Evaluación Comparativa de Modelos de Aprendizaje Automático para la Clasificación de Tráfico de Red utilizando Redes Neuronales Densas y Regresión Logística Multiclase sobre CIC-IDS2017

M. A. Pérez Ávila
Departamento de Computación
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
A01369908@tec.mx

A. M. Sánchez Serratos
Departamento de Computación
Instituto Tecnológico y de Estudios Superiores de Monterrey
Ciudad de México, México
A01771843@tec.mx

Resumen **- La clasificación precisa de tráfico de red es un componente crítico en la detección de amenazas cibernéticas. Este estudio compara el desempeño de un modelo lineal clásico, la Regresión Logística, con una red neuronal densa de dos capas (RN1) para la identificación de diferentes tipos de ataques y tráfico legítimo. Los resultados muestran que RN1 superó consistentemente a la Regresión Logística, alcanzando un accuracy de 0.968 y un macro F1-score de 0.968 frente a 0.909 para el modelo lineal. El análisis por clase evidencia que RN1 maneja mejor las clases parcialmente solapadas, como Normal Traffic, Bots y Web Attacks, mientras que ambas arquitecturas clasifican casi perfectamente DDoS y Port Scanning. RN1 también asigna probabilidades más concentradas a la clase predicha, reflejando mayor certeza en sus decisiones. La validación cruzada K-Fold (k=5) confirma su capacidad de generalización con un accuracy promedio de 0.942 (IC 95%: [0.907, 0.976]). Estos resultados destacan la ventaja de redes neuronales densas de complejidad moderada frente a modelos lineales en aplicaciones críticas de ciberseguridad.**

Índice de Términos **- Aprendizaje Automático, Clasificación de Tráfico de Red, CIC-IDS2017, Ciberseguridad, Detección de Intrusiones, Modelos Predictivos.**

I. Introducción

La creciente dependencia de servicios digitales y la expansión de infraestructuras críticas conectadas a redes han elevado significativamente la superficie de exposición a ciberamenazas. En este contexto, los Sistemas de Detección de Intrusos (IDS) se han consolidado como un componente fundamental en la arquitectura de seguridad de red moderna. Tradicionalmente, estos sistemas han operado basándose en firmas predefinidas de ataques conocidos, una metodología efectiva contra amenazas identificadas, pero inherentemente incapaz de detectar ataques zero-day o variantes novedosas de malware.

El aprendizaje automático (Machine Learning - ML) emerge como un paradigma disruptivo para superar estas limitaciones, permitiendo a los IDS analizar patrones de tráfico de red, aprender comportamientos normales y anómalos, y realizar clasificaciones inteligentes con una capacidad de adaptación superior.

Recibido el 18 de septiembre de 2025; versión preliminar para evaluación académica.

La aplicación de algoritmos de ML para la clasificación de tráfico promete una detección proactiva de intrusiones, reduciendo la dependencia de actualizaciones manuales de firmas y mejorando la resiliencia de la red.

No obstante, la efectividad de un modelo de ML está sujeta a múltiples factores, incluyendo la selección del algoritmo, la calidad y representatividad del conjunto de datos de entrenamiento, y la naturaleza de las características (features) extraídas del tráfico de red. En este ámbito, existe una necesidad continua de investigación empírica y evaluación comparativa que guíe la selección e implementación de modelos óptimos para entornos operativos específicos, particularmente en escenarios de clasificación multiclase donde se requiere discriminar entre múltiples categorías de ataques y tráfico benigno.

El presente estudio presenta una evaluación comparativa entre dos modelos de aprendizaje automático representativos de paradigmas distintos: una Red Neuronal Densa (DNN), como ejemplo de un aproximador de funciones no lineal de alta capacidad, y la Regresión Logística Multiclase (MLR), un modelo lineal robusto y eficiente computacionalmente. El experimento se conduce utilizando el conjunto de datos CIC-IDS2017 [1], un referente público moderno y ampliamente citado que contiene tráfico de red realista con tráfico benigno y ataques contemporáneos perpetrados bajo un escenario configurado.

El principal objetivo de este estudio es determinar el rendimiento comparativo de estos modelos en la tarea de clasificación multiclase de tráfico de red, evaluando métricas clave como precisión, exhaustividad (recall), puntuación F1 y exactitud (accuracy). Los resultados buscan proporcionar insights valiosos sobre las compensaciones (trade-offs) entre complejidad computacional y poder predictivo, ofreciendo una base empírica para el diseño de IDS basados en ML que sean a la vez precisos y factibles de implementar.

II. Marco Teórico

A. Ciberseguridad y Detección de Intrusiones

La ciberseguridad en redes constituye una disciplina crítica orientada a la protección de los activos de información mediante la garantía de los principios de confidencialidad, integridad y disponibilidad de los datos que transitan a través de la infraestructura de red [2]. La creciente sofisticación y el volumen de las ciberamenazas representan un desafío permanente, exigiendo mecanismos de defensa proactivos y adaptativos para la protección de sistemas interconectados.

El espectro de amenazas en entornos de red es diverso e incluye categorías como los ataques de denegación de servicio (DoS/DDoS), diseñados para comprometer la disponibilidad de los recursos; ataques

de infiltración y exfiltración de datos, que violan la confidencialidad; intentos de acceso no autorizado mediante fuerza bruta; exploits basados en web como inyecciones SQL y Cross-Site Scripting (XSS); y la propagación de software malicioso (malware). Esta variedad de vectores de ataque demanda soluciones de seguridad capaces de analizar grandes volúmenes de tráfico para identificar patrones tanto conocidos como novedosos.

Los Sistemas de Detección de Intrusos (IDS) se erigen como una tecnología fundamental en este contexto. Su función principal es la monitorización continua del tráfico de red o de la actividad del sistema para identificar intentos de acceso no autorizado, uso inapropiado o violaciones de políticas de seguridad. Desde una perspectiva metodológica, los IDS se clasifican en dos paradigmas principales [3]. Los sistemas basados en firmas operan mediante la comparación de la actividad observada con una base de datos de patrones de ataques conocidos. Si bien esta aproximación es altamente efectiva para la identificación de amenazas previamente catalogadas, presenta una limitación inherente: su incapacidad para detectar ataques zero-day o variantes no conocidas. Por otro lado, los sistemas basados en anomalías construyen un modelo del comportamiento normal esperado del sistema o red. Cualquier desviación significativa de este perfil basal es señalada como potencialmente maliciosa. Este último enfoque, si bien conceptualmente superior para la detección de ataques novedosos, históricamente ha enfrentado desafíos relacionados con altas tasas de falsos positivos. La investigación actual se centra en el empleo de técnicas de aprendizaje automático para refinar la precisión de los modelos de detección de anomalías, que es el ámbito de contribución del presente trabajo.

