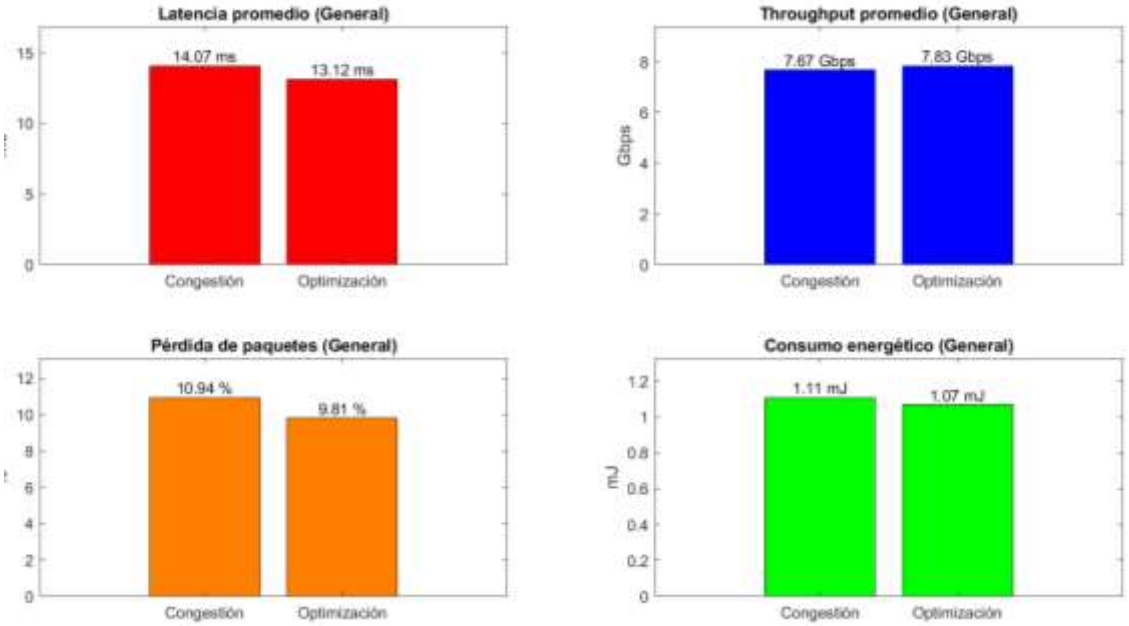
El modelo Vecinos más cercanos (KNN) mostró un rendimiento notable en la validación con una eficacia del 97.87% y un error del 2.13%, el problema radica en su alto tiempo de enrutamiento del modelo basado en instancia que impacta negativamente en latencia real.

Ilustración 22. Estado final de topología malla con KNN



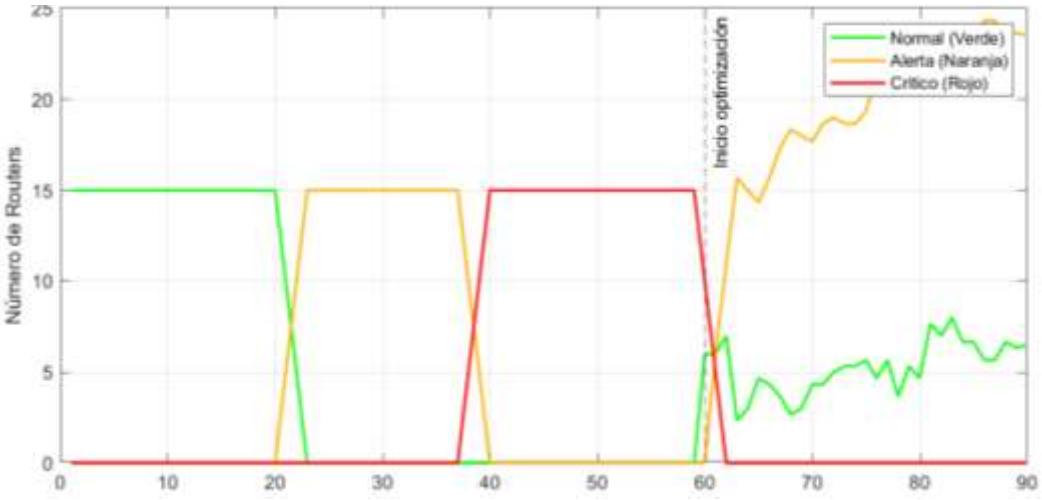
Autor: Jordy Lucas

Ilustración 23. Resultados de simulación KNN



Autor: Jordy Lucas

Ilustración 24. Evolución de estados en los routers con KNN



Autor: Jordy Lucas

Tabla 10. Métricas de eficiencia KNN

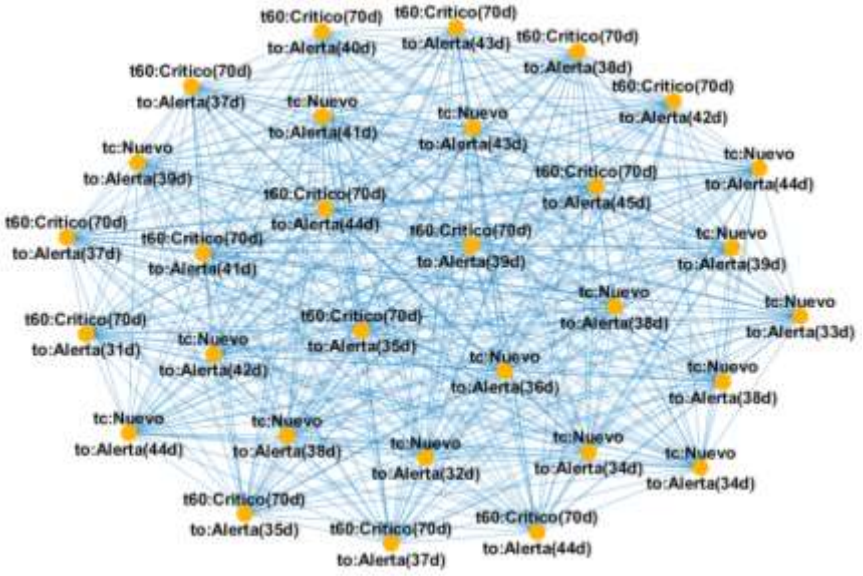
Métrica de eficiencia	Valor
Reducción de latencia (%)	6.7693
Reducción pérdida de paquetes (%)	15.229
Eficacia de validación (%)	98.361
Error de validación (%)	1.6393

Autor: Jordy Lucas

4.2.2.3. Resultado simulación de red malla con LOGR

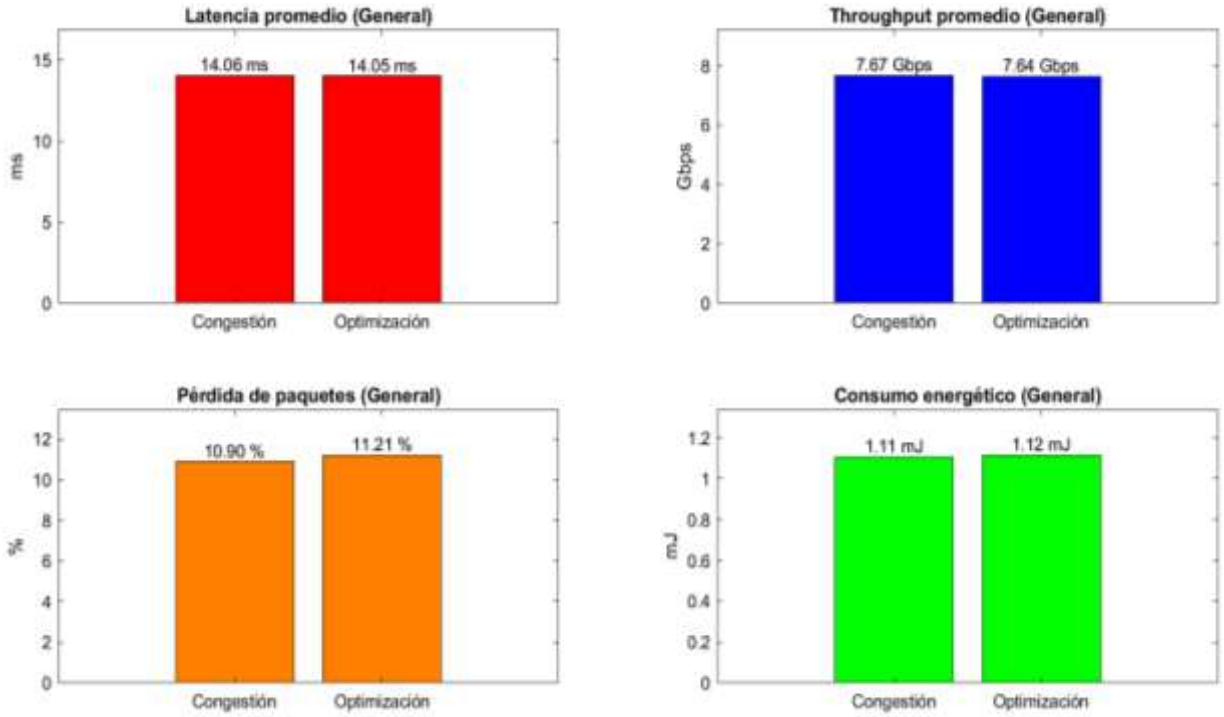
El modelo de regresión logística (LOGR) tuvo el desempeño más bajo en la clasificación con una eficacia del 92.62% y un error más alto del 7.38% entre los modelos comparados, este rendimiento refleja una optimización marginal entre todos los modelos.

Ilustración 25. Estado final de topología malla con LOGR



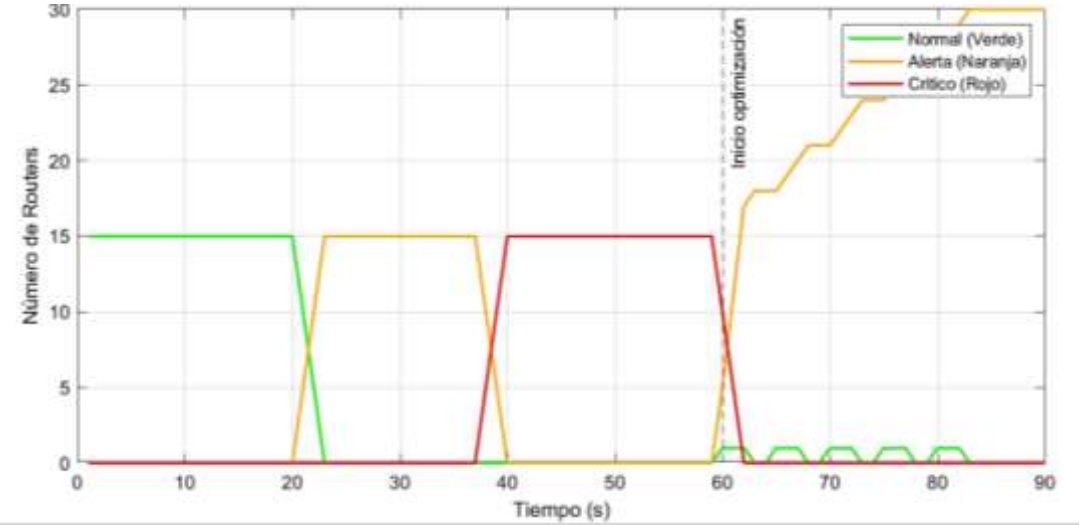
Autor: Jordy Lucas

Ilustración 26. Resultados de simulación LOGR



Autor: Jordy Lucas

Ilustración 27. Evolución de estados en los routers con LOGR



Autor: Jordy Lucas

Tabla 11. Métricas de eficiencia LOGR

Métrica de eficiencia	Valor
Reducción de latencia (%)	0.0679
Reducción pérdida de paquetes (%)	-0.422
Eficacia de validación (%)	-2.793
Error de validación (%)	-0.749

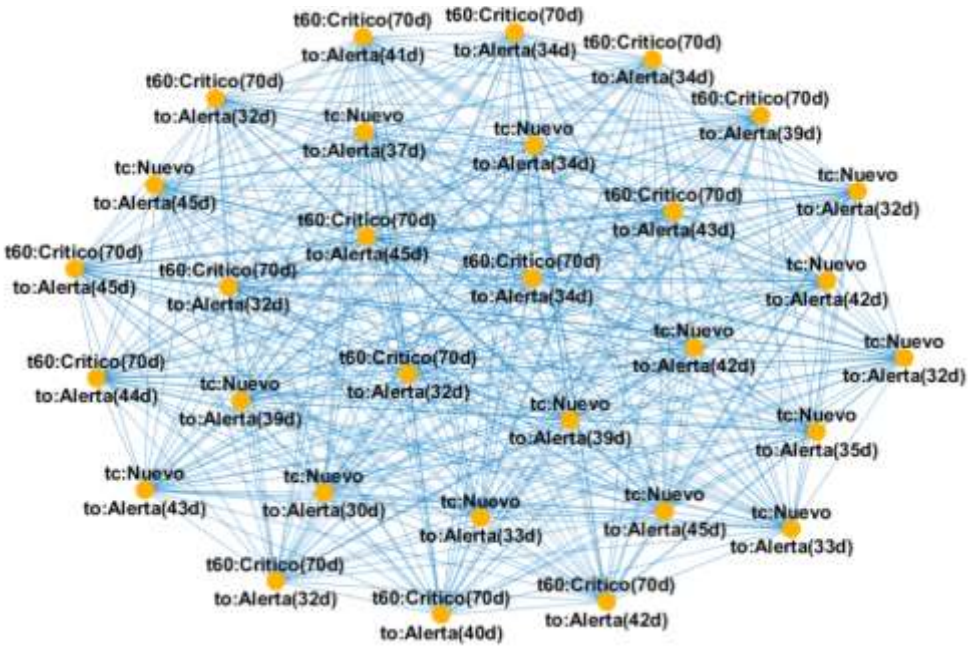
Autor: Jordy Lucas

LOGR mostró un resultado deficiente con una reducción en latencia casi nula (0.07%) y muy bajo rendimiento con pérdida de paquetes en valores negativos, esto indica que su naturaleza lineal no es adecuada para el enrutamiento dinámico complejo en redes 5G.

4.2.2.4. Resultado simulación de red malla con LSTM

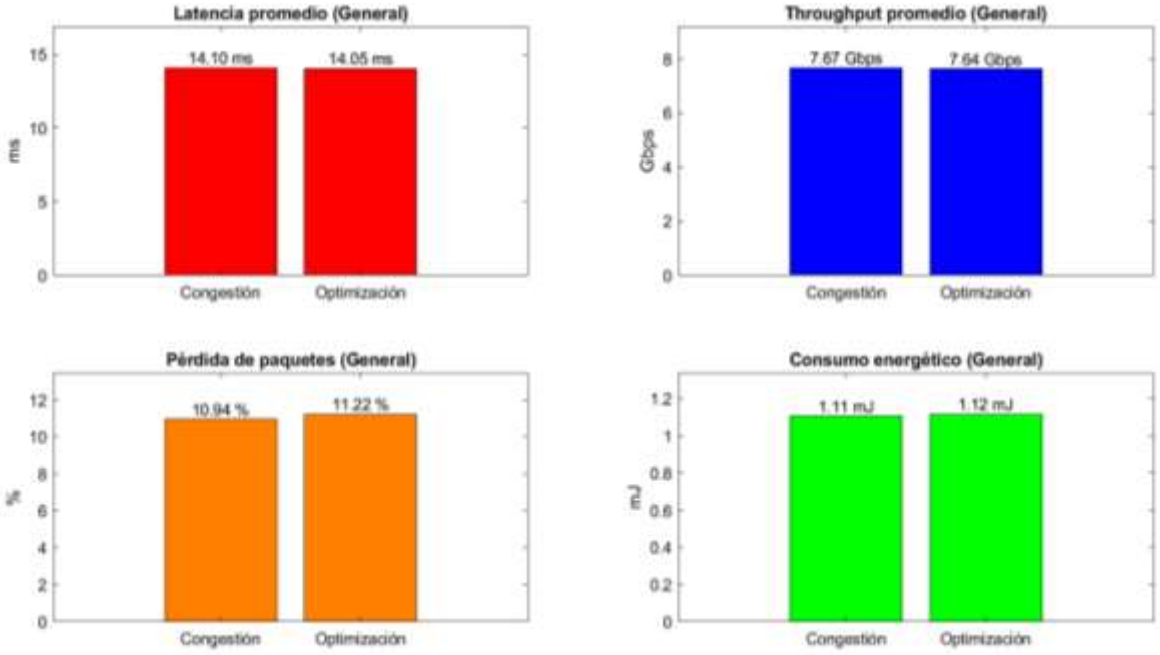
El modelo de memoria a corto y largo plazo (LSTM) demostró una alta precisión en su validación con una eficiencia del 95.90% y error del 4.10%, siendo el error un porcentaje considerable a determinar por su tiempo de respuesta como enrutador.

Ilustración 28. Estado final de topología malla con LSTM



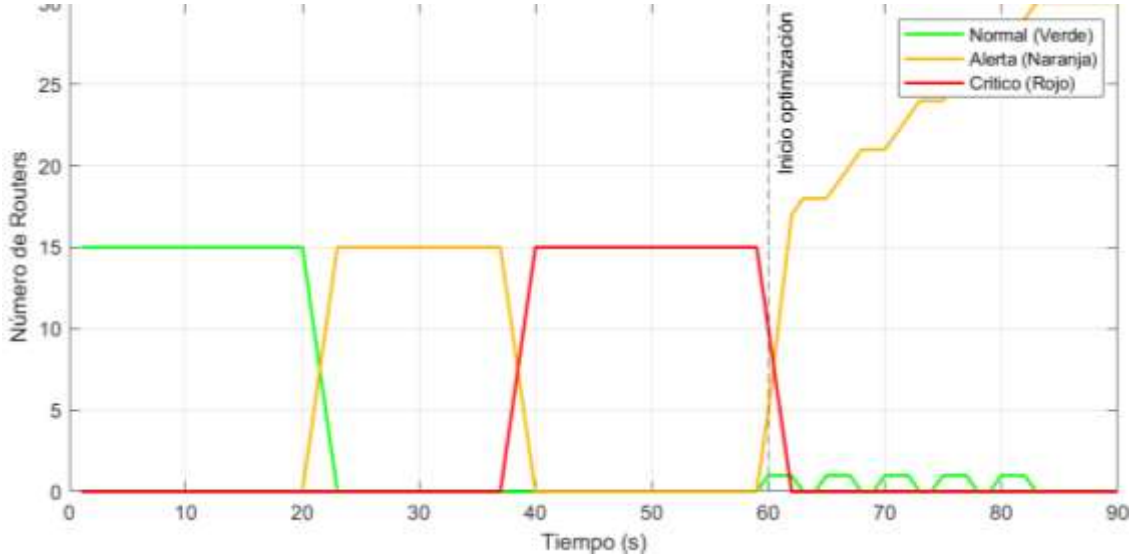
Autor: Jordy Lucas

Ilustración 29. Resultados de simulación LSTM



Autor: Jordy Lucas

Ilustración 30. Evolución de estados en los routers con LSTM



Autor: Jordy Lucas

Tabla 12. Métricas de eficiencia LSTM

Métrica de eficiencia	Valor
Reducción de latencia (%)	0.345
Reducción pérdida de paquetes (%)	-0.461
Eficacia de validación (%)	-2.559
Error de validación (%)	-0.930

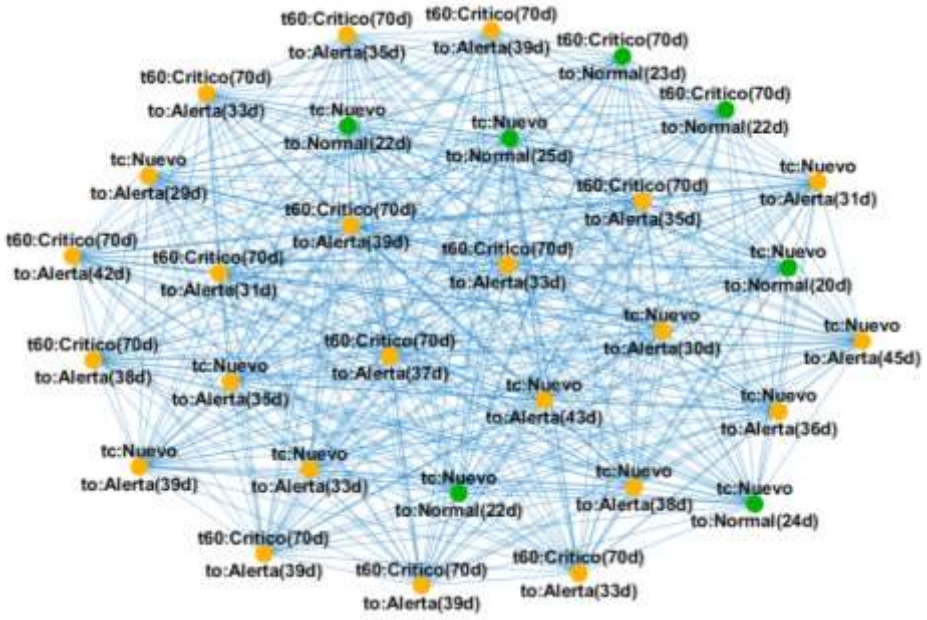
Autor: Jordy Lucas

LSTM logró una reducción mínima de latencia del 0.35%, pérdida de paquetes y consumo de energía bastante notable debido a ciertos factores como el uso en conjunto de otros modelos para su correcto entrenamiento.