B. Tráfico de Red y Análisis de Datos

El tráfico de red constituye la manifestación primaria de la actividad en sistemas de comunicación, representado mediante flujos de paquetes discretos que transportan información de usuario y metadatos de control. Cada paquete encapsula dos componentes esenciales: la carga útil (payload) con información de usuario y encabezados de protocolo que incorporan metadatos críticos para la entrega y procesamiento de los datos. Estos metadatos incluyen direcciones IP de origen y destino, puertos de transporte, flags de control TCP/UDP, y marcas de tiempo, los cuales constituyen la base forense para el análisis de seguridad [4]. La examinación sistemática de estos atributos mediante técnicas de inspección profunda de paquetes permite caracterizar el comportamiento normal de la red e identificar desviaciones potencialmente maliciosas.

La transición desde el tráfico de red crudo hacia un espacio de características analíticamente tratable requiere un proceso de ingeniería de características metodológico. La aproximación predominante en la literatura especializada utiliza el paradigma de flujos de red (network flows), donde conjuntos de paquetes relacionados se agregan en entidades lógicas basadas en tuplas quintuplas (dirección IP origen, dirección IP destino, puerto origen, puerto destino, protocolo). De estos flujos se extraen tres categorías principales de características discriminativas:

- Métricas Volumétricas: Número total de paquetes, bytes transmitidos en ambas direcciones, y ratios de asimetría en el volumen de datos.
- Métricas Temporales: Duración del flujo, intervalos inter-paquete, tasas de transferencia instantáneas y promedio.
- Métricas Estadísticas: Momentos de orden superior (media, varianza, asimetría) de distribuciones de tamaño de paquetes, y distribuciones de flags TCP.

Este proceso de transformación reduce la dimensionalidad inherente del tráfico de red mientras preserva patrones discriminativos esenciales para la clasificación.

La validez ecológica (es decir, la capacidad del modelo para funcionar correctamente en situaciones del mundo real) de los modelos de detección depende críticamente de la representatividad estadística y la integridad del conjunto de entrenamiento. Un conjunto de datos ideal debe exhibir: (1) equilibrio representativo entre clases benignas y maliciosas, (2) cobertura exhaustiva de variantes de ataques contemporáneos, (3) etiquetado preciso con verificación ground truth, y (4) características temporales realistas que reflejen la no-estacionariedad de entornos operativos reales. La violación de estos principios conduce a artefactos de evaluación como el sobreajuste a características espurias, degradación de la generalización cruzada, y estimaciones sesgadas del rendimiento operativo [5]. Así, la adopción de conjuntos de referencia estandarizados con protocolos de generación transparentes y métricas de calidad validadas se ha convertido en un requisito metodológico fundamental para la investigación reproducible en este dominio.

C. Aprendizaje Automático en Ciberseguridad

El aprendizaje automático (Machine Learning, ML) constituye una rama de la inteligencia artificial que permite a los sistemas computacionales mejorar su desempeño mediante la experiencia, sin necesidad de programación explícita para cada tarea específica. En el contexto de seguridad de redes, el ML se manifiesta en tres paradigmas principales: el aprendizaje supervisado, donde modelos se entrenan con datos etiquetados para aprender funciones de mapeo entre características y categorías de tráfico, el aprendizaje no supervisado, que identifica patrones intrínsecos y agrupaciones naturales en datos no etiquetados; y el aprendizaje por refuerzo, donde agentes autónomos aprenden políticas óptimas de decisión mediante interacción continua con el entorno de red y retroalimentación basado en recompensas [6].

Las aplicaciones de ML en seguridad de red abarcan un espectro tecnológico amplio y sofisticado. Los sistemas de detección de intrusiones (IDS) modernos emplean algoritmos de clasificación para discriminar entre tráfico benigno y malicioso con base en patrones estadísticos aprendidos. Técnicas de clustering permiten descubrir campañas de ataques coordinadas o variantes de malware previamente desconocidas mediante la detección de agrupaciones anómalas en el espacio de características. Adicionalmente, modelos de regresión se utilizan para predecir tendencias de ataques y estimar niveles de riesgo operacional, mientras que los sistemas de recomendación basados en ML optimizan las políticas de firewall y control de acceso [7]. Estas aplicaciones demuestran la versatilidad del ML para abordar desafíos de seguridad en múltiples capas de la infraestructura de red.

Las ventajas del ML frente a los métodos tradicionales basados en firmas son sustanciales y multidimensionales. Mientras los sistemas basados en firmas requieren actualizaciones manuales constantes y resultan inefectivos contra amenazas zero-day, los modelos de ML pueden generalizar patrones de ataques novedosos a partir de principios estadísticos subyacentes. La capacidad de detección proactiva se ve reforzada por la identificación automática de desviaciones comportamentales sutiles que escapan a las firmas predefinidas. Operacionalmente, los sistemas habilitados con ML exhiben escalabilidad superior al procesar grandes volúmenes de tráfico en tiempo real con mínima intervención humana, reduciendo tanto los costos operativos como la ventana de exposición a amenazas [7]. Esta evolución paradigmática desde la detección reactiva hacia la prevención predictiva representa un avance estratégico en la arquitectura de ciberseguridad moderna.