4.2.2.5. Resultado simulación de red malla con RF

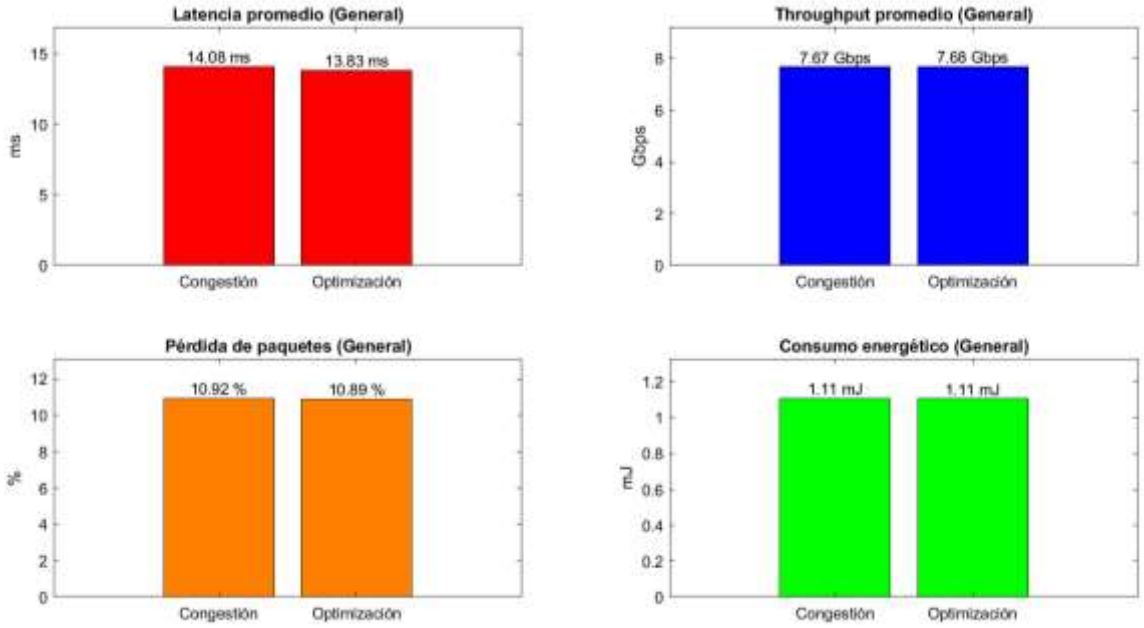
El modelo Random forest (RF) obtuvo una eficacia del 95.08% y un error del 4.92%, si bien tuvo un margen de error considerable por factores como el entorno variable en cantidad de dispositivos, se logró defender correctamente gracias a su aplicación de modelo clasificador con mejor balance entre velocidad de enrutamiento y rendimiento.

Ilustración 31. Estado final de topología malla con RF



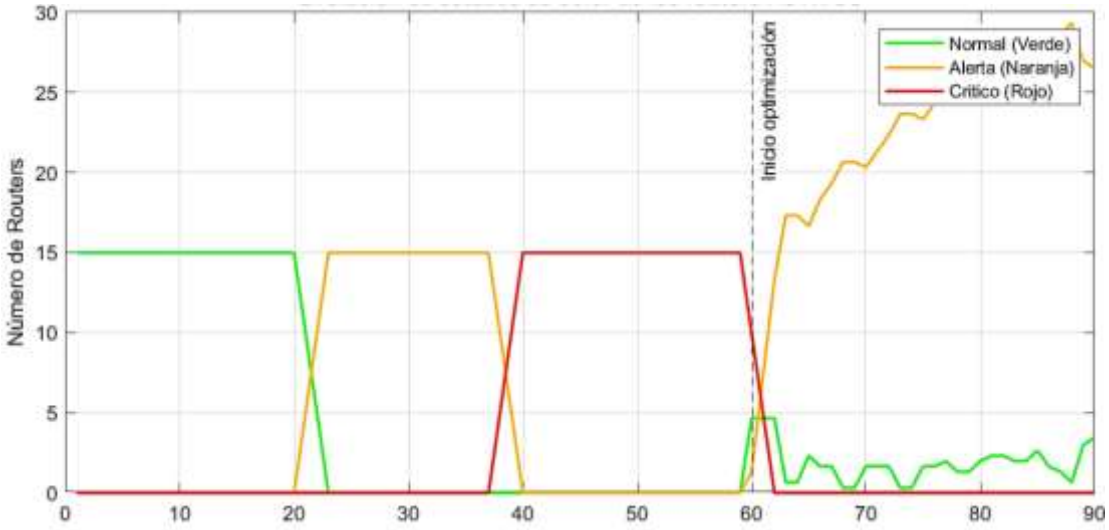
Autor: Jordy Lucas

Ilustración 32. Resultados de simulación RF



Autor: Jordy Lucas

Ilustración 33. Evolución de estados en los routers con RF



Autor: Jordy Lucas

Tabla 13. Métricas de eficiencia RF

Métrica de eficiencia	Valor
Reducción de latencia (%)	1.806
Reducción pérdida de paquetes (%)	0.136
Eficacia de validación (%)	0.308
Error de validación (%)	0.066

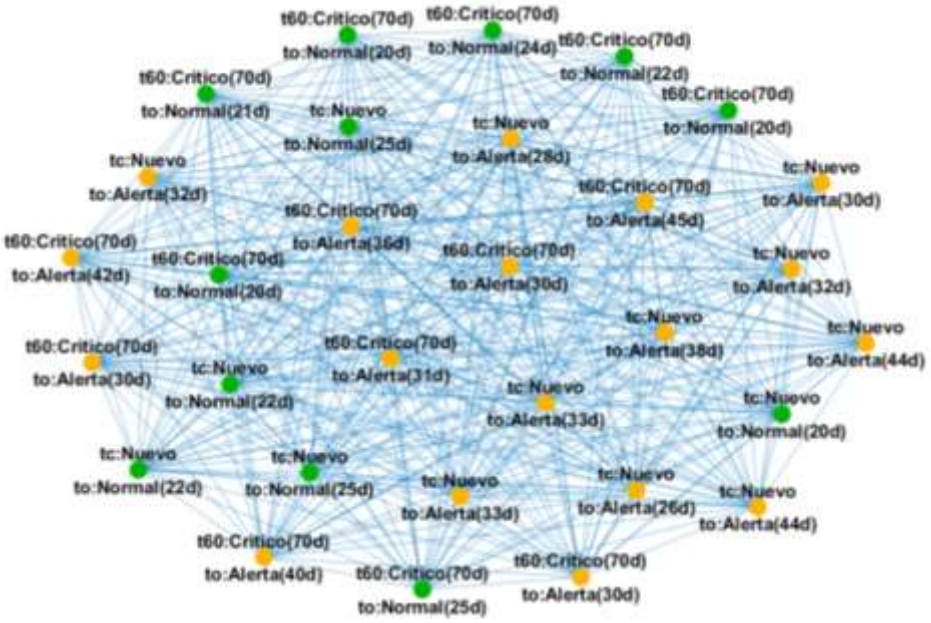
Autor: Jordy Lucas

RF mostró una reducción de latencia del 1.81% con un aumento marginal en el throughput, aunque el rendimiento es estable la mejora de latencia fue moderada e indica una correcta clasificación.

4.2.2.6. Resultado simulación de red malla con SVM

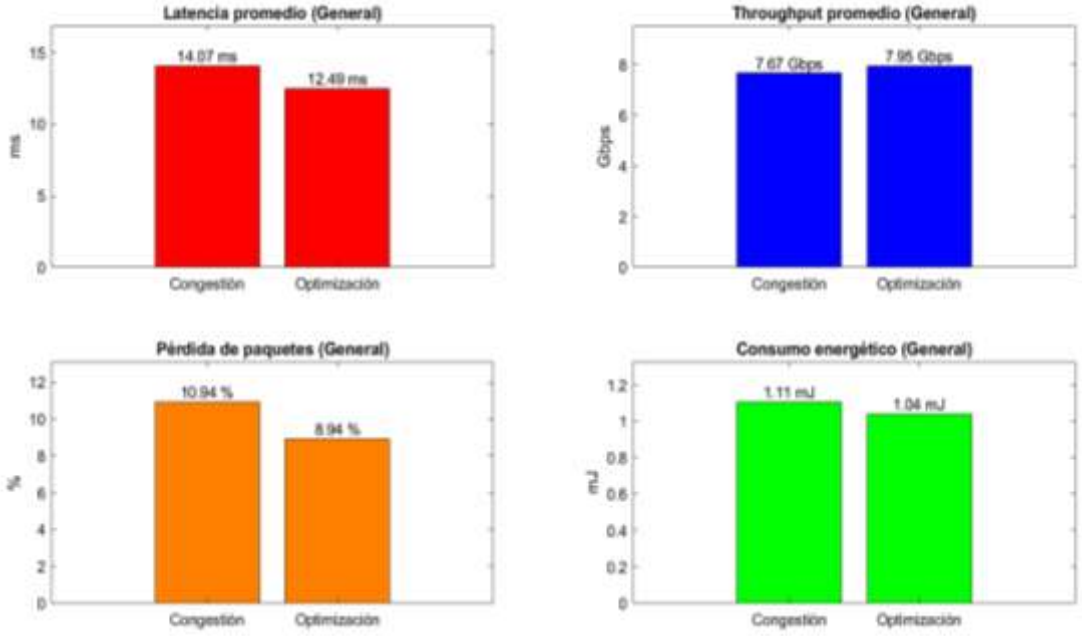
La máquina de vectores de soporte (SVM) fue el modelo de mayor precisión de validación con una eficacia del 99.67% y error mínimo con solo 0.33%, su alta fiabilidad de clasificación resultó en el mejor rendimiento general.

Ilustración 34. Estado final de topología malla con SVM



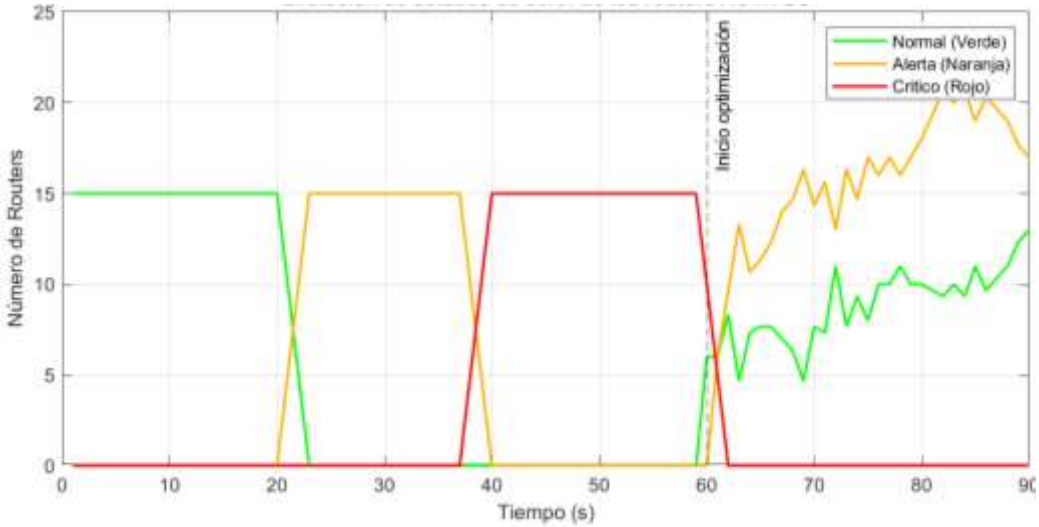
Autor: Jordy Lucas

Ilustración 35. Resultados de simulación SVM



Autor: Jordy Lucas

Ilustración 36. Evolución de estados en los routers con SVM



Autor: Jordy Lucas

Tabla 14. Métricas de eficiencia SVM

Métrica de eficiencia	Valor
Reducción de latencia (%)	1.806
Reducción pérdida de paquetes (%)	0.136
Eficacia de validación (%)	0.308
Error de validación (%)	0.066

Autor: Jordy Lucas

4.3. Análisis consolidado de reducción de latencia

4.3.1. Matriz de rendimiento en algoritmos

El objetivo final de la investigación se centra en el análisis del impacto del enrutamiento dinámico basado en aprendizaje automático para reducir latencia en redes 5G, la siguiente matriz demuestra los resultados de los seis modelos probados sobre la topología malla con los mismos parámetros que permitan comparar su precisión de clasificación con el impacto real de las métricas de rendimiento como latencia y perdida de paquetes.

Tabla 15. matriz de rendimiento en algoritmos

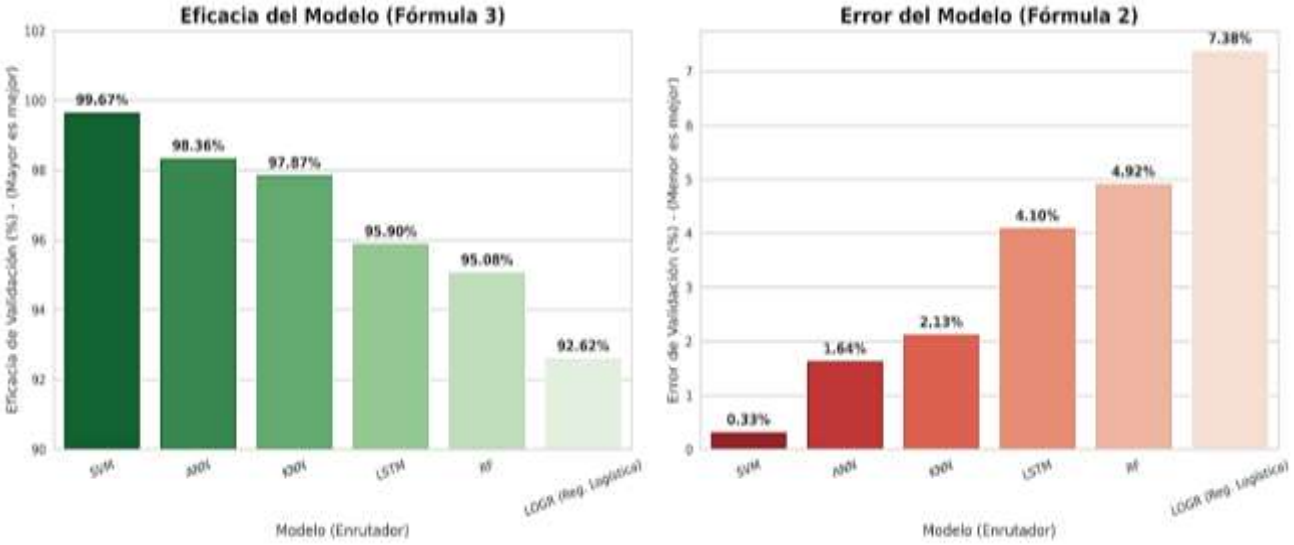
Modelo	Eficacia de validación	Error de predicción	Reducción de latencia	Pérdida de paquetes	Rendimiento energético
SVM	99.67%	0.33%	11.26%	18.26%	0.13095 pj/bit
ANN	98.36%	1.64%	9.57%	15.23%	0.13261 pj/bit
KNN	97.87%	2.13%	6.77%	10.27%	0.13625 pj/bit
RF	95.08%	4.92%	1.71%	0.31%	0.14402 pj/bit
LSTM	95.90%	4.10%	0.35%	-2.56%	0.14623 pj/bit
LOGR	92.62%	7.38%	0.07%	-2.79%	0.14595 pj/bit

Autor: Jordy Lucas

4.3.1.1. Análisis de precisión del enrutador

La efectividad de la reducción en latencia depende directamente de la capacidad del modelo para diagnosticar correctamente el estado de la red, los resultados de la validación demuestran una clara jerarquía de precisión donde la máquina de vectores de soporte (SVM) y red neuronal artificial (ANN) exhibieron la menor tasa de error.

Ilustración 37. Comparación de precisión del enrutador

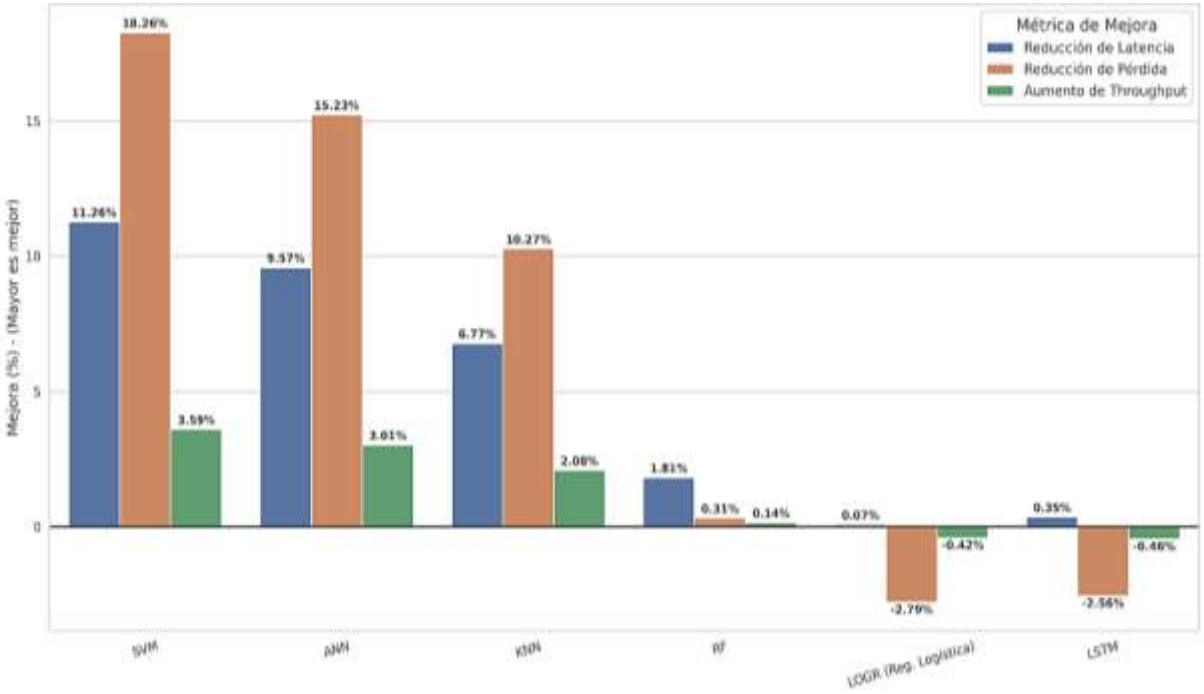


Autor: Jordy Lucas

4.3.1.2. Impacto en latencia, pérdida y throughput

Los resultados confirman que el impacto en el rendimiento está intrínsecamente ligado a la precisión del modelo:

Ilustración 38. Comparación de métricas en mejora del rendimiento



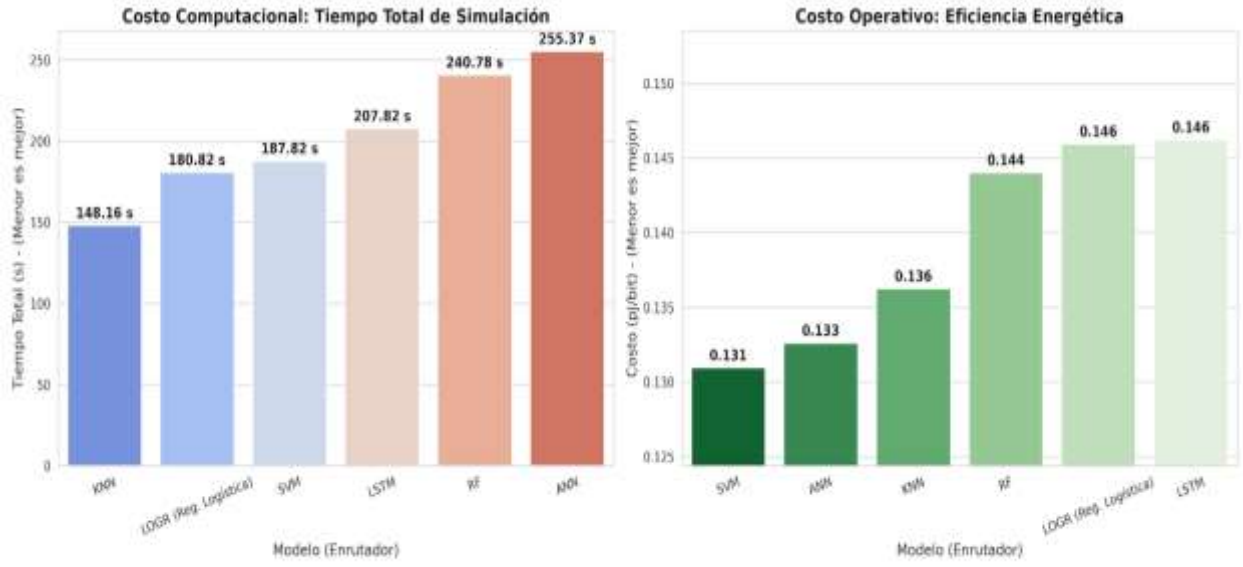
Autor: Jordy Lucas

La SVM se establece como el algoritmo de mayor impacto al lograr la mayor reducción de latencia con un 11.26% y mayor reducción de pérdida de paquetes con 18.26%, este rendimiento es un reflejo directo de la precisión que permite al enrutador dinámico reasignar el tráfico de manera óptima y aumentar el throughput. Los modelos LOGR y LSTM resultaron ineficaces al registrar valores de impacto negativo en la pérdida de paquetes, esto demuestra que son modelos no aptos para la viabilidad en redes 5G.

4.3.1.3. Costo operativo y algoritmo óptimo

La elección del modelo optimo debe considerar el costo operativo como eficiencia energética y costo computacional mediante el tiempo de ejecución para determinar una gran respuesta de adaptación al entorno.

Ilustración 39. Comparación de costos del enrutador



Autor: Jordy Lucas

SVM es el modelo más eficiente energéticamente con un costo operativo más bajo del 0.13 pJ/bit siendo un factor crucial para la sostenibilidad de la red, su tiempo de simulación es competitivo sumado a su superioridad en la reducción de latencia y pérdida de paquetes, esto demuestra a SVM como el algoritmo óptimo para la implementación del enrutamiento dinámico en redes 5G de topología malla.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones:

El estudio actual pone de manifiesto la trascendencia de emplear estrategias auto adaptativas con aprendizaje automático para optimizar el rendimiento y eficiencia en redes inalámbricas de quinta generación, mediante la integración de la revisión teórica con un riguroso proceso de simulación se logró resultados decisivos que respaldan la factibilidad de estas técnicas para reducir latencia de forma sustancial. A continuación, se expone las conclusiones derivadas de los objetivos específicos de la investigación:

Conclusión 1: Modelos según el estado del arte

La investigación fundamentada en el estado del arte priorizó un conjunto de seis algoritmos de aprendizaje automático (SVM, ANN, KNN, RF, LSTM LOGR). Esta selección se basó en el potencial teórico de los modelos para clasificación de estados y toma de decisiones dinámicas para el enrutamiento, este análisis preliminar no solo guio la elección de los modelos a simular, también justificó la adopción de la topología malla como la estructura base para los escenarios. Dicha topología se impuso sobre la estrella y árbol por su superioridad inherente en adaptación y menor latencia inicial con 14.07 ms que da por sentado las bases sólidas para el diseño experimental y la posterior fase de optimización.

Conclusión 2: Diseño de escenarios simulados

El diseño experimental implementado centrado en la topología mallada confirmó su eficacia como entorno de prueba ideal para el enrutamiento dinámico. La simulación comparativa de los seis algoritmos de aprendizaje automático (SVM, ANN, KNN, RF, LSTM y LOGR) reveló una diferencia considerable en sus capacidades de optimización. Mientras que los algoritmos de alta precisión (SVM y ANN) lograron mejoras significativas durante la fase de optimización, aquellos que obtuvieron clasificaciones bajas (LOGR) o que utilizaron enfoques inadecuados (LSTM) resultaron ineficaces. Este contraste demostró que su intervención era perjudicial o nula, con un empeoramiento de las medidas de pérdida de paquetes. Este resultado confirma la necesidad de seleccionar únicamente algoritmos con una alta fiabilidad predictiva.

Conclusión 3: Análisis consolidado de reducción de latencia

Un análisis exhaustivo del impacto en las técnicas de aprendizaje automático indica que las máquinas de vectores de soporte (SVM) representan la solución más eficaz y robusta para gestionar la latencia. Este algoritmo logró un rendimiento óptimo en tres métricas clave: precisión de clasificación (99,67 %), tasa de reducción de la latencia (11,26 %) y eficiencia energética (0,131 pJ/bit). Aunque las redes neuronales artificiales (ANN) demostraron un rendimiento muy competitivo (logrando una reducción del 9,57 % en la latencia), las SVM prevalecieron debido a su menor costo computacional y su mayor eficacia en la reducción de la pérdida de paquetes. Es decir, las SVM logran el equilibrio óptimo entre un rendimiento excepcional y la sostenibilidad operativa que se ajusta plenamente a los objetivos fundamentales de este estudio.

5.2. Recomendaciones

A partir de los resultados obtenidos en esta investigación se presenta las siguientes recomendaciones orientadas a fortalecer el desarrollo, validación y futura implementación de estrategias adaptativas a redes inalámbricas con un énfasis en el rendimiento del modelo respecto a sus escenarios con mayor exigencia:

- **Dar prioridad a la implementación de máquinas de vectores de soporte (SVM) para el enrutamiento dinámico:** Se recomienda poner en marcha las máquinas de vectores de soporte (SVM) en el marco de una prueba piloto para enrutadores dinámicos en entornos 5G controlados o reales. Esta validación es crucial para confirmar las ventajas de sus resultados de simulación: reducción del retraso en un 11,26%, reducción de la pérdida de paquetes en un 18,26 % y eficiencia energética óptima (0,131 pJ/bit), lo que garantiza el mantenimiento de su rendimiento de clasificación (precisión del 99,67 %) en entornos operativos.

- **Exploración de la optimización de modelos híbridos y aprendizaje profundo:** Se propone desarrollar modelos híbridos que combinen la alta precisión de las redes neuronales artificiales (ANN) con tecnologías capaces de reducir los costos computacionales y el consumo de energía. Este enfoque aprovechará el potencial del aprendizaje profundo y al mismo tiempo evitará los tiempos de simulación excesivos o los altos costos energéticos asociados a las ANN puras, ofreciendo una solución alternativa a las máquinas de vectores de soporte (SVM).

- **Implementar protocolos de monitoreo continuo y reentrenamiento:** el sistema implementado debe integrar mecanismos de monitoreo en tiempo real para registrar continuamente las métricas de la red. Los modelos de aprendizaje automático deben ser reentrenados y calibrados periódicamente para garantizar que sus capacidades predictivas se adapten a los cambios en los patrones de tráfico, la expansión de la red y la introducción de nuevas tecnologías, preservando así el rendimiento que permiten sus capacidades de adaptación.

- **Investigación sobre la escalabilidad de las topologías en malla:** dado que las topologías en malla han demostrado ser la infraestructura más robusta, se recomienda realizar investigaciones futuras para determinar sus límites para la expansión a gran escala. Se debe establecer el número máximo de enrutadores y dispositivos que esta topología puede gestionar antes de que la carga de la complejidad geométrica y los requisitos de enrutamiento dinámico superen los beneficios obtenidos gracias a la reducción de la latencia.

- **Desarrollo de un protocolo de red nativo para las decisiones en el ámbito del aprendizaje automático:** se recomienda desarrollar y probar un protocolo de enrutamiento especialmente optimizado para la transmisión de las decisiones generadas por los algoritmos de aprendizaje automático. Este enfoque reduce la latencia durante la conmutación y la ejecución de las decisiones, mejorando así la eficacia de la integración entre el ámbito del software (algoritmos de aprendizaje automático) y el ámbito del hardware (enrutadores físicos).

CAPÍTULO VI

REFERENCIAS BIBLIOGRÁFICAS

Bibliografía

[1] Itu-r, “Minimum requirements related to technical performance for IMT-2020 radio interface(s) M Series Mobile, radiodetermination, amateur and related satellite services,” 2017. [Online]. Available: http://www.itu.int/ITU-R/go/patents/en

[2] Andrews Jeffrey *et al.*, “What will 5G be?,” *IEEE Journal on Selected Areas in Communications*, vol. 32, pp. 1065–1082, 2014, doi: 10.1109/JSAC.2014.2328098.

[3] Zhang Chaoyun, Patras Paul, and Haddadi Hamed, “Deep Learning in Mobile and Wireless Networking: A Survey,” *IEEE Communications Surveys and Tutorials*, vol. 21, pp. 2224–2287, 2019, doi: 10.1109/COMST.2019.2904897.

[4] Simeone Osvaldo, “A brief introduction to machine learning for engineers,” 2018, *Now Publishers Inc*. doi: 10.1561/2000000102.

[5] Unión Internacional de Telecomunicaciones, “ITU Standardization on 5G,” Oct. 2020, Accessed: Sep. 08, 2025. [Online]. Available: http://www.itu.int/pub/R-REP-M/en

[6] George Hardesty, “WiFi: Una guía completa: estándares 802.11 y componentes clave,” WiFi: Una guía completa. Accessed: Sep. 11, 2025. [Online]. Available: https://www.data-alliance.net/es/wifi-una-gu%C3%ADa-completa-est%C3%A1ndares-80211-y-componentes-clave

[7] Nakivo, “Tipos de Topología de Red: Visión General Completa,” https://www.nakivo.com/es/blog/types-of-network-topology-explained/.

[8] Data Science, “Machine Learning: definición, funcionamiento, usos,” https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos.

[9] Irigaray Martín, “Inteligencia Artificial para Predicción (Parte 2),” https://www.ideati.net/blog/ia-predictivo-construccion/.

[10] Bagnato Juan, “Aplicaciones del Machine Learning,” https://www.aprendemachinelearning.com/aplicaciones-del-machine-learning/.

[11] Diego Calvo, “Aprendizaje supervisado,” Aprendizaje supervisado. Accessed: Sep. 12, 2025. [Online]. Available: https://www.diegocalvo.es/aprendizaje-supervisado/

[12] Goldberger Jacob, Roweis Sam, Hinton Geoff, and Salakhutdinov Ruslan, “Neighbourhood Components Analysis,” May 2005, Accessed: Sep. 11, 2025.

[Online]. Available: https://papers.nips.cc/paper_files/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf

[13] MathWorks Inc, “Predecir etiquetas de clases mediante el bloque de predicción ClassificationKNN - MATLAB y Simulink,” Predict Class Labels Using ClassificationKNN Predict Block. Accessed: Nov. 02, 2025. [Online]. Available: https://la.mathworks.com/help/stats/predict-class-labels-using-classification-knn-predict-block.html

[14] Andrés Juan and Gacharná Ariza, “Detección de ataques de reconocimiento en Redes IoT empleando modelos de Machine Learning,” Universidad de los Andres Colombia, 2023. Accessed: Nov. 02, 2025. [Online]. Available: https://hdl.handle.net/1992/63988

[15] Luis Torres, “Regresión logística ¿Cómo funciona? Qué secretos esconde?” Accessed: Nov. 02, 2025. [Online]. Available: https://www.themachinelearners.com/regresion-logistica-regularizacion/

[16] César Rodriguez, “Maquina de Soporte Vectorial (SVM) | by César Chique Rodriguez | Medium,” *Medium*, Sep. 2020, Accessed: Sep. 12, 2025. [Online]. Available: https://medium.com/@csarchiquerodriguez/maquina-de-soporte-vectorial-svm-92e9f1b1b1ac

[17] Scikit learn, “Árboles de decisión .” Accessed: Sep. 12, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html

[18] Jesús Bobadilla Sancho, *Machine Learning y Deep Learning: Usando Python, Scikit y Keras - Jesús Bobadilla - Google Libros*. Bogotá, 2021. Accessed: Nov. 03, 2025. [Online]. Available: https://books.google.com.ec/books?hl=es&lr=&id=iAAyEAAAQBAJ&oi=fnd&pg=PA11&dq=Deep+Learning+concepto&ots=QiB4t0nKaw&sig=H4I8PYS3UaxKxj552tVyoP02wpE&redir_esc=y#v=onepage&q=Deep%20Learning%20concepto&f=false

[19] Raul Lopez Briega, “Introducción al Deep Learning,” Jun. 2017. Accessed: Nov. 03, 2025. [Online]. Available: https://relopezbriega.github.io/blog/2017/06/13/introduccion-al-deep-learning/

[20] Turing, “¿Cómo funcionan los modelos de redes neuronales en el aprendizaje automático?” Accessed: Sep. 12, 2025. [Online]. Available: https://www.turing.com/kb/how-neural-network-models-in-machine-learning-work

[21] Sepp Hochreiter and Jürgen Schmidhuber, *Long Short-Term Memory | Neural Computation | MIT Press*. Cambridge, Massachusetts: Massachusetts Institute of Technology, 1997. Accessed: Nov. 04, 2025. [Online]. Available: https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory?redirectedFrom=fulltext

[22] CMAX, “Redes Neuronales LSTM,” Dec. 2023. Accessed: Nov. 04, 2025. [Online]. Available: https://ciberseguridadmax.com/lstm/

[23] Speedman Telecom, “Comprendiendo la Diferencia entre Latencia, Ancho de Banda y Throughput en Redes - Speedmax.” Accessed: Sep. 12, 2025. [Online]. Available: https://speedmax.pe/comprendiendo-la-diferencia-entre-latencia-ancho-de-banda-y-throughput-en-redes/

[24] Alejandro Gillis, “¿Qué es la pérdida de paquetes? Causas, detección y correcciones.” Accessed: Sep. 12, 2025. [Online]. Available: https://www.techtarget.com/searchnetworking/definition/packet-loss

[25] Movistar, “(2) Qué es el jitter y como impacta en la experiencia de usuario.” Accessed: Sep. 12, 2025. [Online]. Available: https://ww2.movistar.cl/blog/post/que-es-jitter/

[26] Nvidia, “¿Qué es la eficiencia energética y por qué es importante? | Glosario de NVIDIA.” Accessed: Sep. 12, 2025. [Online]. Available: https://www.nvidia.com/en-us/glossary/power-efficiency/

[27] Goodwin Michael, “¿Qué es la latencia? | IBM,” https://www.ibm.com/mx-es/topics/latency.

[28] Rodríguez Luis, “Análisis del Rendimiento de Redes 5G utilizando Machine Learning,” https://dialnet.unirioja.es/servlet/articulo?codigo=9714319.

[29] Solano David, “Algoritmos de aprendizaje automático para optimizar las redes 5G: Desarrollo y evaluación del rendimiento,” https://revistas.udistrital.edu.co/index.php/vinculos/article/view/16061.

[30] Shukla Neha, Siloiya Ayush, Singh Apoorva, and Saini Aayushi., “Xcelerate5G: Optimizing Resource Allocation Strategies for 5G Network Using ML,” in *Proceedings - International Conference on Computing, Power, and Communication Technologies, IC2PCT 2024*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 417–423. doi: 10.1109/IC2PCT60090.2024.10486750.

[31] Thales Bf, “Presentando la tecnología y redes 5G (definición, características, 5G vs 4G y casos de uso),” https://www.thalesgroup.com/es/countries/americas/latin-america/dis/movil/inspiracion/5g.

[32] Rodríguez Luis, “Análisis del Rendimiento de Redes 5G utilizando Machine Learning,” https://dialnet.unirioja.es/servlet/articulo?codigo=9714319.