D. Modelos de Clasificación Multiclase.

La clasificación multiclase constituye un paradigma fundamental en el aprendizaje automático supervisado donde el objetivo es asignar cada instancia de entrada a una entre K categorías mutuamente excluyentes. En el contexto de seguridad de redes, se formaliza el problema mediante el conjunto de entrenamiento $D = \{(x_i, y_i)\}_{i=1}^{N}$, donde $x_i \in \mathbb{R}^d$ representa el vector de características de dimensión d extraídas del tráfico de red y $y_i \in \{1,2,\dots,K\}$ denota la etiqueta de clase correspondiente entre K posibles categorías, el objetivo radica en aprender una función de hipótesis $f: \mathbb{R}^d \to \{1,2,\dots,K\}$

La clasificación multiclase en el contexto del tráfico de red se ve profundamente influenciada por la heterogeneidad y la dinámica inherente de los datos. Cada flujo de red puede presentar comportamientos distintos dependiendo del tipo de aplicación, la hora del día, la topología de la red y la presencia de actividades maliciosas, generando patrones que en muchos casos se superponen entre diferentes categorías de tráfico o ataques. Esta complejidad intrínseca plantea desafíos fundamentales para los modelos de aprendizaje automático. En primer lugar, la discriminación entre clases con patrones solapados es crítica, ya que observaciones con características similares pueden corresponder a categorías distintas, lo que exige que el modelo capture sutilezas en los atributos de los flujos de red. Adicionalmente, el desbalance de clases constituye un reto frecuente en conjuntos de datos de seguridad, donde algunas categorías de ataques están subrepresentadas en comparación con el tráfico benigno, lo que puede inducir sesgos predictivos y afectar la robustez del modelo. Por último, la generalización a datos no vistos es indispensable, dado que los ataques evolucionan constantemente y los patrones de tráfico varían con el tiempo, requiriendo que los modelos sean capaces de aprender interacciones complejas y no lineales entre características para mantener un desempeño consistente y confiable en entornos operativos reales.

Dado este contexto, se seleccionan modelos representativos de dos paradigmas complementarios de aprendizaje automático: la Regresión Logística Multinomial (MLR) y las Redes Neuronales Densas (DNN). La MLR ofrece un enfoque lineal, eficiente y altamente interpretable, proporcionando una base sólida para la clasificación cuando las relaciones entre características y clases pueden aproximarse linealmente. Las DNN, por su parte, permiten aproximar funciones no lineales complejas y capturar interacciones sutiles entre características, ofreciendo flexibilidad para reconocer patrones sofisticados en el tráfico de red. Esta combinación permite evaluar las compensaciones entre simplicidad, eficiencia y capacidad predictiva, constituyendo la base metodológica para la comparación empírica realizada en este estudio.

1) Regresión Logística Multinomial (MLR)

La Regresión Logística Multinomial generaliza la regresión logística binaria a $K > 2$ clases mediante la función softmax, que proyecta los datos de entrada en un espacio de probabilidades, donde cada componente indica la probabilidad asociada a una categoría. Sea $x \in \mathbb{R}^d$ el vector de características de una observación y K el número de clases posibles, la probabilidad de que la observación pertenezca a la clase k se expresa como:

$$P(y = k \mid x) = \frac{\exp\left(w_k{}^{\top} x + b_k\right)}{\sum_{j=1}^{K} \exp\left(w_j{}^{\top} x + b_j\right)} \qquad (1)$$

En esta expresión, $w_k \in \mathbb{R}^n$ es el vector de pesos correspondiente a la clase, y $b_k \in \mathbb{R}$ es el sesgo asociado. El numerador $\exp\left(w_k{}^{\top} x + b_k\right)$ representa una medida no normalizada de afinidad de x hacia la clase k, mientras que el denominador $\sum_{j=1}^{K} \exp\left(w_j{}^{\top} x + b_j\right)$ asegura que la suma de probabilidades sobre todas las clases sea 1. La predicción de clase se realiza mediante:

$$\hat{y} = argmax\, P(y = k \mid x) \qquad (2)$$

El modelo se entrena mediante maximización de la verosimilitud, equivalente a minimizar la pérdida de entropía cruzada:

$$L(W, b) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{i,k} \log P(y_i = k \mid x_i) \qquad (3)$$

Donde $y_{i,k}$ es 1 si la observación i pertenece a la clase k y cero en caso contrario. La optimización se realiza mediante gradiente descendente o variantes como Adam, ajustando iterativamente los pesos w_k y sesgos b_k para minimizar la pérdida. Asimismo, los valores de w_k proporcionan interpretabilidad directa: un coeficiente positivo indica que incrementos en la característica correspondiente aumentan la probabilidad de pertenecer a la clase K, mientras que un valor negativo sugiere asociación con otras clases.

2) Redes Neuronales Densas (DNN)

Las Redes Neuronales Densas son modelos de aprendizaje no lineales conformados por múltiples capas de neuronas completamente conectadas, donde cada neurona de una capa se enlaza con todas las neuronas de la capa siguiente. Este tipo de redes permite transformar las características de entrada en representaciones internas de alta complejidad, capaces de capturar patrones y relaciones no lineales que facilitan la separación de clases complejas en problemas de clasificación multiclase. La estructura de estas redes consiste en una capa de entrada que recibe las características del conjunto de datos, una o varias capas ocultas que realizan transformaciones intermedias mediante funciones de activación no lineales, y una capa de salida que genera las probabilidades o valores finales para cada clase objetivo. La salida de la neurona j en la capa l se define como:

$$h_j{}^{(l)} = \sigma\left(\sum_{i=1}^{n(l-1)} w_{ij}{}^{(l)} h_i{}^{(l-1)} + b_l{}^{(l)}\right) \qquad (4)$$

Donde:

- $h_j{}^{(l)}$ → salida (activación) de la neurona $j$en la capa l.
- $\sigma(\cdot)$ → función de activación (ej. Sigmoide, ReLU, tanh…).
- $w_{ij}{}^{(l)}$ → peso que conecta la neurona i de la capa anterior $(l-1)$ con la neurona j de la capa actual l.
- $b_l{}^{(l)}$ → sesgo de la neurona j en la capa l.
- $h_i{}^{(l-1)}$ → salida de la neurona i en la capa anterior $(l-1)$.

Por otro lado, la capa de salida $h^{(L)}$ se transforma mediante softmax para producir probabilidades multiclase:

$$P(y = k \mid x) = \frac{\exp\left(h_k{}^{(L)}\right)}{\sum_{j=1}^{K} \exp\left(h_j{}^{(L)}\right)} \qquad (5)$$

El entrenamiento se realiza mediante backpropagation, aplicando la regla de la cadena para calcular los gradientes de la función de pérdida de entropía cruzada respecto a todos los pesos y sesgos, y actualizando iterativamente los parámetros con algoritmos de optimización como Adam o el gradiente descendente estocástico (SGD). Esta técnica

permite que los gradientes fluyan desde la capa de salida hacia las capas iniciales, ajustando los pesos de manera que la red aprenda interacciones complejas y no lineales entre las características del tráfico de red.

A pesar de que la DNN es menos interpretable que la MLR, cada capa actúa como un transformador no lineal que extrae representaciones jerárquicas del tráfico de red, combinando atributos temporales, volumétricos y estadísticos para discriminar entre clases, lo que la hace particularmente eficaz en escenarios con patrones solapados o complejos.