[33] LOT, “LEY ORGANICA DE TELECOMUNICACIONES,” Jun. 2015. [Online]. Available: www.lexis.com.ec

[34] ARCOTEL, “AGENCIA DE REGULACIÓN Y CONTROL DE LAS TELECOMUNICACIONES ARCOTEL INFORME DE GESTIÓN Periodo: Enero a Diciembre 2023 Contenido,” 2015. Accessed: Sep. 09, 2025. [Online]. Available: https://www.arcotel.gob.ec/wp-content/uploads/2024/01/informe_de_gestio%CC%81n_an%CC%83o_2023_con-formato-final-signed.pdf

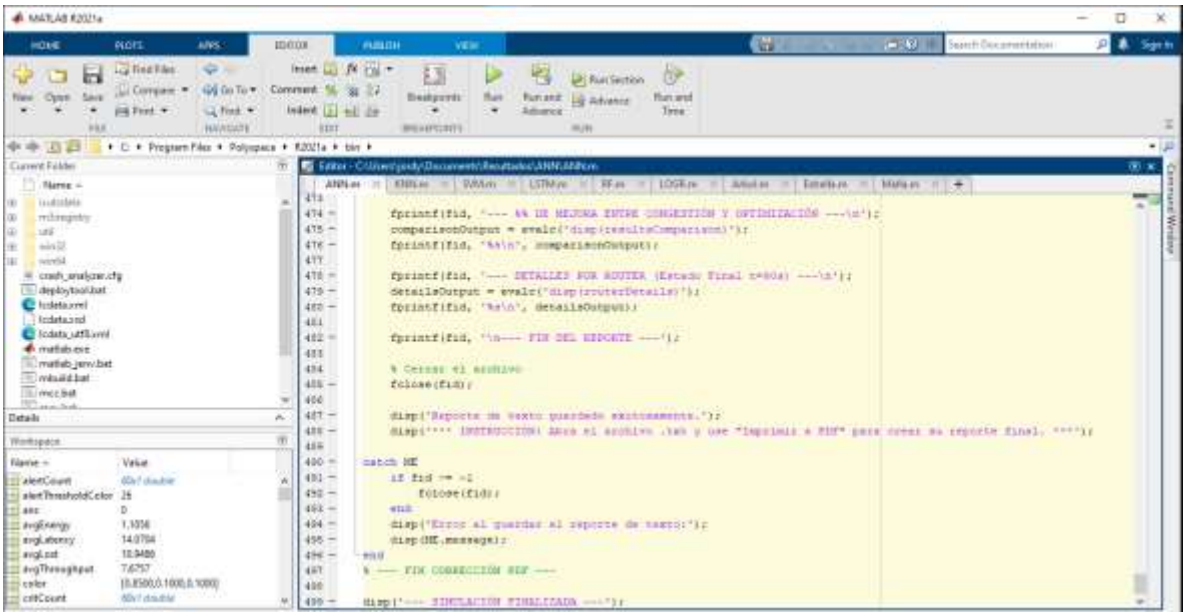
CAPÍTULO VII

REFERENCIAS BIBLIOGRÁFICAS

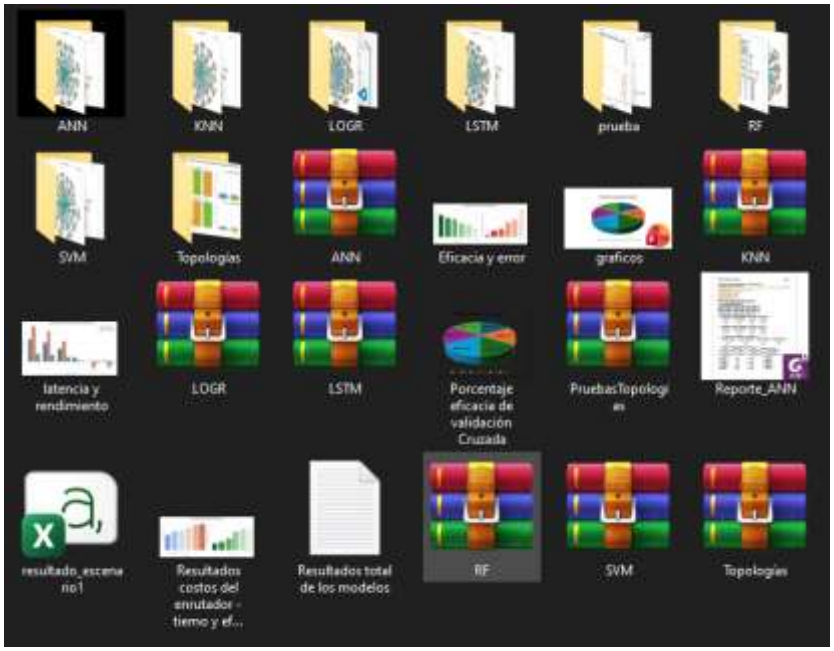
Anexo 1. Enlace de repositorio con archivos

https://github.com/JordyLucas7/Tesis5G_ML

Anexo 2. Interfaz de Matlab con archivos de simulación



Anexo 3. Clasificación de resultados



Anexo 4. Código de topología Malla

```
%% =========================================================================
% SIMULACIÓN DE RED 5G - 15 ROUTERS MALLA
% OBJETIVO: Simular la FASE DE CONGESTIÓN (t=1-60) con topología de
Malla.
% 1. [LÓGICA DE REDUNDANCIA] Las métricas de latencia y pérdida NO se
%    acumulan. Cada router maneja su propia carga, simulando resiliencia
de la topología de malla
% 2. [SIN rng(1)] Se elimina 'rng(1)' para permitir variabilidad natural.
% 3. [Fórmulas] Se usan las fórmulas de métricas unificadas (v5).
% 4. [Reporte] Se usa el guardado de gráficos y reporte .txt estándar.
% 5. [ARREGLO DE ERROR] Corregido el typo 'lostPackB' a 'lostPackets'.
%
%% =========================================================================

clear; clc; close all;
% --- 'rng(1)' SE ELIMINA INTENCIONALMENTE para que esta simulación
% --- sea diferente a la de Estrella y Árbol.

%% ===== PARAMETROS =====
numRouters = 15;
maxDevicesPerRouter = 70; % Estandarizado a 70
simTime = 60; % Solo Fase de Congestión

% Carpeta destino para guardar resultados
outputDir = 'C:\Users\jordy\Documents\Resultados\Topologías\TMALLA\';
if ~exist(outputDir, 'dir')
    mkdir(outputDir);
end

% Crear routers
routers = strcat("Router_", string(1:numRouters));

% Crear grafo base solo con routers
G = graph();
G = addnode(G, routers);

% ================== CONECTAR TOPOLOGÍA MALLA (Completa)
for i = 1:numRouters
    for j = i+1:numRouters
        G = addedge(G, routers(i), routers(j));
    end
end

%% ===== CLASIFICADOR DE COLOR (Árbol de Decisión Simple) ====
% (Estandarizado con los scripts de ML)
trainDataColor = []; labelsColor = [];
critThresholdColor = 46; % >= 46 Crítico (Rojo)
alertThresholdColor = 26; % 26-45 Alerta (Naranja)
% <= 25 Normal (Verde)
for d = 1:maxDevicesPerRouter
    if d >= critThresholdColor; stateColor = "Critico";
    elseif d >= alertThresholdColor; stateColor = "Alerta";
    else; stateColor = "Normal"; end
    trainDataColor = [trainDataColor; d];
    labelsColor = [labelsColor; stateColor];
```

```
end
labelsColor = categorical(labelsColor);
Mdl_Color = fitctree(trainDataColor, labelsColor);

%% ===== INICIALIZAR VARIABLES =====
latency = NaN(simTime, numRouters);
throughput = NaN(simTime, numRouters);
lostPackets = NaN(simTime, numRouters);
energyConsumption = NaN(simTime, numRouters);
packetsSent = zeros(simTime, numRouters);
packetsReceived = zeros(simTime, numRouters);
devicesConnected = zeros(simTime, numRouters);
routerStatesColor = strings(simTime, numRouters);
finalLaptops = zeros(numRouters,1);
finalPhones = zeros(numRouters,1);
laptopColor = [0 0.6 0]; phoneColor  = [1 0 0];

%% ================== SIMULACIÓN ==================
fig1 = figure('Name','Simulación Red 5G - Malla (Lógica
Corregida)','Position',[100 100 1200 700]);
tic;

for t = 1:simTime
    Gtemp = G;

    % --- Bucle 1: Calcular métricas base para cada router ---
    for r = 1:numRouters
        % --- Carga Estandarizada ---
        numTotal = round((t/simTime) * maxDevicesPerRouter);
        if numTotal == 0; numTotal = 1; end % Evitar 0
        devicesConnected(t,r) = numTotal;
        numLaptops = randi([round(0.4*numTotal), round(0.6*numTotal)]); %
Split 40/60
        numPhones  = numTotal - numLaptops;

        if numLaptops > 0
            laptops = strcat("L_", string(r), "_", string(1:numLaptops));
            Gtemp = addnode(Gtemp, laptops);
            for L = 1:numLaptops; Gtemp = addedge(Gtemp, routers(r),
laptops(L)); end
        end
        if numPhones > 0
            phones  = strcat("P_", string(r), "_", string(1:numPhones));
            Gtemp = addnode(Gtemp, phones);
            for P = 1:numPhones; Gtemp = addedge(Gtemp, routers(r),
phones(P)); end
        end

        if t == simTime; finalLaptops(r) = numLaptops; finalPhones(r) =
numPhones; end

        % --- Tráfico Estandarizado ---
        if numTotal > 0
            trafficLaptops = poissrnd(60, [1 numLaptops]);
            trafficPhones = poissrnd(90, [1 numPhones]) +
randn(1,numPhones)*15;
            trafficPhones(trafficPhones < 0) = 0;
```

```
            currentTraffic = sum(trafficLaptops) + sum(trafficPhones);
            if isempty(currentTraffic); currentTraffic = 0; end
            packetsSent(t,r) = currentTraffic / 10;
        else
            packetsSent(t,r) = 0;
        end

        % --- Fórmulas de Métricas Estandarizadas (Base) ---
        if numTotal > 0
            packetsReceived(t,r) = packetsSent(t,r) * (0.8 -
0.3*(numTotal/maxDevicesPerRouter));
            latency(t,r) = max(5, 7 + (numTotal^1.2)/12 + rand*1.5);
            lostPackets(t,r) = max(0, min(25, (numTotal/3.5) +
rand*1.5));
            throughput(t,r) = max(1, 9 - (numTotal^1.0)/25 + rand*0.2);
            energyConsumption(t,r) = max(0.4, 0.7 + 0.01*numTotal +
rand*0.1);
        else
            packetsReceived(t,r) = 0;
            latency(t,r) = NaN; lostPackets(t,r) = NaN; throughput(t,r) =
NaN; energyConsumption(t,r) = NaN;
        end
    end % Fin bucle for r

    % --- INICIO DE LÓGICA DE TOPOLOGÍA (MALLA) ---
    % En la Malla, la latencia y la pérdida NO se acumulan.
    % La redundancia  previene los cuellos de botella y cascadas.
    % Por lo tanto, NO se añade ningún bucle de acumulación.
    % Las métricas base calculadas arriba son las métricas finales.
    % --- FIN DE LÓGICA DE TOPOLOGÍA ---

    % --- Clasificación de Color ---
    for r = 1:numRouters
        predColor = predict(Mdl_Color, devicesConnected(t,r));
        routerStatesColor(t,r) = string(predColor);
    end

    % ===== VISUALIZACIÓN =====
    subplot(2,2,[1 3]); cla;
    h = plot(Gtemp,'Layout','force','NodeLabel',{}); % Layout de Malla

    nodeIndicesPC = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name,"L_")));
    if ~isempty(nodeIndicesPC); highlight(h, nodeIndicesPC, 'NodeColor',
laptopColor, 'MarkerSize', 3); end
    nodeIndicesMovil = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name,"P_")));
    if ~isempty(nodeIndicesMovil); highlight(h, nodeIndicesMovil,
'NodeColor', phoneColor, 'MarkerSize', 3); end

    for r = 1:numRouters
        state = routerStatesColor(t,r); nodeIdx = findnode(Gtemp,
routers(r));
        if nodeIdx > 0; if state == "Normal"; color = [0 0.7 0]; elseif
state == "Alerta"; color = [1 0.7 0]; else; color = [0.85 0.1 0.1]; end
            highlight(h, nodeIdx, 'NodeColor', color, 'MarkerSize', 7);
end
```

```
    end
    title(sprintf('Topología en Malla - Segundo %d',t));

    subplot(2,2,2); cla;
plot(1:t,mean(latency(1:t,:),2,'omitnan'),'b','LineWidth',1.5);
title('Latencia promedio de la red'); xlabel('Tiempo (s)'); ylabel('ms');
grid on; xlim([0 simTime]);
    subplot(2,2,4); cla;
plot(1:t,mean(lostPackets(1:t,:),2,'omitnan'),'r','LineWidth',1.5);
title('Pérdida de paquetes promedio (%)'); xlabel('Tiempo (s)');
ylabel('%'); grid on; xlim([0 simTime]);

    drawnow; pause(0.05);
end
elapsedTime = toc*1000;
disp(['Tiempo total de simulación: ', num2str(elapsedTime/1000, '%.2f'),
' segundos']);

%% ================= RESULTADOS (Estandarizados) =================
avgLatency = mean(mean(latency,'omitnan'));
avgThroughput = mean(mean(throughput,'omitnan'));
avgLost = mean(mean(lostPackets,'omitnan'));
avgEnergy = mean(mean(energyConsumption,'omitnan'));

resultsGeneral = table(avgLatency, avgThroughput, avgLost, avgEnergy, ...
    'VariableNames', {'Latencia_Promedio_ms', 'Throughput_Promedio_Gbps',
'Perdida_Promedio_p', 'Energia_Promedio_mJ'});

routerNames = routers';
% --- INICIO ARREGLO DE ERROR TYPO ---
finalRouterState = table(routerNames, finalLaptops, finalPhones, ...
    devicesConnected(end,:)', latency(end,:)', throughput(end,:)',
lostPackets(end,:)', energyConsumption(end,:)', ...
    'VariableNames',
{'Router','Laptops','Telefonos','Dispositivos_Finales','Latencia_ms_Final
','Throughput_Gbps_Final','Perdida_p_Final','Energia_mJ_Final'});
% --- FIN ARREGLO DE ERROR TYPO ---

disp('--- RESULTADOS GENERALES DE LA TOPOLOGÍA (Promedio t=1-60) ---');
disp(resultsGeneral);
disp('--- ESTADO FINAL DE ROUTERS (t=60s) ---');
disp(finalRouterState);

%% ===== GUARDAR REPORTE PDF (en .txt) =====
reportPath = fullfile(outputDir, 'Reporte_Malla.txt');
try
    fid = fopen(reportPath, 'w', 'n', 'UTF-8');
    if fid == -1; error('No se pudo crear el archivo de reporte en %s',
reportPath); end
    disp(['Creando reporte de texto en: ', reportPath]);

    fprintf(fid,
'============================================================\n');
    fprintf(fid, ' REPORTE DE SIMULACIÓN - TOPOLOGÍA MALLA
(CORREGIDO)\n');
    fprintf(fid,
'============================================================\n\n');
```

```
    fprintf(fid, 'Fecha de simulación: %s\n\n', datestr(now));
    fprintf(fid, '--- RESULTADOS GENERALES DE LA TOPOLOGÍA (Promedio t=1-
60) ---\n');
    reportOutput = evalc('disp(resultsGeneral)');
    fprintf(fid, '%s\n', reportOutput);
    fprintf(fid, '--- ESTADO FINAL DE ROUTERS (t=60s) ---\n');
    reportOutput = evalc('disp(finalRouterState)');
    fprintf(fid, '%s\n', reportOutput);
    fprintf(fid, '\n--- FIN DEL REPORTE ---');
    fclose(fid);
    disp('Reporte de texto guardado exitosamente.');
catch ME
    if fid ~= -1; fclose(fid); end
    disp('Error al guardar el reporte de texto:');
    disp(ME.message);
end

%% ================= GRÁFICOS (Estandarizados) =================
try; close(fig1); catch; end
fig2 = figure('Name', 'Resultados Promedio - Malla', 'Position', [100 100
1200 600], 'Visible', 'off');
subplot(2,2,1); bar(avgLatency,'r'); title('Latencia Promedio (ms)');
ylabel('ms');
subplot(2,2,2); bar(avgThroughput,'b'); title('Throughput Promedio
(Gbps)'); ylabel('Gbps');
subplot(2,2,3); bar(avgLost,'FaceColor',[1 0.5 0]); title('Pérdida
Paquetes Promedio (%)'); ylabel('%');
subplot(2,2,4); bar(avgEnergy,'g'); title('Consumo Energético Promedio
(mJ)'); ylabel('mJ');
drawnow; saveas(fig2, fullfile(outputDir,
'Malla_Resultados_Promedio.png')); close(fig2);

fig3 = figure('Name', 'Topologia Final - Malla', 'Position', [100 100 900
600], 'Visible', 'off');
hFinal = plot(G,'Layout','force','NodeLabel',{});
title('Topologia Final - Estado en t=60s');
for r = 1:numRouters
    routerName = routers(r); finalState = routerStatesColor(end, r);
    if finalState == "Normal"; color = [0 0.7 0]; elseif finalState ==
"Alerta"; color = [1 0.7 0]; else; color = [0.85 0.1 0.1]; end
    nodeIdxInG = findnode(G, routerName);
    if nodeIdxInG > 0
        highlight(hFinal, nodeIdxInG, 'NodeColor', color, 'MarkerSize',
8);
        textFinal = sprintf('%s\n%s (%dd)', routerName, finalState,
devicesConnected(end, r));
        text(hFinal.XData(nodeIdxInG), hFinal.YData(nodeIdxInG)-0.05,
textFinal, 'HorizontalAlignment', 'center', 'VerticalAlignment', 'top',
'FontSize', 9, 'Color', 'k', 'FontWeight', 'bold');
    end
end
drawnow; saveas(fig3, fullfile(outputDir, 'Malla_TopologiaFinal.png'));
close(fig3);

fig4 = figure('Name', 'Evolución de Estados (Color) - Malla', 'Position',
[100 100 900 400], 'Visible', 'off');
```

```
normCount = sum(routerStatesColor=="Normal", 2); alertCount =
sum(routerStatesColor=="Alerta", 2); critCount =
sum(routerStatesColor=="Critico", 2);
plot(1:simTime, movmean(normCount,3), 'g', 'LineWidth', 1.5); hold on;
plot(1:simTime, movmean(alertCount,3), 'Color', [1 0.7 0], 'LineWidth',
1.5);
plot(1:simTime, movmean(critCount,3), 'r', 'LineWidth', 1.5);
xlabel('Tiempo (s)'); ylabel('Número de Routers'); grid on;
legend('Normal (Verde)', 'Alerta (Naranja)', 'Critico (Rojo)');
title('Evolución de estados de color de los routers');
drawnow; saveas(fig4, fullfile(outputDir,
'Malla_EvolucionEstadosColor.png')); close(fig4);

fig6 = figure('Name', 'Evolución en Tiempo Real - Malla', 'Position',
[100 100 1200 500], 'Visible', 'off');
mean_latency = mean(latency, 2,'omitnan'); mean_throughput =
mean(throughput, 2,'omitnan');
subplot(1,2,1); plot(1:simTime, mean_latency, 'r', 'LineWidth', 1.5);
title('Evolución Latencia Promedio (Activos)'); xlabel('Tiempo (s)');
ylabel('ms'); grid on; xlim([0 simTime]); ylim('auto');
subplot(1,2,2); plot(1:simTime, mean_throughput, 'b', 'LineWidth', 1.5);
title('Evolución Throughput Promedio (Activos)'); xlabel('Tiempo (s)');
ylabel('Gbps'); grid on; xlim([0 simTime]); ylim('auto');
drawnow; saveas(fig6, fullfile(outputDir,
'Malla_EvolucionTiempoReal.png')); close(fig6);

disp('--- SIMULACIÓN DE TOPOLOGÍA MALLA CORREGIDA Y FINALIZADA ---');
```

Anexo 5. Código topología Estrella

```
%========================================================================
% SIMULACIÓN DE RED 5G - 15 ROUTERS ESTRELLA
% OBJETIVO: Simular la FASE DE CONGESTIÓN (t=1-60) con topología de
Estrella.
% CAMBIOS REALIZADOS (v7 - Lógica de Topología Real):
% 1. [LÓGICA DE CUELLO DE BOTELLA] Se añade lógica donde la latencia y
%    pérdida del nodo central (Router_1) se SUMA a la de todos los
%    demás nodos, simulando un punto de fallo central.
% 2. [SIN rng(1)] Se elimina 'rng(1)' para permitir variabilidad natural.
% 3. [Fórmulas] Se usan las fórmulas de métricas unificadas (v5).
% 4. [Reporte] Se usa el guardado de gráficos y reporte .txt estándar.
%
%========================================================================

clear; clc; close all;
% --- 'rng(1)' SE ELIMINA INTENCIONALMENTE para que esta simulación
% --- sea diferente a la de Malla y Árbol.

%% ===== PARAMETROS =====
numRouters = 15;
maxDevicesPerRouter = 70; % Estandarizado a 70
simTime = 60; % Solo Fase de Congestión

% Carpeta destino para guardar resultados
outputDir = 'C:\Users\jordy\Documents\Resultados\Topologías\TESTRELLA\';
```

```
if ~exist(outputDir, 'dir')
    mkdir(outputDir);
end

% Crear routers
routers = strcat("Router_", string(1:numRouters));

% Crear grafo base solo con routers
G = graph();
G = addnode(G, routers);

% ================== CONECTAR TOPOLOGÍA ESTRELLA ==================
% Router_1 es el nodo central
nodoCentral = routers(1);
for j = 2:numRouters
    G = addedge(G, nodoCentral, routers(j));
end

%% ===== CLASIFICADOR DE COLOR (Árbol de Decisión Simple) ====
% (Estandarizado con los scripts de ML)
trainDataColor = []; labelsColor = [];
critThresholdColor = 46; % >= 46 Crítico (Rojo)
alertThresholdColor = 26; % 26-45 Alerta (Naranja)
% <= 25 Normal (Verde)
for d = 1:maxDevicesPerRouter
    if d >= critThresholdColor; stateColor = "Critico";
    elseif d >= alertThresholdColor; stateColor = "Alerta";
    else; stateColor = "Normal"; end
    trainDataColor = [trainDataColor; d];
    labelsColor = [labelsColor; stateColor];
end
labelsColor = categorical(labelsColor);
Mdl_Color = fitctree(trainDataColor, labelsColor);


%% ===== INICIALIZAR VARIABLES =====
latency = NaN(simTime, numRouters);
throughput = NaN(simTime, numRouters);
lostPackets = NaN(simTime, numRouters);
energyConsumption = NaN(simTime, numRouters);
packetsSent = zeros(simTime, numRouters);
packetsReceived = zeros(simTime, numRouters);
devicesConnected = zeros(simTime, numRouters);
routerStatesColor = strings(simTime, numRouters);
finalLaptops = zeros(numRouters,1);
finalPhones = zeros(numRouters,1);
laptopColor = [0 0.6 0]; phoneColor  = [1 0 0];

%% ================== SIMULACIÓN ==================
fig1 = figure('Name','Simulación Red 5G - Estrella (Lógica
Corregida)','Position',[100 100 1200 700]);
tic;

for t = 1:simTime
    Gtemp = G;
```

```
    % --- Bucle 1: Calcular métricas base para cada router ---
    for r = 1:numRouters
        % --- Carga Estandarizada ---
        numTotal = round((t/simTime) * maxDevicesPerRouter);
        if numTotal == 0; numTotal = 1; end % Evitar 0
        devicesConnected(t,r) = numTotal;
        numLaptops = randi([round(0.4*numTotal), round(0.6*numTotal)]); %
Split 40/60
        numPhones  = numTotal - numLaptops;

        if numLaptops > 0
            laptops = strcat("L_", string(r), "_", string(1:numLaptops));
            Gtemp = addnode(Gtemp, laptops);
            for L = 1:numLaptops; Gtemp = addedge(Gtemp, routers(r),
laptops(L)); end
        end
        if numPhones > 0
            phones  = strcat("P_", string(r), "_", string(1:numPhones));
            Gtemp = addnode(Gtemp, phones);
            for P = 1:numPhones; Gtemp = addedge(Gtemp, routers(r),
phones(P)); end
        end

        if t == simTime; finalLaptops(r) = numLaptops; finalPhones(r) =
numPhones; end

        % --- Tráfico Estandarizado ---
        if numTotal > 0
            trafficLaptops = poissrnd(60, [1 numLaptops]);
            trafficPhones = poissrnd(90, [1 numPhones]) +
randn(1,numPhones)*15;
            trafficPhones(trafficPhones < 0) = 0;
            currentTraffic = sum(trafficLaptops) + sum(trafficPhones);
            if isempty(currentTraffic); currentTraffic = 0; end
            packetsSent(t,r) = currentTraffic / 10;
        else
            packetsSent(t,r) = 0;
        end

        % --- Fórmulas de Métricas Estandarizadas (Base) ---
        if numTotal > 0
            packetsReceived(t,r) = packetsSent(t,r) * (0.8 -
0.3*(numTotal/maxDevicesPerRouter));
            latency(t,r) = max(5, 7 + (numTotal^1.2)/12 + rand*1.5);
            lostPackets(t,r) = max(0, min(25, (numTotal/3.5) +
rand*1.5));
            throughput(t,r) = max(1, 9 - (numTotal^1.0)/25 + rand*0.2);
            energyConsumption(t,r) = max(0.4, 0.7 + 0.01*numTotal +
rand*0.1);
        else
            packetsReceived(t,r) = 0;
            latency(t,r) = NaN; lostPackets(t,r) = NaN; throughput(t,r) =
NaN; energyConsumption(t,r) = NaN;
        end
    end % Fin bucle for r
```

```
    % --- INICIO DE LÓGICA DE TOPOLOGÍA (CUELLO DE BOTELLA) ---
    % La latencia y pérdida del nodo central (Router 1) impacta a todos.
    latencia_central = latency(t,1);
    perdida_central = lostPackets(t,1);

    for r = 2:numRouters % Iterar solo sobre los routers "hoja"
        latency(t,r) = latency(t,r) + latencia_central;
        lostPackets(t,r) = lostPackets(t,r) + perdida_central;
    end
    % Asegurarse de que la pérdida no supere un máximo (ej. 100%)
    lostPackets(t,:) = min(lostPackets(t,:), 100);
    % --- FIN DE LÓGICA DE TOPOLOGÍA ---

    % --- Clasificación de Color (se basa en la carga, no en la latencia
acumulada) ---
    for r = 1:numRouters
        predColor = predict(Mdl_Color, devicesConnected(t,r));
        routerStatesColor(t,r) = string(predColor);
    end

    % ===== VISUALIZACIÓN =====
    subplot(2,2,[1 3]); cla;
    h = plot(Gtemp,'Layout','force','NodeLabel',{}); % Layout de Estrella

    nodeIndicesPC = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name,"L_")));
    if ~isempty(nodeIndicesPC); highlight(h, nodeIndicesPC, 'NodeColor',
laptopColor, 'MarkerSize', 3); end
    nodeIndicesMovil = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name,"P_")));
    if ~isempty(nodeIndicesMovil); highlight(h, nodeIndicesMovil,
'NodeColor', phoneColor, 'MarkerSize', 3); end

    for r = 1:numRouters
        state = routerStatesColor(t,r); nodeIdx = findnode(Gtemp,
routers(r));
        if nodeIdx > 0; if state == "Normal"; color = [0 0.7 0]; elseif
state == "Alerta"; color = [1 0.7 0]; else; color = [0.85 0.1 0.1]; end
            highlight(h, nodeIdx, 'NodeColor', color, 'MarkerSize', 7);
end
    end
    title(sprintf('Topología en Estrella - Segundo %d',t));

    subplot(2,2,2); cla;
plot(1:t,mean(latency(1:t,:),2,'omitnan'),'b','LineWidth',1.5);
title('Latencia promedio de la red (Acumulativa)'); xlabel('Tiempo (s)');
ylabel('ms'); grid on; xlim([0 simTime]);
    subplot(2,2,4); cla;
plot(1:t,mean(lostPackets(1:t,:),2,'omitnan'),'r','LineWidth',1.5);
title('Pérdida de paquetes promedio (%) (Acumulativa)'); xlabel('Tiempo
(s)'); ylabel('%'); grid on; xlim([0 simTime]);

    drawnow; pause(0.05);
end
```

```
elapsedTime = toc*1000;
disp(['Tiempo total de simulación: ', num2str(elapsedTime/1000, '%.2f'),
' segundos']);

%% ================= RESULTADOS (Estandarizados) =================
avgLatency = mean(mean(latency,'omitnan'));
avgThroughput = mean(mean(throughput,'omitnan'));
avgLost = mean(mean(lostPackets,'omitnan'));
avgEnergy = mean(mean(energyConsumption,'omitnan'));

resultsGeneral = table(avgLatency, avgThroughput, avgLost, avgEnergy, ...
    'VariableNames', {'Latencia_Promedio_ms', 'Throughput_Promedio_Gbps',
'Perdida_Promedio_p', 'Energia_Promedio_mJ'});

routerNames = routers';
finalRouterState = table(routerNames, finalLaptops, finalPhones, ...
    devicesConnected(end,:)', latency(end,:)', throughput(end,:)',
lostPackets(end,:)', energyConsumption(end,:)', ...
    'VariableNames',
{'Router','Laptops','Telefonos','Dispositivos_Finales','Latencia_ms_Final
','Throughput_Gbps_Final','Perdida_p_Final','Energia_mJ_Final'});

disp('--- RESULTADOS GENERALES DE LA TOPOLOGÍA (Promedio t=1-60) ---');
disp(resultsGeneral);
disp('--- ESTADO FINAL DE ROUTERS (t=60s) ---');
disp(finalRouterState);

%% ===== GUARDAR REPORTE PDF (en .txt) =====
reportPath = fullfile(outputDir, 'Reporte_Estrella.txt');
try
    fid = fopen(reportPath, 'w', 'n', 'UTF-8');
    if fid == -1; error('No se pudo crear el archivo de reporte en %s',
reportPath); end
    disp(['Creando reporte de texto en: ', reportPath]);

    fprintf(fid,
'============================================================\n');
    fprintf(fid, ' REPORTE DE SIMULACIÓN - TOPOLOGÍA ESTRELLA
(CORREGIDO)\n');
    fprintf(fid,
'============================================================\n\n');
    fprintf(fid, 'Fecha de simulación: %s\n\n', datestr(now));
    fprintf(fid, '--- RESULTADOS GENERALES DE LA TOPOLOGÍA (Promedio t=1-
60) ---\n');
    reportOutput = evalc('disp(resultsGeneral)');
    fprintf(fid, '%s\n', reportOutput);
    fprintf(fid, '--- ESTADO FINAL DE ROUTERS (t=60s) ---\n');
    reportOutput = evalc('disp(finalRouterState)');
    fprintf(fid, '%s\n', reportOutput);
    fprintf(fid, '\n--- FIN DEL REPORTE ---');
    fclose(fid);
    disp('Reporte de texto guardado exitosamente.');
catch ME
    if fid ~= -1; fclose(fid); end
    disp('Error al guardar el reporte de texto:');
    disp(ME.message);
```

```
end

%% ================= GRÁFICOS (Estandarizados) =================
try; close(fig1); catch; end
fig2 = figure('Name', 'Resultados Promedio - Estrella', 'Position', [100
100 1200 600], 'Visible', 'off');
subplot(2,2,1); bar(avgLatency,'r'); title('Latencia Promedio (ms)');
ylabel('ms');
subplot(2,2,2); bar(avgThroughput,'b'); title('Throughput Promedio
(Gbps)'); ylabel('Gbps');
subplot(2,2,3); bar(avgLost,'FaceColor',[1 0.5 0]); title('Pérdida
Paquetes Promedio (%)'); ylabel('%');
subplot(2,2,4); bar(avgEnergy,'g'); title('Consumo Energético Promedio
(mJ)'); ylabel('mJ');
drawnow; saveas(fig2, fullfile(outputDir,
'Estrella_Resultados_Promedio.png')); close(fig2);

fig3 = figure('Name', 'Topologia Final - Estrella', 'Position', [100 100
900 600], 'Visible', 'off');
hFinal = plot(G,'Layout','force','NodeLabel',{});
title('Topologia Final - Estado en t=60s');
for r = 1:numRouters
    routerName = routers(r); finalState = routerStatesColor(end, r);
    if finalState == "Normal"; color = [0 0.7 0]; elseif finalState ==
"Alerta"; color = [1 0.7 0]; else; color = [0.85 0.1 0.1]; end
    nodeIdxInG = findnode(G, routerName);
    if nodeIdxInG > 0
        highlight(hFinal, nodeIdxInG, 'NodeColor', color, 'MarkerSize',
8);
        textFinal = sprintf('%s\n%s (%dd)', routerName, finalState,
devicesConnected(end, r));
        text(hFinal.XData(nodeIdxInG), hFinal.YData(nodeIdxInG)-0.05,
textFinal, 'HorizontalAlignment', 'center', 'VerticalAlignment', 'top',
'FontSize', 9, 'Color', 'k', 'FontWeight', 'bold');
    end
end
drawnow; saveas(fig3, fullfile(outputDir,
'Estrella_TopologiaFinal.png')); close(fig3);

fig4 = figure('Name', 'Evolución de Estados (Color) - Estrella',
'Position', [100 100 900 400], 'Visible', 'off');
normCount = sum(routerStatesColor=="Normal", 2); alertCount =
sum(routerStatesColor=="Alerta", 2); critCount =
sum(routerStatesColor=="Critico", 2);
plot(1:simTime, movmean(normCount,3), 'g', 'LineWidth', 1.5); hold on;
plot(1:simTime, movmean(alertCount,3), 'Color', [1 0.7 0], 'LineWidth',
1.5);
plot(1:simTime, movmean(critCount,3), 'r', 'LineWidth', 1.5);
xlabel('Tiempo (s)'); ylabel('Número de Routers'); grid on;
legend('Normal (Verde)', 'Alerta (Naranja)', 'Critico (Rojo)');
title('Evolución de estados de color de los routers');
drawnow; saveas(fig4, fullfile(outputDir,
'Estrella_EvolucionEstadosColor.png')); close(fig4);

fig6 = figure('Name', 'Evolución en Tiempo Real - Estrella', 'Position',
[100 100 1200 500], 'Visible', 'off');
```

```
mean_latency = mean(latency, 2,'omitnan'); mean_throughput =
mean(throughput, 2,'omitnan');
subplot(1,2,1); plot(1:simTime, mean_latency, 'r', 'LineWidth', 1.5);
title('Evolución Latencia Promedio (Activos)'); xlabel('Tiempo (s)');
ylabel('ms'); grid on; xlim([0 simTime]); ylim('auto');
subplot(1,2,2); plot(1:simTime, mean_throughput, 'b', 'LineWidth', 1.5);
title('Evolución Throughput Promedio (Activos)'); xlabel('Tiempo (s)');
ylabel('Gbps'); grid on; xlim([0 simTime]); ylim('auto');
drawnow; saveas(fig6, fullfile(outputDir,
'Estrella_EvolucionTiempoReal.png')); close(fig6);

disp('--- SIMULACIÓN DE TOPOLOGÍA ESTRELLA CORREGIDA Y FINALIZADA ---');
```