3) Comparación entre Algoritmos de Clasificación

La descripción conjunta de la Regresión Logística Multinomial (MLR) y las Redes Neuronales Densas (DNN) permite evidenciar los compromisos fundamentales entre interpretabilidad, complejidad computacional y capacidad predictiva de cada paradigma. La MLR constituye un modelo lineal cuya complejidad computacional es aproximadamente del orden $O(nk)$, con n características y K clases, lo que la hace eficiente en entornos con recursos limitados y adecuada para problemas aproximadamente linealmente separables. Su principal fortaleza radica en la transparencia. Los coeficientes w_k permiten un análisis directo de la contribución de cada característica a la probabilidad de pertenencia a la clase K, lo que facilita tareas de auditoría y análisis forense de tráfico de red.

En contraste, las DNN presentan una complejidad del orden $O(\sum_{l=1}^{L} d_l d_{l-1})$ donde L representa el número de capas en la red neuronal, d_l es número de neuronas en la capa l, y d_{l-1} el número de neuronas en la capa anterior. Las DNN son capaces de capturar relaciones altamente no lineales mediante múltiples capas que aplican transformaciones no lineales de forma jerárquica. Esta capacidad las hace más adecuadas para escenarios donde los patrones de tráfico presentan interacciones complejas, aunque exige volúmenes sustanciales de datos y recursos computacionales significativos para su entrenamiento. Asimismo, su carácter de "caja negra" limita la interpretabilidad directa, comprometiendo la trazabilidad de las decisiones.

E. Conjunto de Datos CIC-IDS2017

La investigación en detección de intrusiones en redes (IDS) ha dependido históricamente de conjuntos de datos públicos para el entrenamiento, validación y evaluación de modelos de aprendizaje automático. Algunos de los conjuntos de datos históricos más citados incluyen KDD Cup 1999, derivado del conjunto de datos DARPA 1998, que proporcionó un primer estándar para detección de intrusiones, y NSL-KDD, que corrigió problemas de redundancia y desbalance en KDD. Más recientemente, UNSW-NB15 incorporó ataques contemporáneos y tráfico más realista. No obstante, la mayoría de estos conjuntos presentan limitaciones importantes: tráfico sintético o poco representativo, cobertura limitada de ataques conocidos, anonimización de payloads y ausencia de metadatos completos, lo que compromete la confiabilidad y la capacidad de generalización de los modelos de IDS.

En respuesta a estas limitaciones, el Canadian Institute for Cybersecurity (CIC) desarrolló CIC-IDS2017, un conjunto de datos diseñado para reflejar de manera fiel el tráfico de red contemporáneo y los ataques más comunes. Este conjunto contiene tanto tráfico benigno como ataques reales, incluyendo Brute Force FTP y SSH, DoS, DdoS, Heartbleed, Web Attacks, Infiltration y Bots, capturados durante cinco días de operación continua (3–7 de julio de 2017). La generación de tráfico benigno se realizó mediante B-profiles, un enfoque que encapsula los comportamientos abstractos de los usuarios y distribuciones de protocolos mediante técnicas de machine learning y análisis estadístico, como K-Means, Random Forest, SVM y J48. Cada B-profile modela patrones de tráfico como distribuciones de tamaños de paquetes, número de paquetes por flujo, patrones en los payloads, tamaños de payload y distribución temporal de solicitudes para protocolos específicos. Se simularon protocolos HTTPS, HTTP, SMTP, POP3, IMAP, SSH y FTP, observando que la mayoría del tráfico benigno correspondía a HTTP y HTTPS. Este enfoque permitió generar un tráfico de fondo realista, con interacciones complejas tanto entre usuarios como con servicios de red [1]–[8].

La captura de datos se realizó utilizando CICFlowMeter, una herramienta desarrollada por el CIC que permite la generación de flujos de red y etiquetarlos de manera agregada. CICFlowMeter procesa el tráfico de red registrado en archivos PCAP y consolida los paquetes en flujos, los cuales no representan paquetes individuales, sino secuencias de paquetes que comparten atributos comunes y que reflejan la interacción o "conversación" entre dos nodos en la red. Cada flujo se caracteriza mediante información completa de direcciones IP de origen y destino, puertos de comunicación, protocolos de transporte, marcas temporales y etiquetas de ataque, produciendo así un conjunto de datos altamente estructurado de 2,830,743 flujos con 78 características.

Para optimizar el conjunto de datos original para su uso en modelos de aprendizaje automático, se empleó la versión preprocesada "CICIDS2017: Cleaned & Preprocessed" disponible en Kaggle, desarrollada por Eric Anacleto Ribeiro [9]. Esta versión incorpora un conjunto de transformaciones sistemáticas orientadas a mejorar la integridad, la representatividad y la utilidad analítica del dataset original. Los pasos de preprocesamiento incluyen:

- La fusión de archivos CSV, consolidando múltiples archivos que componían el dataset original en un único conjunto unificado, lo que facilita la manipulación y el análisis.
- La eliminación de duplicados, aplicada tanto a filas como a columnas redundantes, asegura la integridad del dataset y evita la distorsión de las métricas de entrenamiento de los modelos.
- El manejo de valores infinitos y faltantes se llevó a cabo reemplazando los infinitos por NaN y eliminando las filas con valores faltantes, las cuales representan menos del 1% del total de registros. Esta decisión minimiza el riesgo de introducir sesgos a través de imputaciones y simplifica la preparación de datos.
- La normalización de nombres de columnas consistió en la eliminación de espacios y la homogeneización de la nomenclatura, garantizando consistencia y evitando errores de interpretación durante el procesamiento automatizado.
- La selección de características se ejecutó en varias etapas: inicialmente se eliminaron columnas con un único valor, ya que carecían de capacidad discriminativa. Posteriormente, se aplicó un análisis de correlación para remover atributos altamente correlacionados ($\rho \geq 0.99$), reduciendo la multicolinealidad. Esta selección se complementó con pruebas estadísticas de relevancia (H-Statistics) y técnicas basadas en Random Forest, asegurando que únicamente se conservaran características informativas y estadísticamente significativas.
- La transformación de la columna objetivo consistió en renombrar "Label" como "Attack Type" y agrupar etiquetas de ataques similares (por ejemplo, DoS Hulk y DoS GoldenEye como "DoS"), mientras que ataques poco frecuentes como Heartbleed e Infiltration fueron eliminados para prevenir sobreajuste y mejorar la generalización de los modelos.