Anexo 6. código de simulación con ANN

```
%========================================================================
% SIMULACIÓN DE RED 5G CONGESTIÓN Y OPTIMIZACIÓN (VERSIÓN ANN)
% VERSIÓN FINAL CORREGIDA:
% 1. [MÉTRICA CORREGIDA] Calcula Fórmulas 2 (Error) y 3 (Eficacia).
% 2. [REPORTE PDF] Genera Resultados...Reporte.pdf
% 3. [MÉTRICA EF. ENE] Corregida a 'pJ/bit' para evitar valor 0.00.
% 4. [ARREGLO GRÁFICO] Evita el error Java NullPointerException.
% 5. [FIGURA EN VIVO] La figura 1 (en vivo) NO se cierra.
% 6. [LÓGICA GRADUAL] Routers 16-30 se introducen gradualmente.
% 7. [rng(1)] Se mantiene para datos deterministas.
%
%========================================================================
clear; clc; close all;
% --- Fija la semilla del generador aleatorio (DATOS CERTEROS) ---
rng(1, 'twister');
%% ===== PARAMETROS =====
numRoutersInit = 15;
numRoutersMax = 30; % Total final de routers
maxDevicesPerRouter = 70; % congestión máxima inicial
simTimeCong = 60;
simTimeOpt = 30;
simTimeTot = simTimeCong + simTimeOpt;
routersAll = strcat("Router", string(1:numRoutersMax));
% --- Parámetros para introducción gradual ---
routersToAdd = numRoutersMax - numRoutersInit; % 15 routers nuevos
addInterval = 5; % Añadir routers cada 5 segundos
routersPerInterval = 3; % Añadir 3 routers por intervalo
numIntervals = routersToAdd / routersPerInterval; % 5 intervalos
% --- Fin parámetros graduales ---
% Colores dispositivos
pcColor = [0 0.6 0]; % Verde para PCs
phoneColor = [1 0 0]; % Rojo para Móviles

%% ===== CLASIFICADOR DE COLOR (Árbol de Decisión Simple) ====
% Entrenado para clasificar según carga (usado solo para visualización)
trainDataColor = []; labelsColor = [];
critThresholdColor = 46; % >= 46 Crítico (Rojo)
alertThresholdColor = 26; % 26-45 Alerta (Naranja)
% <= 25 Normal (Verde)
for d = 1:maxDevicesPerRouter
    if d >= critThresholdColor
```

```
        stateColor = "Critico";
    elseif d >= alertThresholdColor
        stateColor = "Alerta";
    else
        stateColor = "Normal";
    end
    trainDataColor = [trainDataColor; d];
    labelsColor = [labelsColor; stateColor];
end
labelsColor = categorical(labelsColor);
Mdl_Color = fitctree(trainDataColor, labelsColor); % Árbol simple es
suficiente
%% ===== ANN (Entrenamiento para ENRUTAMIENTO dinámico) ====
disp('Preparando datos de entrenamiento para ANN (Enrutador)...');
datasetANN = []; labelsANN = []; % Labels para decisión de carga
critThresholdANN = 46; % Umbrales para el OBJETIVO del enrutador
alertThresholdANN = 26;
for d = 10:1:maxDevicesPerRouter
     for i = 1:10
        lat = 10 + (d^1.3)/10 + rand*3;
        loss = min(40, (d/3) + rand*3);
        thr = max(0.5, 9 - (d^1.1)/20 + rand*0.3);

        if d >= critThresholdANN
            targetStateANN = "Critico";
        elseif d >= alertThresholdANN
            targetStateANN = "Alerta";
        else
            targetStateANN = "Normal";
        end
        datasetANN = [datasetANN; d thr lat loss];
        labelsANN = [labelsANN; targetStateANN];
    end
end
X_matrix = datasetANN;
Y_cat = categorical(labelsANN);
classNamesANN = categories(Y_cat); % Guardar nombres de clases para ANN
Y_onehot = ind2vec(double(Y_cat)');
numObservaciones = size(X_matrix, 1);
cv = cvpartition(numObservaciones, 'HoldOut', 0.2);
idxTrain = training(cv);
idxValidation = test(cv);
XTrain = X_matrix(idxTrain, :)';
YTrain = Y_onehot(:, idxTrain);
XValidation = X_matrix(idxValidation, :)';
YValidation_idx = double(Y_cat(idxValidation));
net = patternnet(20);
net.trainParam.showWindow = false;
disp('Entrenando red ANN (Enrutador)...');
[Mdl_ANN, tr] = train(net, XTrain, YTrain);
YValidation_pred_vec = Mdl_ANN(XValidation);
YValidation_pred_idx = vec2ind(YValidation_pred_vec);
% --- Cálculos Fórmulas 2 y 3 ---
validationAccuracy = 100 * sum(YValidation_pred_idx' == YValidation_idx)
/ numel(YValidation_idx); % Eficacia (Fórmula 3)
annErrorPercent = 100 - validationAccuracy; % Error (Fórmula 2)
```

```
disp(['Eficacia de validación ANN (Fórmula 3): ',
num2str(validationAccuracy, '%.2f'), '%']);
disp(['Error de validación ANN (Fórmula 2): ', num2str(annErrorPercent,
'%.2f'), '%']);
% --- Fin Cálculos ---
%% ===== INICIALIZAR VARIABLES =====
latency = NaN(simTimeTot, numRoutersMax); % Usar NaN para routers
inactivos
throughput = NaN(simTimeTot, numRoutersMax);
lostPackets = NaN(simTimeTot, numRoutersMax);
energyConsumption = NaN(simTimeTot, numRoutersMax);
packetsSent = zeros(simTimeTot, numRoutersMax); % Usar 0 para inactivos
packetsReceived = zeros(simTimeTot, numRoutersMax);
devicesConnected = zeros(simTimeTot, numRoutersMax);
routerStatesColor = strings(simTimeTot, numRoutersMax); % Para colores
routerStatesColor(:,:) = "Inactivo"; % Inicializar como inactivo
finalPCs = zeros(numRoutersMax, 1);
finalPhones = zeros(numRoutersMax, 1);

%% ===== CONFIGURACIÓN DE GUARDADO (ANN) =====
savePath = 'C:\Users\jordy\Documents\Resultados\ANN\';
if ~exist(savePath, 'dir')
    mkdir(savePath);
end
disp('--- INICIANDO SIMULACIÓN ANN ---');

%% ===== FIGURA PRINCIPAL =====
% Guardamos el handle de la figura principal
fig1 = figure('Name', 'Simulación Red 5G - Congestión y Optimización
(ANN)', 'Position', [50 50 1600 800]);

%% ===== SIMULACIÓN =====
tic; % medir tiempo de enrutamiento
routersActivos = numRoutersInit; % Empezar con 15
for t = 1:simTimeTot

    if t > simTimeCong && mod(t - simTimeCong - 1, addInterval) == 0 &&
routersActivos < numRoutersMax
        routersNuevosEsteIntervalo = min(routersPerInterval,
numRoutersMax - routersActivos);
        routersActivos = routersActivos + routersNuevosEsteIntervalo;
        disp(['t=', num2str(t), 's: Activando ',
num2str(routersNuevosEsteIntervalo), ' routers nuevos. Total activos: ',
num2str(routersActivos)]);
    end

    Gtemp = graph();
    activeRouterNames = routersAll(1:routersActivos);
    Gtemp = addnode(Gtemp, activeRouterNames);
    for i = 1:routersActivos
        for j = i+1:routersActivos
            Gtemp = addedge(Gtemp, activeRouterNames(i),
activeRouterNames(j));
        end
    end
    for r = 1:routersActivos
```

```
        if t <= simTimeCong
             if r <= numRoutersInit
                numTotal = round((t/simTimeCong) * maxDevicesPerRouter);
             else
                 numTotal = 0;
             end
        else
            isNewRouter = (r > numRoutersInit);
            prevStateTime = max(1, t-1);

            if isNewRouter && devicesConnected(prevStateTime, r) == 0
                 features_ann = [0; 10; 5; 0];
                 predANN_Enrutador = "Normal";
            else
                 prevDev = devicesConnected(prevStateTime,r);
                 prevThr =
fillmissing(throughput(prevStateTime,r),'constant', 10);
                 prevLat =
fillmissing(latency(prevStateTime,r),'constant', 5);
                 prevLoss =
fillmissing(lostPackets(prevStateTime,r),'constant', 0);
                 features_ann = [prevDev; prevThr; prevLat; prevLoss];

                output_ann = Mdl_ANN(features_ann);
                [~, predIdx] = max(output_ann);
                predANN_Enrutador = classNamesANN{predIdx};
            end

            if strcmp(predANN_Enrutador, 'Critico')
                numTotal = randi([10, 20]);
            elseif strcmp(predANN_Enrutador, 'Alerta')
                numTotal = randi([20, 30]);
            else % "Normal"
                numTotal = randi([30, 45]);
            end

             if isNewRouter && devicesConnected(prevStateTime, r) == 0
                  numTotal = randi([5, 10]);
             elseif numTotal <= 0 && devicesConnected(prevStateTime,r) >
0 && ~isNewRouter
                 numTotal = randi([1,5]);
             end
        end
        devicesConnected(t,r) = numTotal;

        numPCs = randi([round(0.4*numTotal), round(0.6*numTotal)]);
        numPhones = numTotal - numPCs;
        if numPCs > 0; pcs = strcat("PC", string(r), "_",
string(1:numPCs)); Gtemp = addnode(Gtemp, pcs); for L = 1:numPCs; Gtemp =
addedge(Gtemp, activeRouterNames(r), pcs(L)); end; else; pcs =
strings(0); end
        if numPhones > 0; phones = strcat("Movil", string(r), "_",
string(1:numPhones)); Gtemp = addnode(Gtemp, phones); for P =
1:numPhones; Gtemp = addedge(Gtemp, activeRouterNames(r), phones(P));
end; else; phones = strings(0); end
```

```
        if t == simTimeTot
            finalPCs(r) = numPCs;
            finalPhones(r) = numPhones;
        end

        if numTotal > 0; trafficPCs = poissrnd(60, [1 numPCs]);
trafficPhones = poissrnd(90, [1 numPhones]) + randn(1,numPhones)*15;
trafficPhones(trafficPhones < 0) = 0; currentTraffic = sum(trafficPCs) +
sum(trafficPhones); packetsSent(t,r) = currentTraffic / 10;
        else; currentTraffic = 0; packetsSent(t,r) = 0; end

        if t <= simTimeCong; if numTotal > 0; packetsReceived(t,r) =
packetsSent(t,r) * (0.8 - 0.3*(numTotal/maxDevicesPerRouter)); else;
packetsReceived(t,r) = 0; end
        else; step = (t - simTimeCong) / simTimeOpt; if numTotal > 0;
packetsReceived(t,r) = packetsSent(t,r) * (0.5 + 0.5*step); else;
packetsReceived(t,r) = 0; end; end

        if numTotal > 0
            latency(t,r) = max(5, 7 + (numTotal^1.2)/12 + rand*1.5);
            lostPackets(t,r) = max(0, min(25, (numTotal/3.5) +
rand*1.5));
            throughput(t,r) = max(1, 9 - (numTotal^1.0)/25 + rand*0.2);
            energyConsumption(t,r) = max(0.4, 0.7 + 0.01*numTotal +
rand*0.1);
        else; latency(t,r) = NaN; lostPackets(t,r) = NaN; throughput(t,r)
= NaN; energyConsumption(t,r) = NaN; end
    end % Fin bucle for r

    for r = 1:routersActivos
        predColor = predict(Mdl_Color, devicesConnected(t,r));
        routerStatesColor(t,r) = string(predColor);
    end
    % ===== VISUALIZACIÓN =====
    subplot(1,2,1); cla; h = plot(Gtemp, 'Layout', 'force'); h.NodeLabel
= {};
    nodeIndicesPC = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name, "PC")));
    if ~isempty(nodeIndicesPC); highlight(h, nodeIndicesPC, 'NodeColor',
pcColor, 'MarkerSize', 3); end
    nodeIndicesMovil = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name, "Movil")));
    if ~isempty(nodeIndicesMovil); highlight(h, nodeIndicesMovil,
'NodeColor', phoneColor, 'MarkerSize', 3); end
    for r = 1:routersActivos
        state = routerStatesColor(t,r); nodeIdx = findnode(Gtemp,
activeRouterNames(r));
         if nodeIdx > 0; if state == "Normal"; color = [0 0.7 0]; elseif
state == "Alerta"; color = [1 0.7 0]; else; color = [0.85 0.1 0.1]; end
            highlight(h, nodeIdx, 'NodeColor', color, 'MarkerSize', 9);
end
    end
    if t <= simTimeCong; title(sprintf('Congestión - Segundo %d / %d (%d
Routers)', t, simTimeTot, routersActivos));
    else; title(sprintf('Optimización (ANN) - Segundo %d / %d (%d
Routers)', t, simTimeTot, routersActivos)); end
```

```
    subplot(2,2,2); cla; plot(1:t, mean(latency(1:t,1:routersActivos),
2,'omitnan'), 'r', 'LineWidth', 1.5); xline(60, '--k', 'Inicio
optimización'); title('Latencia promedio (En vivo)'); xlabel('Tiempo
(s)'); ylabel('ms'); grid on; xlim([0 simTimeTot]); ylim('auto');
    subplot(2,2,4); cla; plot(1:t, mean(throughput(1:t,1:routersActivos),
2,'omitnan'), 'b', 'LineWidth', 1.5); xline(60,'--k', 'Inicio
optimización'); title('Throughput promedio (En vivo)'); xlabel('Tiempo
(s)'); ylabel('Gbps'); grid on; xlim([0 simTimeTot]); ylim('auto');

    drawnow; pause(0.05);
end % Fin bucle for t

elapsedTime = toc*1000;
disp(['Tiempo total de simulación: ', num2str(elapsedTime/1000, '%.2f'),
' segundos']);

%% ===== RESULTADOS =====
avgLatencyBefore =
mean(mean(latency(1:simTimeCong,1:numRoutersInit),'omitnan'));
avgLatencyAfter = mean(mean(latency(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
avgThrBefore = mean(mean(throughput(1:simTimeCong,
1:numRoutersInit),'omitnan'));
avgThrAfter = mean(mean(throughput(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
avgLossBefore = mean(mean(lostPackets(1:simTimeCong,
1:numRoutersInit),'omitnan'));
avgLossAfter = mean(mean(lostPackets(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
avgEnergyBefore = mean(mean(energyConsumption(1:simTimeCong,
1:numRoutersInit),'omitnan'));
avgEnergyAfter = mean(mean(energyConsumption(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
resultsBefore = table(avgLatencyBefore, avgThrBefore, avgLossBefore,
avgEnergyBefore, 'VariableNames', {'Latencia_ms', 'Throughput_Gbps',
'Perdida_p', 'Energia_mJ'});
resultsAfter = table(avgLatencyAfter, avgThrAfter, avgLossAfter,
avgEnergyAfter, 'VariableNames', {'Latencia_ms', 'Throughput_Gbps',
'Perdida_p', 'Energia_mJ'});
improvement = ((resultsBefore{:,:} - resultsAfter{:,:}) ./
resultsBefore{:,:}) * 100;
improvement(2) = ((resultsAfter{:,2} - resultsBefore{:,2}) /
resultsBefore{:,2}) * 100;
resultsComparison = array2table(improvement, 'VariableNames',
{'Mejora_Latencia_p', 'Mejora_Throughput_p', 'Mejora_Perdida_p',
'Mejora_Energia_p'});

disp('--- RESULTADOS ANTES DE OPTIMIZACIÓN (Fase Congestión t=1-60) ---
'); disp(resultsBefore);
disp('--- RESULTADOS DESPUÉS DE OPTIMIZACIÓN (Fase ANN t=61-90) ---');
disp(resultsAfter);
disp('--- % DE MEJORA ENTRE CONGESTIÓN Y OPTIMIZACIÓN ---');
disp(resultsComparison);

%% ===== TABLA DETALLADA POR ROUTER =====
```

```
routerNames = routersAll(1:numRoutersMax)';
finalDevices = devicesConnected(end,1:numRoutersMax)';
finalSent = packetsSent(end,1:numRoutersMax)'/1e6;
finalReceived = packetsReceived(end,1:numRoutersMax)'/1e6;
finalEnergy = energyConsumption(end,1:numRoutersMax)';
finalLost = lostPackets(end,1:numRoutersMax)';
finalStateColor = routerStatesColor(end,1:numRoutersMax)';
routerDetails = table(routerNames, finalDevices,
finalPCs(1:numRoutersMax), finalPhones(1:numRoutersMax), finalSent,
finalReceived, finalLost, finalEnergy, finalStateColor, 'VariableNames',
{'Router', 'Dispositivos', 'PCs', 'Moviles', 'Paquetes_Enviados_M',
'Paquetes_Recibidos_M', 'Perdida_p', 'Consumo_Energia_mJ',
'Estado_Final_Color'});
disp('--- DETALLES POR ROUTER (Estado Final t=90s) ---');
disp(routerDetails);


%% ===== MÉTRICAS DE EFICIENCIA DEL ANN (Fórmulas 2 y 3) =====
% --- INICIO CORRECCIÓN MÉTRICA: Fórmulas 2, 3 y Ef. Ene ---
timeToOptimize = simTimeOpt;
latencyReduction = improvement(1);
lossReduction = improvement(3);

% Calcular Eficiencia Energética (J/bit -> pJ/bit)
avgEnergyJoule = avgEnergyAfter / 1000; % (mJ/s) -> (J/s) (promedio)
avgBitsPerSec = avgThrAfter * 1e9;    % (Gbit/s) -> (bit/s) (promedio)
if avgThrAfter > 0 && avgEnergyJoule > 0
    effEnergy_J_per_bit = avgEnergyJoule / avgBitsPerSec; % (J/bit)
else
    effEnergy_J_per_bit = 0;
end
effEnergy_pJ_per_bit = effEnergy_J_per_bit * 1e12; % Convertir J/bit a
pJ/bit

annAccuracyPercent = validationAccuracy; % Tu Fórmula 3 (Eficacia)
annErrorPercent = 100 - validationAccuracy; % Tu Fórmula 2 (Error)

efficiencyMetrics = table(elapsedTime, timeToOptimize, latencyReduction,
lossReduction, effEnergy_pJ_per_bit, annErrorPercent, annAccuracyPercent,
...
    'VariableNames', {'TiempoEnrutamiento_ms', 'TiempoOptimiz_s',
'ReduccionLatencia_p', 'ReduccionPerdida_p',
'EficienciaEnergetica_pJbit', 'ErrorValidacion_ANN_p',
'EficaciaValidacion_ANN_p'});
disp('--- MÉTRICAS DE EFICIENCIA DEL ANN (Enrutador) ---');
disp(efficiencyMetrics);
% --- FIN CORRECCIÓN MÉTRICA ---


%% ===== GUARDAR RESULTADOS DEL ANN =====
Resultados_ANN.Tablas.Before = resultsBefore;
Resultados_ANN.Tablas.After = resultsAfter;
Resultados_ANN.Tablas.Comparison = resultsComparison;
Resultados_ANN.Tablas.RouterDetails = routerDetails;
Resultados_ANN.Tablas.Efficiency = efficiencyMetrics;
Resultados_ANN.Series.Latency = latency;
```

```
Resultados_ANN.Series.Throughput = throughput;
Resultados_ANN.Series.Loss = lostPackets;
Resultados_ANN.Series.Energy = energyConsumption;
Resultados_ANN.Series.StatesColor = routerStatesColor;
Resultados_ANN.Modelo.Net = Mdl_ANN;
Resultados_ANN.Modelo.TrainInfo = tr;

save(fullfile(savePath, 'Resultados_ANN.mat'), 'Resultados_ANN');
writetable(resultsBefore, fullfile(savePath,
'Resultados_ANN_Before.csv'));
writetable(resultsAfter, fullfile(savePath, 'Resultados_ANN_After.csv'));
writetable(resultsComparison, fullfile(savePath,
'Resultados_ANN_Comparison.csv'));
writetable(routerDetails, fullfile(savePath,
'Resultados_ANN_RouterDetails.csv'));
writetable(efficiencyMetrics, fullfile(savePath,
'Resultados_ANN_Efficiency.csv'));

disp(['--- TODOS LOS RESULTADOS DEL ANN SE HAN GUARDADO EN: ', savePath,
' ---']);
%% ===== GRÁFICOS COMPARATIVOS (FIGURAS 2-5) =====
% Figura 2: Comparación Antes vs Después (con valores)
fig2 = figure('Name', 'Comparación Antes vs Después', 'Position', [100
100 1200 600], 'Visible', 'off'); % Crear invisible
subplot(2,2,1); b1 = bar([avgLatencyBefore avgLatencyAfter], 'r');
ylabel('ms'); title('Latencia promedio (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgLatencyBefore, avgLatencyAfter, 1])*1.2]); text(b1.XEndPoints,
b1.YEndPoints, sprintfc('%.2f ms', b1.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
subplot(2,2,2); b2 = bar([avgThrBefore avgThrAfter], 'b');
ylabel('Gbps'); title('Throughput promedio (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgThrBefore, avgThrAfter, 1])*1.2]); text(b2.XEndPoints,
b2.YEndPoints, sprintfc('%.2f Gbps', b2.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
subplot(2,2,3); b3 = bar([avgLossBefore avgLossAfter], 'FaceColor', [1
0.5 0]); ylabel('%'); title('Pérdida de paquetes (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgLossBefore, avgLossAfter, 1])*1.2]); text(b3.XEndPoints,
b3.YEndPoints, sprintfc('%.2f %%', b3.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
subplot(2,2,4); b4 = bar([avgEnergyBefore avgEnergyAfter], 'g');
ylabel('mJ'); title('Consumo energético (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgEnergyBefore, avgEnergyAfter, 0.1])*1.2]); text(b4.XEndPoints,
b4.YEndPoints, sprintfc('%.2f mJ', b4.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
drawnow; % Forzar dibujo
saveas(fig2, fullfile(savePath, 'ANN_ComparacionAntesDespues.png'));
close(fig2); % Cerrar después de guardar


% Figura 3: Topología Final (con texto)
fig3 = figure('Name', 'Topologia Final', 'Position', [100 100 900 600],
'Visible', 'off');
```

```
Gfinal = graph(); finalActiveRouters = routersAll(1:routersActivos);
Gfinal = addnode(Gfinal, finalActiveRouters);
for i = 1:routersActivos; for j = i+1:routersActivos; Gfinal =
addedge(Gfinal, finalActiveRouters(i), finalActiveRouters(j)); end; end
hFinal = plot(Gfinal, 'Layout', 'force', 'NodeLabel', {});
title('Topologia final - Estado Inicial (t=Congestión) vs Final
(t=Optimización)');
colorsFinal = zeros(routersActivos, 3);
for r = 1:routersActivos
    routerName = finalActiveRouters(r); routerIndexGlobal =
find(strcmp(routersAll, routerName));
    finalState = routerStatesColor(end, routerIndexGlobal);
    if finalState == "Normal"; colorsFinal(r,:) = [0 0.7 0]; elseif
finalState == "Alerta"; colorsFinal(r,:) = [1 0.7 0]; else;
colorsFinal(r,:) = [0.85 0.1 0.1]; end
    nodeIdxInGfinal = findnode(Gfinal, routerName);
    if nodeIdxInGfinal > 0
        highlight(hFinal, nodeIdxInGfinal, 'NodeColor', colorsFinal(r,:),
'MarkerSize', 8);
        if routerIndexGlobal <= numRoutersInit; textInicial =
sprintf('t60:%s(%dd)', routerStatesColor(simTimeCong, routerIndexGlobal),
devicesConnected(simTimeCong, routerIndexGlobal));
        else; textInicial = 'tc:Nuevo'; end
        textFinal = sprintf('to:%s(%dd)', routerStatesColor(end,
routerIndexGlobal), devicesConnected(end, routerIndexGlobal));
        text(hFinal.XData(nodeIdxInGfinal),
hFinal.YData(nodeIdxInGfinal)+0.06, textInicial, 'HorizontalAlignment',
'center', 'VerticalAlignment', 'bottom', 'FontSize', 9, 'Color', 'k',
'FontWeight', 'bold');
        text(hFinal.XData(nodeIdxInGfinal),
hFinal.YData(nodeIdxInGfinal)-0.06, textFinal, 'HorizontalAlignment',
'center', 'VerticalAlignment', 'top', 'FontSize', 9, 'Color', 'k',
'FontWeight', 'bold');
    end
end
drawnow; % Forzar dibujo
saveas(fig3, fullfile(savePath, 'ANN_TopologiaFinal.png'));
close(fig3); % Cerrar después de guardar

% Figura 4: Evolución de Estados (Color)
fig4 = figure('Name', 'Evolución de Estados (Color)', 'Position', [100
100 900 400], 'Visible', 'off');
normCount = zeros(simTimeTot, 1); alertCount = zeros(simTimeTot, 1);
critCount = zeros(simTimeTot, 1);
for t_idx = 1:simTimeTot
    currentActiveRouters = sum(devicesConnected(t_idx, :) > 0 |
~isnan(latency(t_idx,:)));
    if t_idx <= simTimeCong; currentActiveRouters = numRoutersInit; end;
if isnan(currentActiveRouters) || currentActiveRouters==0;
currentActiveRouters=1; end
    activeStates = routerStatesColor(t_idx, 1:currentActiveRouters);
    normCount(t_idx) = sum(activeStates == "Normal");
    alertCount(t_idx) = sum(activeStates == "Alerta");
    critCount(t_idx) = sum(activeStates == "Critico");
end
plot(1:simTimeTot, movmean(normCount,3), 'g', 'LineWidth', 1.5); hold on;
```

```
plot(1:simTimeTot, movmean(alertCount,3), 'Color', [1 0.7 0],
'LineWidth', 1.5); % Naranja
plot(1:simTimeTot, movmean(critCount,3), 'r', 'LineWidth', 1.5);
xline(60,'--k', 'Inicio optimización'); xlabel('Tiempo (s)');
ylabel('Número de Routers'); grid on;
legend('Normal (Verde)', 'Alerta (Naranja)', 'Critico (Rojo)');
title('Evolución de estados de color de los routers ACTIVOS');
drawnow; % Forzar dibujo
saveas(fig4, fullfile(savePath, 'ANN_EvolucionEstadosColor.png'));
close(fig4); % Cerrar después de guardar

% Figura 5: Métricas de eficiencia ANN (con valores)
% --- INICIO CORRECCIÓN MÉTRICA: Fórmulas 2 y Ef. Ene ---
fig5 = figure('Name', 'Métricas de eficiencia ANN', 'Position', [100 100
1000 400], 'Visible', 'off');
subplot(1,2,1);
metrics = [elapsedTime/1000, timeToOptimize, latencyReduction,
lossReduction, effEnergy_pJ_per_bit, annErrorPercent]; % Usar Error y
pJ/bit
labels = {'T. Sim (s)', 'T. Opt (s)', 'Red. Lat %', 'Red. Perd %', 'Ef.
Ene (pJ/bit)', 'Error Val. %'}; % Etiqueta de Error y pJ/bit
b5 = bar(metrics);
set(gca, 'XTickLabel', labels);
ylabel('Valores'); title('Métricas ANN (Enrutador)'); grid on;
ylim([0, max([metrics, 1])*1.2]);
text(b5.XEndPoints, b5.YData, sprintfc('%.2f', b5.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');

subplot(1,2,2);
theta = linspace(0, 2*pi, length(metrics)+1); rho = [metrics metrics(1)];
rho_norm = max(0, rho / max(rho) * 100);
polarplot(theta, rho_norm, '-ob', 'LineWidth', 1.5, 'MarkerFaceColor',
'b');
thetaticks(rad2deg(theta(1:end-1)));
thetaticklabels(labels);
rticks([25 50 75 100]);
text(theta(1:end-1).*1.15, rho_norm(1:end-1).*1.15, sprintfc('%.1f',
rho(1:end-1)),'HorizontalAlignment','center','FontSize',8);
title('Radar de métricas del ANN (Enrutador)');
drawnow; % Forzar dibujo
saveas(fig5, fullfile(savePath, 'ANN_MetricasEficiencia.png'));
close(fig5); % Cerrar después de guardar
% --- FIN CORRECCIÓN MÉTRICA ---
% Figura 6: Evolución en Tiempo Real (Gráficos de Línea)
fig6 = figure('Name', 'Evolución en Tiempo Real', 'Position', [100 100
1200 500], 'Visible', 'off');
mean_latency = zeros(simTimeTot, 1);
mean_throughput = zeros(simTimeTot, 1);
for t_idx = 1:simTimeTot
     if t_idx <= simTimeCong; currentActive = 1:numRoutersInit;
     else
         numActiveNow = numRoutersInit + floor((t_idx - simTimeCong -
1)/addInterval)*routersPerInterval + min(routersPerInterval, mod(t_idx -
simTimeCong -1, addInterval)+1);
         numActiveNow = min(numActiveNow, numRoutersMax);
         currentActive = 1:numActiveNow;
     end
```

```
     mean_latency(t_idx) = mean(latency(t_idx, currentActive),'omitnan');
     mean_throughput(t_idx) = mean(throughput(t_idx,
currentActive),'omitnan');
end
subplot(1,2,1); plot(1:simTimeTot, mean_latency, 'r', 'LineWidth', 1.5);
xline(60, '--k', 'Inicio optimización'); title('Evolución Latencia
Promedio (Activos)'); xlabel('Tiempo (s)'); ylabel('ms'); grid on;
xlim([0 simTimeTot]); ylim('auto');
subplot(1,2,2); plot(1:simTimeTot, mean_throughput, 'b', 'LineWidth',
1.5); xline(60, '--k', 'Inicio optimización'); title('Evolución
Throughput Promedio (Activos)'); xlabel('Tiempo (s)'); ylabel('Gbps');
grid on; xlim([0 simTimeTot]); ylim('auto');
drawnow; % Forzar dibujo
saveas(fig6, fullfile(savePath, 'ANN_EvolucionTiempoReal.png'));
close(fig6); % Cerrar después de guardar


% --- INICIO CORRECCIÓN PDF: Guardar Tablas en 1 solo archivo .txt ---
% Crear el nombre del archivo de reporte
reportPath = fullfile(savePath, 'Resultados_ANN_Reporte.txt');
try
    % Abrir el archivo de texto para escribir
    fid = fopen(reportPath, 'w', 'n', 'UTF-8');
    if fid == -1
        error('No se pudo crear el archivo de reporte en %s',
reportPath);
    end
    disp(['Creando reporte de texto en: ', reportPath]);

    % --- Escribir todas las tablas de resultados al archivo ---
    fprintf(fid,
'============================================================\n');
    fprintf(fid, ' REPORTE DE SIMULACIÓN - ENRUTADOR ANN\n');
    fprintf(fid,
'============================================================\n\n');
    fprintf(fid, 'Fecha de simulación: %s\n\n', datestr(now));

    fprintf(fid, '--- MÉTRICAS DE EFICIENCIA DEL ANN (Enrutador) ---\n');
    efficiencyOutput = evalc('disp(efficiencyMetrics)');
    fprintf(fid, '%s\n', efficiencyOutput);

    fprintf(fid, '--- RESULTADOS ANTES DE OPTIMIZACIÓN (Fase Congestión
t=1-60) ---\n');
    beforeOutput = evalc('disp(resultsBefore)');
    fprintf(fid, '%s\n', beforeOutput);

    fprintf(fid, '--- RESULTADOS DESPUÉS DE OPTIMIZACIÓN (Fase ANN t=61-
90) ---\n');
    afterOutput = evalc('disp(resultsAfter)');
    fprintf(fid, '%s\n', afterOutput);

    fprintf(fid, '--- %% DE MEJORA ENTRE CONGESTIÓN Y OPTIMIZACIÓN ---
\n');
    comparisonOutput = evalc('disp(resultsComparison)');
    fprintf(fid, '%s\n', comparisonOutput);
```

```
    fprintf(fid, '--- DETALLES POR ROUTER (Estado Final t=90s) ---\n');
    detailsOutput = evalc('disp(routerDetails)');
    fprintf(fid, '%s\n', detailsOutput);

    fprintf(fid, '\n--- FIN DEL REPORTE ---');

    % Cerrar el archivo
    fclose(fid);

    disp('Reporte de texto guardado exitosamente.');
    disp('*** INSTRUCCIÓN: Abra el archivo .txt y use "Imprimir a PDF"
para crear su reporte final. ***');

catch ME
    if fid ~= -1
        fclose(fid);
    end
    disp('Error al guardar el reporte de texto:');
    disp(ME.message);
end
% --- FIN CORRECCIÓN PDF ---

disp('--- SIMULACIÓN FINALIZADA ---');
```