El conjunto de datos resultante comprende 52 características de ingeniería de tráfico que encapsulan propiedades fundamentales del comportamiento de red. Estas características se categorizan en métricas volumétricas (ej. Total Fwd Packets, Total Length of Fwd Packets),

temporales (ej. Flow Duration, Flow IAT Mean, Fwd IAT Total), estadísticas de distribución de paquetes (ej. Fwd Packet Length Std, Packet Length Variance), indicadores de flags TCP (ej. FIN Flag Count, ACK Flag Count), patrones de actividad e inactividad (ej. Active Mean, Idle Max), parámetros de ventana TCP (ej. Init_Win_bytes_forward), y métricas de flujo agregado (ej. Flow Bytes/s, Flow Packets/s). Para facilitar la reproducibilidad y el análisis detallado, el **Apéndice A** proporciona un esquema completo con la descripción formal de cada característica, su tipo de dato y rango típico de valores.

La distribución de clases tras el preprocesamiento se presenta en la **Tabla I**:

TABLA I
DISTRIBUCIÓN DE CLASES CIC-IDS2017 PREPROCESADO

Tipo de Tráfico	Instancias
Normal Traffic	2,095,057
DoS	193,745
DDoS	128,014
Port Scanning	90,694
Brute Force	9,150
Web Attacks	2,143
Bots	1,948

El preprocesamiento implementado en el conjunto de datos garantiza que este sea representativo, limpio y adecuado para el entrenamiento y evaluación de modelos multiclase, reduciendo sesgos, evitando sobreajuste y asegurando reproducibilidad.

F. Métricas de Evaluación

La evaluación cuantitativa de sistemas de detección de intrusiones basados en aprendizaje automático requiere métricas capaces de reflejar de manera integral el desempeño en entornos caracterizados por un marcado desbalance de clases. Dado que las clases de interés, correspondientes a distintos tipos de ataques, suelen representar únicamente una fracción reducida del tráfico total, métricas convencionales como la exactitud (accuracy) resultan insuficientes debido a su sensibilidad al desbalanceo. En este contexto, se recurre a un conjunto de métricas derivadas de la matriz de confusión multiclase, las cuales permiten una caracterización más detallada y matizada del rendimiento del modelo, incluyendo su capacidad de detectar correctamente las clases minoritarias y minimizar falsos positivos y falsos negativos.

Sea un modelo de clasificación multiclase, donde para cada clase k se define:

- $TP_K (True\ Positives)$: instancias correctamente clasificadas como clase k.
- $FP_K (False\ Positives)$: instancias de otras clases incorrectamente clasificadas como clase k.
- $FN_K (False\ Negatives)$: instancias de clase k clasificadas incorrectamente como otra clase.
- $TN_K (True\ Negatives)$: instancias de otras clases correctamente clasificadas como no pertenecientes a k.

La exactitud o accuracy mide la proporción total de instancias correctamente clasificadas sobre el total de instancias.

$$Exactitud = \frac{\sum_{k=1}^{K} TP_K}{\sum_{k=1}^{K} (TP_K + FP_K + FN_K)} \quad (6)$$

La precisión indica la proporción de instancias clasificadas como clase k que realmente pertenecen a esa clase.

$$Precisión_k = \frac{TP_K}{TP_K + FP_K} \quad (7)$$

El recuerdo o recall refleja la capacidad del modelo para identificar correctamente todas las instancias de la clase k:

$$Recuerdo_k = \frac{TP_K}{TP_K + FN_K} \quad (8)$$

La puntuación F1 o F1-score, definida como el promedio armónico entre precisión y recuerdo, proporciona un balance que es especialmente útil en escenarios con clases desbalanceadas:

$$F1_k = 2 \cdot \frac{Precisión_k \cdot Recuerdo_k}{Precisión_k + Recuerdo_k} \quad (9)$$

Adicionalmente, la curva ROC (Receiver Operating Characteristic) y su área bajo la curva (AUC-ROC) permiten evaluar la capacidad de discriminación del modelo considerando todos los posibles umbrales de decisión, a diferencia de métricas como precisión, recuerdo o la puntuación F1, que se calculan para un umbral específico.

Para cada clase k, la tasa de falsos positivos (FPR_k) y la tasa de verdaderos positivos (TPR_k) se definen como:

$$FPR_k = \frac{FP_K}{FP_K + TN_K}, \qquad TPR_k = \frac{TP_K}{TP_K + FN_K} \quad (10)$$

La curva ROC representa gráficamente la relación entre TPR_k y FPR_k para distintos umbrales de decisión, proporcionando una visualización completa de la capacidad del modelo para separar clases.

El AUC (Area Under the Curve) agrega esta información en un único valor, donde valores cercanos a 1 indican una excelente discriminación entre clases, mientras que valores próximos a 0.5 reflejan un desempeño cercano al azar. Esta métrica es especialmente útil en escenarios con clases desbalanceadas, porque considera simultáneamente la capacidad del modelo para detectar correctamente los ataques y para evitar falsos positivos en todos los posibles umbrales de decisión. Esto permite elegir de manera informada el umbral óptimo para su implementación en entornos operativos.

G. Revisión de Literatura

La revisión de literatura constituye un componente fundamental en la investigación de detección de intrusiones en redes, ya que permite identificar enfoques consolidados, limitaciones de metodologías existentes y métricas de desempeño críticas para evaluar modelos de aprendizaje automático. Diversos estudios recientes han abordado la detección de intrusiones utilizando el conjunto de datos CICIDS-2017, considerando tanto algoritmos supervisados como no supervisados y evaluando su eficacia mediante métricas cuantitativas de precisión, recuerdo, F1-score y exactitud general.

Jairu y Pankaj [10] realizaron un análisis exhaustivo empleando algoritmos de clasificación supervisada como Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Naïve Bayes, Decision Tree y Random Forest sobre subconjuntos específicos del dataset CICIDS-2017. Sus resultados indican que Random Forest alcanza la mayor exactitud global, reportando hasta 99.93% con un

conjunto de 14 características seleccionadas mediante el coeficiente de correlación de Pearson. En escenarios específicos, como ataques DDoS registrados el 7 de julio de 2017, Random Forest logra una precisión del 100% en la clase BENIGN y 99% en DDoS, superando a otros clasificadores, mientras que Naïve Bayes mostró rendimientos significativamente inferiores (por ejemplo, precisión de 43% en la clase DDoS). Este estudio subraya la importancia de la selección de características representativas y la variabilidad de desempeño según el tipo de ataque y el subconjunto de datos utilizado.