Anexo 7. Código de simulación con KNN

```
================================================================================
% SIMULACIÓN DE RED 5G CONGESTIÓN Y OPTIMIZACIÓN (VERSIÓN KNN)
% BASADO EN EL SCRIPT FUNDAMENTAL (v5)
% 1. [MODELO] Enrutador ANN reemplazado por KNN ('fitcknn').
% 2. [MÉTRICA] Se usa 'kfoldLoss' para calcular Error (Fórmula 2)
%    y Eficacia (Fórmula 3) para una comparación justa.
% 3. [LÓGICA GRADUAL] Se mantiene la lógica de optimización gradual.
% 4. [rng(1)] Se mantiene para datos deterministas.
% 5. [RUTA] Guardado en carpeta KNN.
%
================================================================================
clear; clc; close all;
% --- Fija la semilla del generador aleatorio (DATOS CERTEROS) ---
rng(1, 'twister');

%% ===== PARAMETROS =====
numRoutersInit = 15;
numRoutersMax = 30; % Total final de routers
maxDevicesPerRouter = 70; % congestión máxima inicial
simTimeCong = 60;
simTimeOpt = 30;
simTimeTot = simTimeCong + simTimeOpt;
routersAll = strcat("Router", string(1:numRoutersMax));

% --- Parámetros para introducción gradual ---
routersToAdd = numRoutersMax - numRoutersInit; % 15 routers nuevos
addInterval = 5; % Añadir routers cada 5 segundos
```

```
routersPerInterval = 3; % Añadir 3 routers por intervalo
numIntervals = routersToAdd / routersPerInterval; % 5 intervalos
% --- Fin parámetros graduales ---

% Colores dispositivos
pcColor = [0 0.6 0]; % Verde para PCs
phoneColor = [1 0 0]; % Rojo para Móviles

%% ===== CLASIFICADOR DE COLOR (Árbol de Decisión Simple) ====
% Entrenado para clasificar según carga (usado solo para visualización)
trainDataColor = []; labelsColor = [];
critThresholdColor = 46; % >= 46 Crítico (Rojo)
alertThresholdColor = 26; % 26-45 Alerta (Naranja)
% <= 25 Normal (Verde)
for d = 1:maxDevicesPerRouter
    if d >= critThresholdColor
        stateColor = "Critico";
    elseif d >= alertThresholdColor
        stateColor = "Alerta";
    else
        stateColor = "Normal";
    end
    trainDataColor = [trainDataColor; d];
    labelsColor = [labelsColor; stateColor];
end
labelsColor = categorical(labelsColor);
Mdl_Color = fitctree(trainDataColor, labelsColor); % Árbol simple es
suficiente


%% ===== KNN (Entrenamiento para ENRUTAMIENTO dinámico) ====
% --- INICIO DE MODIFICACIÓN: ANN -> KNN ---
disp('Preparando datos de entrenamiento para KNN (Enrutador)...');
datasetKNN = []; labelsKNN = []; % Labels para decisión de carga
critThresholdKNN = 46; % Umbrales para el OBJETIVO del enrutador
alertThresholdKNN = 26;
for d = 10:1:maxDevicesPerRouter
     for i = 1:10
        lat = 10 + (d^1.3)/10 + rand*3;
        loss = min(40, (d/3) + rand*3);
        thr = max(0.5, 9 - (d^1.1)/20 + rand*0.3);

        if d >= critThresholdKNN
            targetStateKNN = "Critico";
        elseif d >= alertThresholdKNN
            targetStateKNN = "Alerta";
        else
            targetStateKNN = "Normal";
        end
        % Features para KNN: [dispositivos, throughput, latencia,
perdida]
        datasetKNN = [datasetKNN; d thr lat loss];
        labelsKNN = [labelsKNN; targetStateKNN];
    end
end
X_knn = datasetKNN; % Formato [Samples x Features] para fitcknn
```

```
Y_knn = categorical(labelsKNN);
classNamesKNN = categories(Y_knn); % Guardar nombres de clases para KNN

% Entrenamiento del clasificador KNN
disp('Entrenando modelo KNN (Enrutador)...');
% 'NumNeighbors'=5 es un valor común, 'Standardize'=true normaliza los
datos
Mdl_KNN = fitcknn(X_knn, Y_knn, 'NumNeighbors', 5, 'Standardize', true,
'ClassNames', classNamesKNN);

% Validación cruzada del KNN para obtener Error (Fórmula 2)
disp('Calculando Error/Eficacia del modelo KNN...');
cvKNN = crossval(Mdl_KNN, 'KFold', 5);
knnErrorPercent = kfoldLoss(cvKNN) * 100; % Error (Fórmula 2)
knnAccuracyPercent = (1 - kfoldLoss(cvKNN)) * 100; % Eficacia (Fórmula 3)

disp(['Eficacia de validación cruzada KNN (Fórmula 3): ',
num2str(knnAccuracyPercent, '%.2f'), '%']);
disp(['Error de validación cruzada KNN (Fórmula 2): ',
num2str(knnErrorPercent, '%.2f'), '%']);
% --- FIN DE MODIFICACIÓN ---


%% ===== INICIALIZAR VARIABLES =====
latency = NaN(simTimeTot, numRoutersMax); % Usar NaN para routers
inactivos
throughput = NaN(simTimeTot, numRoutersMax);
lostPackets = NaN(simTimeTot, numRoutersMax);
energyConsumption = NaN(simTimeTot, numRoutersMax);
packetsSent = zeros(simTimeTot, numRoutersMax); % Usar 0 para inactivos
packetsReceived = zeros(simTimeTot, numRoutersMax);
devicesConnected = zeros(simTimeTot, numRoutersMax);
routerStatesColor = strings(simTimeTot, numRoutersMax); % Para colores
routerStatesColor(:,:) = "Inactivo"; % Inicializar como inactivo
finalPCs = zeros(numRoutersMax, 1);
finalPhones = zeros(numRoutersMax, 1);

%% ===== CONFIGURACIÓN DE GUARDADO (KNN) =====
% --- RUTA ACTUALIZADA ---
savePath = 'C:\Users\jordy\Documents\Resultados\KNN\';
% --- FIN ACTUALIZACIÓN ---
if ~exist(savePath, 'dir')
    mkdir(savePath);
end
disp('--- INICIANDO SIMULACIÓN KNN ---');

%% ===== FIGURA PRINCIPAL =====
fig1 = figure('Name', 'Simulación Red 5G - Congestión y Optimización
(KNN)', 'Position', [50 50 1600 800]);

%% ===== SIMULACIÓN =====
tic; % medir tiempo de enrutamiento
routersActivos = numRoutersInit; % Empezar con 15

for t = 1:simTimeTot
```

```
    if t > simTimeCong && mod(t - simTimeCong - 1, addInterval) == 0 &&
routersActivos < numRoutersMax
        routersNuevosEsteIntervalo = min(routersPerInterval,
numRoutersMax - routersActivos);
        routersActivos = routersActivos + routersNuevosEsteIntervalo;
        disp(['t=', num2str(t), 's: Activando ',
num2str(routersNuevosEsteIntervalo), ' routers nuevos. Total activos: ',
num2str(routersActivos)]);
    end

    Gtemp = graph();
    activeRouterNames = routersAll(1:routersActivos);
    Gtemp = addnode(Gtemp, activeRouterNames);
    for i = 1:routersActivos
        for j = i+1:routersActivos
            Gtemp = addedge(Gtemp, activeRouterNames(i),
activeRouterNames(j));
        end
    end

    for r = 1:routersActivos

        if t <= simTimeCong
             if r <= numRoutersInit
                numTotal = round((t/simTimeCong) * maxDevicesPerRouter);
             else
                 numTotal = 0;
             end
        else
            isNewRouter = (r > numRoutersInit);
            prevStateTime = max(1, t-1);

            % --- INICIO DE MODIFICACIÓN: ANN -> KNN ---
            if isNewRouter && devicesConnected(prevStateTime, r) == 0
                 features_knn = [0, 10, 5, 0]; % Estado ideal [1 x 4]
                 predKNN_Enrutador = "Normal";
            else
                 prevDev = devicesConnected(prevStateTime,r);
                 prevThr =
fillmissing(throughput(prevStateTime,r),'constant', 10);
                 prevLat =
fillmissing(latency(prevStateTime,r),'constant', 5);
                 prevLoss =
fillmissing(lostPackets(prevStateTime,r),'constant', 0);
                 features_knn = [prevDev, prevThr, prevLat, prevLoss]; %
Formato [1 x 4]

                 % Predecir el estado con KNN (Enrutador)
                 predKNN_Enrutador = predict(Mdl_KNN, features_knn);
            end
            % --- FIN DE MODIFICACIÓN ---

            % Política de enrutamiento GRADUAL (NUEVO ESTÁNDAR)
            if strcmp(predKNN_Enrutador, 'Critico')
                numTotal = randi([10, 20]);
```

```
            elseif strcmp(predKNN_Enrutador, 'Alerta')
                numTotal = randi([20, 30]);
            else % "Normal"
                numTotal = randi([30, 45]);
            end

             if isNewRouter && devicesConnected(prevStateTime, r) == 0
                  numTotal = randi([5, 10]);
             elseif numTotal <= 0 && devicesConnected(prevStateTime,r) >
0 && ~isNewRouter
                 numTotal = randi([1,5]);
             end
        end
        devicesConnected(t,r) = numTotal;

        numPCs = randi([round(0.4*numTotal), round(0.6*numTotal)]);
        numPhones = numTotal - numPCs;
        if numPCs > 0; pcs = strcat("PC", string(r), "_",
string(1:numPCs)); Gtemp = addnode(Gtemp, pcs); for L = 1:numPCs; Gtemp =
addedge(Gtemp, activeRouterNames(r), pcs(L)); end; else; pcs =
strings(0); end
        if numPhones > 0; phones = strcat("Movil", string(r), "_",
string(1:numPhones)); Gtemp = addnode(Gtemp, phones); for P =
1:numPhones; Gtemp = addedge(Gtemp, activeRouterNames(r), phones(P));
end; else; phones = strings(0); end

        if t == simTimeTot
            finalPCs(r) = numPCs;
            finalPhones(r) = numPhones;
        end

        if numTotal > 0; trafficPCs = poissrnd(60, [1 numPCs]);
trafficPhones = poissrnd(90, [1 numPhones]) + randn(1,numPhones)*15;
trafficPhones(trafficPhones < 0) = 0; currentTraffic = sum(trafficPCs) +
sum(trafficPhones); packetsSent(t,r) = currentTraffic / 10;
        else; currentTraffic = 0; packetsSent(t,r) = 0; end

        if t <= simTimeCong; if numTotal > 0; packetsReceived(t,r) =
packetsSent(t,r) * (0.8 - 0.3*(numTotal/maxDevicesPerRouter)); else;
packetsReceived(t,r) = 0; end
        else; step = (t - simTimeCong) / simTimeOpt; if numTotal > 0;
packetsReceived(t,r) = packetsSent(t,r) * (0.5 + 0.5*step); else;
packetsReceived(t,r) = 0; end; end

        if numTotal > 0
            latency(t,r) = max(5, 7 + (numTotal^1.2)/12 + rand*1.5);
            lostPackets(t,r) = max(0, min(25, (numTotal/3.5) +
rand*1.5));
            throughput(t,r) = max(1, 9 - (numTotal^1.0)/25 + rand*0.2);
            energyConsumption(t,r) = max(0.4, 0.7 + 0.01*numTotal +
rand*0.1);
        else; latency(t,r) = NaN; lostPackets(t,r) = NaN; throughput(t,r)
= NaN; energyConsumption(t,r) = NaN; end
    end % Fin bucle for r

    for r = 1:routersActivos
```

```
        predColor = predict(Mdl_Color, devicesConnected(t,r));
        routerStatesColor(t,r) = string(predColor);
    end

    % ===== VISUALIZACIÓN =====
    subplot(1,2,1); cla; h = plot(Gtemp, 'Layout', 'force'); h.NodeLabel
= {};
    nodeIndicesPC = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name, "PC")));
    if ~isempty(nodeIndicesPC); highlight(h, nodeIndicesPC, 'NodeColor',
pcColor, 'MarkerSize', 3); end
    nodeIndicesMovil = findnode(Gtemp,
Gtemp.Nodes.Name(contains(Gtemp.Nodes.Name, "Movil")));
    if ~isempty(nodeIndicesMovil); highlight(h, nodeIndicesMovil,
'NodeColor', phoneColor, 'MarkerSize', 3); end
    for r = 1:routersActivos
        state = routerStatesColor(t,r); nodeIdx = findnode(Gtemp,
activeRouterNames(r));
         if nodeIdx > 0; if state == "Normal"; color = [0 0.7 0]; elseif
state == "Alerta"; color = [1 0.7 0]; else; color = [0.85 0.1 0.1]; end
           highlight(h, nodeIdx, 'NodeColor', color, 'MarkerSize', 9);
end
    end
    % --- INICIO DE MODIFICACIÓN: ANN -> KNN ---
    if t <= simTimeCong; title(sprintf('Congestión - Segundo %d / %d (%d
Routers)', t, simTimeTot, routersActivos));
    else; title(sprintf('Optimización (KNN) - Segundo %d / %d (%d
Routers)', t, simTimeTot, routersActivos)); end
    % --- FIN DE MODIFICACIÓN ---

    subplot(2,2,2); cla; plot(1:t, mean(latency(1:t,1:routersActivos),
2,'omitnan'), 'r', 'LineWidth', 1.5); xline(60, '--k', 'Inicio
optimización'); title('Latencia promedio (En vivo)'); xlabel('Tiempo
(s)'); ylabel('ms'); grid on; xlim([0 simTimeTot]); ylim('auto');
    subplot(2,2,4); cla; plot(1:t, mean(throughput(1:t,1:routersActivos),
2,'omitnan'), 'b', 'LineWidth', 1.5); xline(60,'--k', 'Inicio
optimización'); title('Throughput promedio (En vivo)'); xlabel('Tiempo
(s)'); ylabel('Gbps'); grid on; xlim([0 simTimeTot]); ylim('auto');

    drawnow; pause(0.05);
end % Fin bucle for t

elapsedTime = toc*1000;
disp(['Tiempo total de simulación: ', num2str(elapsedTime/1000, '%.2f'),
' segundos']);

%% ===== RESULTADOS =====
avgLatencyBefore =
mean(mean(latency(1:simTimeCong,1:numRoutersInit),'omitnan'));
avgLatencyAfter = mean(mean(latency(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
avgThrBefore = mean(mean(throughput(1:simTimeCong,
1:numRoutersInit),'omitnan'));
avgThrAfter = mean(mean(throughput(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
```

```
avgLossBefore = mean(mean(lostPackets(1:simTimeCong,
1:numRoutersInit),'omitnan'));
avgLossAfter = mean(mean(lostPackets(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
avgEnergyBefore = mean(mean(energyConsumption(1:simTimeCong,
1:numRoutersInit),'omitnan'));
avgEnergyAfter = mean(mean(energyConsumption(simTimeCong+1:end,
1:numRoutersMax),'omitnan'));
resultsBefore = table(avgLatencyBefore, avgThrBefore, avgLossBefore,
avgEnergyBefore, 'VariableNames', {'Latencia_ms', 'Throughput_Gbps',
'Perdida_p', 'Energia_mJ'});
resultsAfter = table(avgLatencyAfter, avgThrAfter, avgLossAfter,
avgEnergyAfter, 'VariableNames', {'Latencia_ms', 'Throughput_Gbps',
'Perdida_p', 'Energia_mJ'});
improvement = ((resultsBefore{:,:} - resultsAfter{:,:}) ./
resultsBefore{:,:}) * 100;
improvement(2) = ((resultsAfter{:,2} - resultsBefore{:,2}) /
resultsBefore{:,2}) * 100;
resultsComparison = array2table(improvement, 'VariableNames',
{'Mejora_Latencia_p', 'Mejora_Throughput_p', 'Mejora_Perdida_p',
'Mejora_Energia_p'});

disp('--- RESULTADOS ANTES DE OPTIMIZACIÓN (Fase Congestión t=1-60) ---
'); disp(resultsBefore);
disp('--- RESULTADOS DESPUÉS DE OPTIMIZACIÓN (Fase KNN t=61-90) ---');
disp(resultsAfter);
disp('--- % DE MEJORA ENTRE CONGESTIÓN Y OPTIMIZACIÓN ---');
disp(resultsComparison);

%% ===== TABLA DETALLADA POR ROUTER =====
routerNames = routersAll(1:numRoutersMax)';
finalDevices = devicesConnected(end,1:numRoutersMax)';
finalSent = packetsSent(end,1:numRoutersMax)'/1e6;
finalReceived = packetsReceived(end,1:numRoutersMax)'/1e6;
finalEnergy = energyConsumption(end,1:numRoutersMax)';
finalLost = lostPackets(end,1:numRoutersMax)';
finalStateColor = routerStatesColor(end,1:numRoutersMax)';
routerDetails = table(routerNames, finalDevices,
finalPCs(1:numRoutersMax), finalPhones(1:numRoutersMax), finalSent,
finalReceived, finalLost, finalEnergy, finalStateColor, 'VariableNames',
{'Router', 'Dispositivos', 'PCs', 'Moviles', 'Paquetes_Enviados_M',
'Paquetes_Recibidos_M', 'Perdida_p', 'Consumo_Energia_mJ',
'Estado_Final_Color'});
disp('--- DETALLES POR ROUTER (Estado Final t=90s) ---');
disp(routerDetails);


%% ===== MÉTRICAS DE EFICIENCIA DEL KNN (Fórmulas 2 y 3) =====
% --- INICIO DE MODIFICACIÓN: ANN -> KNN ---
timeToOptimize = simTimeOpt;
latencyReduction = improvement(1);
lossReduction = improvement(3);

avgEnergyJoule = avgEnergyAfter / 1000; % (mJ/s) -> (J/s)
avgBitsPerSec = avgThrAfter * 1e9;    % (Gbit/s) -> (bit/s)
```

```
if avgThrAfter > 0 && avgEnergyJoule > 0; effEnergy_J_per_bit =
avgEnergyJoule / avgBitsPerSec; else; effEnergy_J_per_bit = 0; end
effEnergy_pJ_per_bit = effEnergy_J_per_bit * 1e12; % Convertir J/bit a
pJ/bit

knnAccuracyPercent = knnAccuracyPercent; % Tu Fórmula 3 (Eficacia)
knnErrorPercent = knnErrorPercent; % Tu Fórmula 2 (Error)

efficiencyMetrics = table(elapsedTime, timeToOptimize, latencyReduction,
lossReduction, effEnergy_pJ_per_bit, knnErrorPercent, knnAccuracyPercent,
...
    'VariableNames', {'TiempoEnrutamiento_ms', 'TiempoOptimiz_s',
'ReduccionLatencia_p', 'ReduccionPerdida_p',
'EficienciaEnergetica_pJbit', 'ErrorValidacion_KNN_p',
'EficaciaValidacion_KNN_p'});
disp('--- MÉTRICAS DE EFICIENCIA DEL KNN (Enrutador) ---');
disp(efficiencyMetrics);
% --- FIN DE MODIFICACIÓN ---


%% ===== GUARDAR RESULTADOS DEL KNN =====
% --- INICIO DE MODIFICACIÓN: ANN -> KNN ---
Resultados_KNN.Tablas.Before = resultsBefore;
Resultados_KNN.Tablas.After = resultsAfter;
Resultados_KNN.Tablas.Comparison = resultsComparison;
Resultados_KNN.Tablas.RouterDetails = routerDetails;
Resultados_KNN.Tablas.Efficiency = efficiencyMetrics;
Resultados_KNN.Series.Latency = latency;
Resultados_KNN.Series.Throughput = throughput;
Resultados_KNN.Series.Loss = lostPackets;
Resultados_KNN.Series.Energy = energyConsumption;
Resultados_KNN.Series.StatesColor = routerStatesColor;
Resultados_KNN.Modelo.KNN_Model = Mdl_KNN;
Resultados_KNN.Modelo.CV_Info = cvKNN;

save(fullfile(savePath, 'Resultados_KNN.mat'), 'Resultados_KNN');
writetable(resultsBefore, fullfile(savePath,
'Resultados_KNN_Before.csv'));
writetable(resultsAfter, fullfile(savePath, 'Resultados_KNN_After.csv'));
writetable(resultsComparison, fullfile(savePath,
'Resultados_KNN_Comparison.csv'));
writetable(routerDetails, fullfile(savePath,
'Resultados_KNN_RouterDetails.csv'));
writetable(efficiencyMetrics, fullfile(savePath,
'Resultados_KNN_Efficiency.csv'));

disp(['--- TODOS LOS RESULTADOS DEL KNN SE HAN GUARDADO EN: ', savePath,
' ---']);
% --- FIN DE MODIFICACIÓN ---

%% ===== GRÁFICOS COMPARATIVOS (FIGURAS 2-5) =====

% La figura 1 (en vivo) se deja abierta
fig2 = figure('Name', 'Comparación Antes vs Después', 'Position', [100
100 1200 600], 'Visible', 'off');
```

```
subplot(2,2,1); b1 = bar([avgLatencyBefore avgLatencyAfter], 'r');
ylabel('ms'); title('Latencia promedio (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgLatencyBefore, avgLatencyAfter, 1])*1.2]); text(b1.XEndPoints,
b1.YEndPoints, sprintfc('%.2f ms', b1.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
subplot(2,2,2); b2 = bar([avgThrBefore avgThrAfter], 'b');
ylabel('Gbps'); title('Throughput promedio (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgThrBefore, avgThrAfter, 1])*1.2]); text(b2.XEndPoints,
b2.YEndPoints, sprintfc('%.2f Gbps', b2.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
subplot(2,2,3); b3 = bar([avgLossBefore avgLossAfter], 'FaceColor', [1
0.5 0]); ylabel('%'); title('Pérdida de paquetes (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgLossBefore, avgLossAfter, 1])*1.2]); text(b3.XEndPoints,
b3.YEndPoints, sprintfc('%.2f %%', b3.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
subplot(2,2,4); b4 = bar([avgEnergyBefore avgEnergyAfter], 'g');
ylabel('mJ'); title('Consumo energético (General)'); set(gca,
'XTickLabel', {'Congestión', 'Optimización'}); ylim([0,
max([avgEnergyBefore, avgEnergyAfter, 0.1])*1.2]); text(b4.XEndPoints,
b4.YEndPoints, sprintfc('%.2f mJ', b4.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');
drawnow; saveas(fig2, fullfile(savePath,
'KNN_ComparacionAntesDespues.png')); close(fig2);


fig3 = figure('Name', 'Topologia Final', 'Position', [100 100 900 600],
'Visible', 'off');
Gfinal = graph(); finalActiveRouters = routersAll(1:routersActivos);
Gfinal = addnode(Gfinal, finalActiveRouters);
for i = 1:routersActivos; for j = i+1:routersActivos; Gfinal =
addedge(Gfinal, finalActiveRouters(i), finalActiveRouters(j)); end; end
hFinal = plot(Gfinal, 'Layout', 'force', 'NodeLabel', {});
title('Topologia final - Estado Inicial (t=Congestión) vs Final
(t=Optimización)');
colorsFinal = zeros(routersActivos, 3);
for r = 1:routersActivos
    routerName = finalActiveRouters(r); routerIndexGlobal =
find(strcmp(routersAll, routerName));
    finalState = routerStatesColor(end, routerIndexGlobal);
    if finalState == "Normal"; colorsFinal(r,:) = [0 0.7 0]; elseif
finalState == "Alerta"; colorsFinal(r,:) = [1 0.7 0]; else;
colorsFinal(r,:) = [0.85 0.1 0.1]; end
    nodeIdxInGfinal = findnode(Gfinal, routerName);
    if nodeIdxInGfinal > 0
        highlight(hFinal, nodeIdxInGfinal, 'NodeColor', colorsFinal(r,:),
'MarkerSize', 8);
        if routerIndexGlobal <= numRoutersInit; textInicial =
sprintf('t60:%s(%dd)', routerStatesColor(simTimeCong, routerIndexGlobal),
devicesConnected(simTimeCong, routerIndexGlobal));
        else; textInicial = 'tc:Nuevo'; end
        textFinal = sprintf('to:%s(%dd)', routerStatesColor(end,
routerIndexGlobal), devicesConnected(end, routerIndexGlobal));
        text(hFinal.XData(nodeIdxInGfinal),
hFinal.YData(nodeIdxInGfinal)+0.06, textInicial, 'HorizontalAlignment',
```

```
'center', 'VerticalAlignment', 'bottom', 'FontSize', 9, 'Color', 'k',
'FontWeight', 'bold');
        text(hFinal.XData(nodeIdxInGfinal),
hFinal.YData(nodeIdxInGfinal)-0.06, textFinal, 'HorizontalAlignment',
'center', 'VerticalAlignment', 'top', 'FontSize', 9, 'Color', 'k',
'FontWeight', 'bold');
    end
end
drawnow; saveas(fig3, fullfile(savePath, 'KNN_TopologiaFinal.png'));
close(fig3);

fig4 = figure('Name', 'Evolución de Estados (Color)', 'Position', [100
100 900 400], 'Visible', 'off');
normCount = zeros(simTimeTot, 1); alertCount = zeros(simTimeTot, 1);
critCount = zeros(simTimeTot, 1);
for t_idx = 1:simTimeTot
    currentActiveRouters = sum(devicesConnected(t_idx, :) > 0 |
~isnan(latency(t_idx,:)));
    if t_idx <= simTimeCong; currentActiveRouters = numRoutersInit; end;
if isnan(currentActiveRouters) || currentActiveRouters==0;
currentActiveRouters=1; end
    activeStates = routerStatesColor(t_idx, 1:currentActiveRouters);
    normCount(t_idx) = sum(activeStates == "Normal"); alertCount(t_idx) =
sum(activeStates == "Alerta"); critCount(t_idx) = sum(activeStates ==
"Critico");
end
plot(1:simTimeTot, movmean(normCount,3), 'g', 'LineWidth', 1.5); hold on;
plot(1:simTimeTot, movmean(alertCount,3), 'Color', [1 0.7 0],
'LineWidth', 1.5);
plot(1:simTimeTot, movmean(critCount,3), 'r', 'LineWidth', 1.5);
xline(60,'--k', 'Inicio optimización'); xlabel('Tiempo (s)');
ylabel('Número de Routers'); grid on;
legend('Normal (Verde)', 'Alerta (Naranja)', 'Critico (Rojo)');
title('Evolución de estados de color de los routers ACTIVOS');
drawnow; saveas(fig4, fullfile(savePath,
'KNN_EvolucionEstadosColor.png')); close(fig4);

% --- INICIO DE MODIFICACIÓN: ANN -> KNN ---
fig5 = figure('Name', 'Métricas de eficiencia KNN', 'Position', [100 100
1000 400], 'Visible', 'off');
subplot(1,2,1);
metrics = [elapsedTime/1000, timeToOptimize, latencyReduction,
lossReduction, effEnergy_pJ_per_bit, knnErrorPercent]; % Usar Error y
pJ/bit
labels = {'T. Sim (s)', 'T. Opt (s)', 'Red. Lat %', 'Red. Perd %', 'Ef.
Ene (pJ/bit)', 'Error Val. %'}; % Etiqueta de Error y pJ/bit
b5 = bar(metrics);
set(gca, 'XTickLabel', labels);
ylabel('Valores'); title('Métricas KNN (Enrutador)'); grid on;
ylim([0, max([metrics, 1])*1.2]);
text(b5.XEndPoints, b5.YData, sprintfc('%.2f', b5.YData),
'HorizontalAlignment','center', 'VerticalAlignment','bottom');

subplot(1,2,2);
theta = linspace(0, 2*pi, length(metrics)+1); rho = [metrics metrics(1)];
rho_norm = max(0, rho / max(rho) * 100);
```

```
polarplot(theta, rho_norm, '-ob', 'LineWidth', 1.5, 'MarkerFaceColor',
'b');
thetaticks(rad2deg(theta(1:end-1)));
thetaticklabels(labels);
rticks([25 50 75 100]);
text(theta(1:end-1).*1.15, rho_norm(1:end-1).*1.15, sprintfc('%.1f',
rho(1:end-1)),'HorizontalAlignment','center','FontSize',8);
title('Radar de métricas del KNN (Enrutador)');
drawnow; saveas(fig5, fullfile(savePath, 'KNN_MetricasEficiencia.png'));
close(fig5);
% --- FIN DE MODIFICACIÓN ---


fig6 = figure('Name', 'Evolución en Tiempo Real', 'Position', [100 100
1200 500], 'Visible', 'off');
mean_latency = zeros(simTimeTot, 1);
mean_throughput = zeros(simTimeTot, 1);
for t_idx = 1:simTimeTot
     if t_idx <= simTimeCong; currentActive = 1:numRoutersInit;
     else
         numActiveNow = numRoutersInit + floor((t_idx - simTimeCong -
1)/addInterval)*routersPerInterval + min(routersPerInterval, mod(t_idx -
simTimeCong -1, addInterval)+1);
         numActiveNow = min(numActiveNow, numRoutersMax);
         currentActive = 1:numActiveNow;
     end
     mean_latency(t_idx) = mean(latency(t_idx, currentActive),'omitnan');
     mean_throughput(t_idx) = mean(throughput(t_idx,
currentActive),'omitnan');
end
subplot(1,2,1); plot(1:simTimeTot, mean_latency, 'r', 'LineWidth', 1.5);
xline(60, '--k', 'Inicio optimización'); title('Evolución Latencia
Promedio (Activos)'); xlabel('Tiempo (s)'); ylabel('ms'); grid on;
xlim([0 simTimeTot]); ylim('auto');
subplot(1,2,2); plot(1:simTimeTot, mean_throughput, 'b', 'LineWidth',
1.5); xline(60, '--k', 'Inicio optimización'); title('Evolución
Throughput Promedio (Activos)'); xlabel('Tiempo (s)'); ylabel('Gbps');
grid on; xlim([0 simTimeTot]); ylim('auto');
drawnow; saveas(fig6, fullfile(savePath, 'KNN_EvolucionTiempoReal.png'));
close(fig6);


% --- GUARDAR REPORTE PDF (en .txt) ---
reportPath = fullfile(savePath, 'Resultados_KNN_Reporte.txt');
try
    fid = fopen(reportPath, 'w', 'n', 'UTF-8');
    if fid == -1; error('No se pudo crear el archivo de reporte en %s',
reportPath); end
    disp(['Creando reporte de texto en: ', reportPath]);

    % --- Escribir todas las tablas de resultados al archivo ---
    fprintf(fid,
'============================================================\n');
    fprintf(fid, ' REPORTE DE SIMULACIÓN - ENRUTADOR KNN\n');
    fprintf(fid,
'============================================================\n\n');
```

```
    fprintf(fid, 'Fecha de simulación: %s\n\n', datestr(now));

    fprintf(fid, '--- MÉTRICAS DE EFICIENCIA DEL KNN (Enrutador) ---\n');
    efficiencyOutput = evalc('disp(efficiencyMetrics)');
    fprintf(fid, '%s\n', efficiencyOutput);

    fprintf(fid, '--- RESULTADOS ANTES DE OPTIMIZACIÓN (Fase Congestión
t=1-60) ---\n');
    beforeOutput = evalc('disp(resultsBefore)');
    fprintf(fid, '%s\n', beforeOutput);

    fprintf(fid, '--- RESULTADOS DESPUÉS DE OPTIMIZACIÓN (Fase KNN t=61-
90) ---\n');
    afterOutput = evalc('disp(resultsAfter)');
    fprintf(fid, '%s\n', afterOutput);

    fprintf(fid, '--- %% DE MEJORA ENTRE CONGESTIÓN Y OPTIMIZACIÓN ---
\n');
    comparisonOutput = evalc('disp(resultsComparison)');
    fprintf(fid, '%s\n', comparisonOutput);

    fprintf(fid, '--- DETALLES POR ROUTER (Estado Final t=90s) ---\n');
    detailsOutput = evalc('disp(routerDetails)');
    fprintf(fid, '%s\n', detailsOutput);

    fprintf(fid, '\n--- FIN DEL REPORTE ---');
    fclose(fid);

    disp('Reporte de texto guardado exitosamente.');
    disp('*** INSTRUCCIÓN: Abra el archivo .txt y use "Imprimir a PDF"
para crear su reporte final. ***');
catch ME
    if fid ~= -1; fclose(fid); end
    disp('Error al guardar el reporte de texto:');
    disp(ME.message);
end
% --- FIN REPORTE ---
disp('--- SIMULACIÓN FINALIZADA ---');
```