En un enfoque más reciente, Xu y Liu [11] evaluaron tanto modelos supervisados (MLP y CNN) como no supervisados (One-Class SVM y Local Outlier Factor) en el mismo dataset, distinguiendo entre ataques conocidos y previamente no observados. Los modelos supervisados MLP y CNN alcanzaron una precisión de 98.94% y 93.16% respectivamente sobre ataques conocidos, pero sufrieron un colapso en recall frente a ataques desconocidos, descendiendo hasta 17.5% y 19.5% para MLP y CNN respectivamente. Por otro lado, LOF, un modelo basado en densidad, logró un recall de 83.7% en ataques desconocidos, aunque con una precisión moderada de 57.46%, y OCSVM presentó un balance más robusto, alcanzando una exactitud de 79.19% y F1-score de 0.7575 en escenarios de ataques no vistos. Estos resultados destacan la necesidad de considerar la generalización a amenazas novedosas, un aspecto crítico en entornos de red dinámicos.

Finalmente, Panwar et al. [12] aplicaron algoritmos clásicos de WEKA, específicamente Naïve Bayes, Decision Tree y J48, evaluando su desempeño con y sin discretización y mediante técnicas de selección de características como Classifier Subset Evaluator (CSE). Se observó que la discretización supervisada combinada con selección de características mejora de manera consistente la exactitud, recall y F1-score mientras reduce el tiempo de entrenamiento. Por ejemplo, el J48 con discretización alcanzó los mejores resultados para ataques de Brute Force, DoS y DDoS, evidenciando una tendencia similar a los hallazgos de Jairu y Pankaj, donde la optimización de características incrementa la efectividad del clasificador.

En conjunto, estos estudios cuantifican de manera precisa cómo diferentes algoritmos y estrategias de preprocesamiento impactan la detección de intrusiones. La revisión de literatura no solo permite identificar los métodos más efectivos para escenarios específicos de ataques, sino que también ofrece directrices críticas para diseñar sistemas de detección robustos que puedan adaptarse a ataques conocidos y desconocidos, optimizando la selección de características y el tipo de modelo de aprendizaje automático.

III. Metodología

Según lo detallado en la sección de Marco Teórico, el conjunto de datos CIC-IDS2017 constituye un punto de referencia en investigación de detección de intrusiones, caracterizado por su representatividad de tráfico de red contemporáneo que incluye flujos benignos y múltiples categorías de ataques. Para este estudio, se emplea la versión preprocesada y curada disponible en Kaggle [9], la cual ha sido sometida a un pipeline de limpieza que garantiza integridad estadística, consistencia en las instancias y uniformidad en las características, optimizando así la aplicabilidad de algoritmos de aprendizaje automático y asegurando la replicabilidad de los resultados.

La metodología adoptada se organiza en etapas secuenciales: preprocesamiento de datos, selección de características, partición del conjunto de datos, entrenamiento de modelos y evaluación de desempeño. Cada etapa sigue protocolos estandarizados de procesamiento y validación, asegurando la minimización del sesgo de muestreo y la obtención de métricas de desempeño confiables, en línea con las mejores prácticas reportadas en la literatura.

A. Preparación de los Datos

El análisis inicial reveló un notable desbalance entre las clases, siendo la categoría correspondiente a tráfico normal la de mayor frecuencia, con más de 2 millones de muestras, mientras que clases como Bots presentaban menos de 2,000 instancias. Para abordar esta disparidad y favorecer la generalización de los modelos, se aplicó un muestreo aleatorio de cada categoría, ajustando todas las clases al tamaño de la categoría con menor número de muestras. A continuación, se muestra la distribución resultante de muestras por clase tras este proceso de balanceo:

TABLA II
DISTRIBUCIÓN DE CLASES CIC-IDS2017 BALANCEADAS

Clase	Número de Muestras
Bots	1,948
Brute Force	1,948
DDoS	1,948
DoS	1,948
Normal Traffic	1,948
Port Scanning	1,948
Web Attacks	1,948

A pesar de que el conjunto de datos de Kaggle ya fue preprocesado, incluyendo la remoción de atributos altamente correlacionados ($\rho \geq 0.99$) mediante análisis estadístico y técnicas basadas en Random Forest, se consideró pertinente realizar un análisis adicional. Con los datos balanceados, se construyó una matriz de correlación absoluta entre las 52 variables numéricas para examinar de manera cuantitativa las relaciones lineales existentes entre los atributos. Esta matriz permitió identificar grupos de variables altamente correlacionadas, evidenciando redundancias potenciales y señalando oportunidades de reducción de dimensionalidad.

Para abordar la reducción de dimensionalidad, se empleó Análisis de Componentes Principales (PCA). Esta técnica transforma el conjunto original de 52 variables correlacionadas en componentes linealmente independientes, priorizando aquellos que capturan la mayor varianza del conjunto de datos. Con el fin de sustentar empíricamente la selección, se construyó una gráfica de varianza explicada por componente, la cual permite visualizar el aporte relativo de cada uno y facilita la decisión del número óptimo de componentes a retener. A partir de este análisis se seleccionaron los 10 primeros componentes principales, los cuales explican aproximadamente el 85 % de la varianza total. La elección de este umbral responde a un criterio ampliamente adoptado en la literatura: retener la mayor parte de la información significativa mientras se reduce sustancialmente la complejidad del modelo, evitando redundancias y mitigando el riesgo de sobreajuste. Esta transformación permitió preservar la representatividad de los datos y optimizar el desempeño de los modelos en la etapa de clasificación.

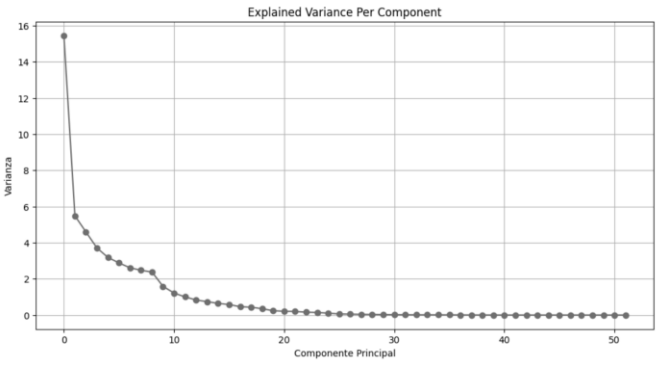

Fig. 1. Varianza explicada por cada componente principal. La gráfica ilustra la contribución relativa de cada componente a la varianza total del conjunto de datos.

Posteriormente, los datos fueron escalados utilizando normalización estándar con apoyo de la herramienta StandardScaler de Scikit-Learn, garantizando que todas las variables presentaran media cero y desviación estándar unitaria. Este procedimiento es crítico para modelos sensibles a la magnitud de los atributos, asegurando una convergencia estable y eficiente de los algoritmos de aprendizaje automático.

Finalmente, se procedió a la separación de los datos en conjuntos de entrenamiento y prueba, reservando un 20 % de las muestras para evaluación y manteniendo la proporción de clases mediante estratificación. Los conjuntos resultantes mostraron un balance consistente, con 10,908 muestras en entrenamiento y 2,728 en prueba, asegurando una representación equitativa de las siete clases en ambas particiones. Adicionalmente, se generaron dos representaciones de los vectores de etiquetas para las muestras: codificación categórica y codificación one-hot. La codificación categórica asigna un valor entero único a cada clase, facilitando la manipulación de las etiquetas en modelos de aprendizaje automático que requieren entradas discretas. Por otro lado, la codificación one-hot transforma cada etiqueta en un vector binario, donde cada dimensión corresponde a una clase y únicamente la posición asociada a la clase verdadera se establece en uno, mientras que las demás se establecen en cero. Esta representación es esencial para modelos de redes neuronales y métodos de aprendizaje profundo que requieren que las salidas se interpreten como distribuciones de probabilidad mediante funciones de activación como softmax. La generación de ambas codificaciones asegura la compatibilidad y flexibilidad con múltiples paradigmas de modelado.

B. Modelado

Con el objetivo de abordar el problema de clasificación multiclase planteado en el dominio de detección de intrusiones, se implementaron dos enfoques complementarios: algoritmos de aprendizaje automático supervisado y modelos de aprendizaje profundo. En particular, se seleccionaron la Regresión Logística y las Redes Neuronales Densas como representantes de cada paradigma. Esta elección responde a la necesidad de evaluar tanto modelos lineales, de alta interpretabilidad y baja complejidad, como arquitecturas no lineales capaces de capturar patrones más complejos en los datos.

1) Regresión Logística

La Regresión Logística se implementó empleando la clase LogisticRegression de Scikit-Learn configurada en modo multinomial para modelar simultáneamente todas las clases mediante la activación softmax. La implementación práctica se realizó mediante un Pipeline que integra, de forma encadenada y reproducible, el escalado de las características y el estimador. Este diseño asegura que las transformaciones (escalado y proyección PCA) se ajusten exclusivamente sobre los subconjuntos de entrenamiento durante la validación cruzada, evitando así fuga de información y sesgos optimistas en la estimación del rendimiento.

La búsqueda de hiperparámetros se llevó a cabo con GridSearchCV en una malla diseñada para explorar de forma sistemática los grados de regularización, el método numérico de optimización, la tolerancia a la convergencia y el tratamiento del balance de clases. La elección de los valores evaluados obedece a criterios metodológicos: la constante de regularización C se muestreó en escala logarítmica para cubrir órdenes de magnitud (desde regularización fuerte hasta débil), lo cual permite evaluar explícitamente el compromiso entre sesgo y varianza. Por otro lado, la penalización $L2$ se priorizó por su estabilidad en problemas multinomiales y por su efecto amortiguador frente a multicolinealidad residual. Asimismo, los solvers considerados (newton-cg y saga) representan alternativas complementarias. Por un lado, newton-cg aporta convergencia estable y eficiente para problemas de tamaño moderado mediante métodos de segundo orden, mientras que saga ofrece escalabilidad y robustez en escenarios con grandes volúmenes o actualizaciones estocásticas. Finalmente, los distintos límites de iteración (max_iter) fueron incluidos para garantizar la convergencia numérica en configuraciones con baja regularización o con solvers que requieren más pasos de optimización. Se evaluaron además las opciones de ponderación de clases (None vs. balanced) a fin de cuantificar la sensibilidad del estimador frente a desbalances residuales, aun cuando el conjunto fue balanceado por muestreo en fases previas.

Como criterio de selección se adoptó el F1-score ponderado durante la validación cruzada estratificada con $K = 5$. La estratificación preserva la proporción de clases en cada partición, condición imprescindible en problemas multiclase con categorías minoritarias; la elección de $K = 5$ responde a un compromiso entre estabilidad de la estimación y coste computacional. El uso del F1-score ponderado como objetivo de optimización obedece a la necesidad de equilibrar precisión y exhaustividad a través de clases con distinto soporte, evitando la miopía que introduce la métrica de exactitud en escenarios desbalanceados.

Desde la perspectiva de reproducibilidad y eficiencia computacional, todas las búsquedas se ejecutaron con semilla fija (random_state) y paralelización (n_jobs=-1) para asegurar trazabilidad y aprovechar recursos disponibles.

La siguiente tabla resume el espacio de búsqueda de hiperparámetros evaluado mediante GridSearchCV. En total, se exploraron 60 configuraciones distintas:

TABLA III
HIPERPARÁMETROS EXPLORADOS EN LA OPTIMIZACIÓN DEL MODELO DE REGRESIÓN LOGÍSTICA

Parámetro	Valores considerados
C	{0.01, 0.1, 1, 10, 100}
Penalización	l2
Solver	{newton-cg, saga}
Max_iter	{100, 500, 1000}
Class_weight	{None, balanced}

2) Redes Neuronales Densas

La implementación de las Redes Neuronales Densas se realizó utilizando la biblioteca Keras sobre TensorFlow, seleccionada por su flexibilidad, eficiencia en operaciones matriciales y compatibilidad con GPU, lo que permite entrenamientos más rápidos en grandes volúmenes de datos. Cada red fue definida como un modelo secuencial (Sequential), en el cual las capas se agregan de manera lineal desde la capa de entrada hasta la capa de salida. Se definieron tres arquitecturas diferenciadas en profundidad y número de neuronas por capa, con el fin de evaluar el impacto de la complejidad de la red en el rendimiento predictivo.

Para todas las configuraciones experimentales se estableció un número de 10 neuronas de entrada, derivadas del análisis de componentes principales (PCA), que garantiza la preservación de la variabilidad significativa de los datos mientras se reduce la dimensionalidad y la correlación entre características. La capa de salida se definió con 7 neuronas, correspondientes al número de clases a predecir, empleando la función de activación Softmax para transformar la salida en probabilidades normalizadas. Las capas intermedias utilizaron la función de activación ReLU, seleccionada por su capacidad para introducir no linealidad, reducir el riesgo de desvanecimiento del gradiente y acelerar la convergencia durante el entrenamiento.

El optimizador seleccionado fue Adam, con una tasa de aprendizaje de 0.01, por su robustez frente a variaciones en la escala de las características y su capacidad para adaptarse de manera dinámica al gradiente durante la optimización. La función de pérdida empleada fue

categorical_crossentropy, adecuada para problemas de clasificación multiclase y consistente con la codificación one-hot de las etiquetas.

El tamaño del batch se ajustó entre 500 y 1000, evaluando su influencia sobre la estabilidad de la convergencia y la eficiencia del entrenamiento. Batches menores permiten una actualización más frecuente de los pesos, acelerando el aprendizaje inicial, pero aumentando la variabilidad del gradiente, mientras que batches mayores favorecen la estabilidad de la optimización y reducen el tiempo total de entrenamiento para arquitecturas más profundas. El número de épocas se estableció en 30 y 50, dependiendo de la profundidad de la red, asegurando una convergencia suficiente sin incurrir en sobreajuste significativo.

Las arquitecturas evaluadas fueron tres. La primera red consistió en dos capas ocultas con 20 y 10 neuronas respectivamente, diseñada para balancear complejidad y generalización, utilizando un batch de 500 y 50 épocas. La segunda red, más profunda, comprendió cuatro capas ocultas con 20, 30, 20 y 10 neuronas, evaluando la capacidad de representación de redes más complejas, con batch de 1000 y 30 épocas para asegurar estabilidad. La tercera red, de menor complejidad, consistió en una sola capa oculta de 20 neuronas, con batch de 1000 y 30 épocas, utilizada como referencia para medir el impacto del aumento de profundidad en la performance del modelo.

A continuación, se presentan de manera sintética los parámetros de entrenamiento y las arquitecturas:

TABLA IV
ARQUITECTURAS Y PARÁMETROS DE ENTRENAMIENTO DE LAS REDES NEURONALES DENSAS EVALUADAS

Modelo	Capas Ocultas	Batch Size	Épocas	Descripción
RN1	20 – 10	500	50	Arquitectura intermedia que optimiza el equilibrio entre complejidad y generalización. Batch menor permite actualizaciones más frecuentes de los pesos, acelerando el aprendizaje inicial sin comprometer la estabilidad del entrenamiento.
RN2	20 – 30 – 20 – 10	1000	30	Arquitectura más profunda con mayor capacidad de representación, utilizada para capturar patrones complejos no lineales. Batch grande asegura estabilidad de gradiente y reducción del tiempo de entrenamiento.
RN3	20	1000	30	Arquitectura simplificada empleada como control experimental, permitiendo evaluar la contribución de la profundidad adicional en la capacidad predictiva del modelo.

C. Implementación

1) Ambiente de implementación

Todos los experimentos fueron ejecutados en un equipo de cómputo personal Asus ROG Strix G16, equipado con una unidad de procesamiento gráfico (GPU) NVIDIA RTX y un procesador central (CPU) de 32 núcleos, acompañado de 16 GiB de memoria RAM. El sistema operativo utilizado fue Debian GNU/Linux 12. El desarrollo y la ejecución de los modelos se llevaron a cabo con Python 3.11.13, empleando Scikit-learn 1.7.2 para la implementación de algoritmos de regresión logística y preprocesamiento, y TensorFlow 2.20.0 junto con Keras 3.11.3 para la construcción y evaluación de la red neuronal densa. La aceleración de cómputo en GPU se habilitó mediante Nvidia cuDNN-cu12 versión 9.13.0.50. Para el manejo de datos y cálculos matriciales se utilizaron Pandas 2.3.2 y NumPy 2.3.3, mientras que la visualización de resultados se realizó con Matplotlib 3.10.6 y Plotly 6.3.0.

2) Flujo de Predicción en Entorno Reproducible

El flujo de predicción implementado tiene como objetivo transferir los modelos entrenados desde un entorno experimental hacia un entorno confiable y reproducible, permitiendo su aplicación directa para la clasificación de tráfico de red. Este proceso integra la serialización de modelos y objetos de transformación, la normalización y reducción de dimensionalidad de los datos, la ejecución de predicciones y la visualización de resultados, asegurando consistencia con los parámetros aprendidos durante el entrenamiento.

El proceso se inicia con la serialización de los modelos y objetos de preprocesamiento mediante la librería pickle de Python. En esta fase se almacenaron la regresión logística optimizada (regLogModel), la red neuronal densa entrenada (rnModel), el objeto de análisis de componentes principales (pcaObject), el escalador estándar (scaler) y la lista de categorías posibles (categories). La persistencia de estos objetos en archivos binarios permite recuperar de manera exacta los pesos, transformaciones y parámetros definidos durante el entrenamiento, garantizando reproducibilidad y estabilidad de los resultados.

Durante la predicción, los objetos serializados se cargan con pickle.load, habilitando su uso inmediato sin necesidad de reentrenamiento. La entrada de datos consiste en 52 variables descriptivas del tráfico de red, que pueden suministrarse manualmente mediante un formulario interactivo o mediante la carga de archivos de texto .txt, los cuales son validados para asegurar que contengan todos los valores requeridos y sean numéricos.

Los vectores de entrada son sometidos a preprocesamiento, consistente en normalización y proyección sobre los componentes principales determinados por PCA, respetando la configuración utilizada en el entrenamiento. Posteriormente, los modelos aplican la predicción de manera secuencial sobre los datos transformados, retornando probabilidades de pertenencia a cada clase para ambos modelos. Los resultados se visualizan mediante gráficos de línea, donde las categorías se representan en el eje horizontal y las probabilidades en el eje vertical, proporcionando una interpretación cuantitativa clara de la confianza de cada modelo (Figura 2).

La interacción con el sistema se facilita mediante una interfaz gráfica basada en Tkinter, diseñada para usuarios sin conocimientos de programación. La interfaz incluye un formulario desplazable para el ingreso manual de las 52 variables, así como la opción de cargar archivos de texto validados. La ejecución de predicciones se realiza mediante botones dedicados que invocan la función de predicción y generan automáticamente los gráficos de salida. La interfaz incorpora validación de datos y manejo de errores, asegurando la robustez y usabilidad del sistema en entornos operativos (Figura 3).

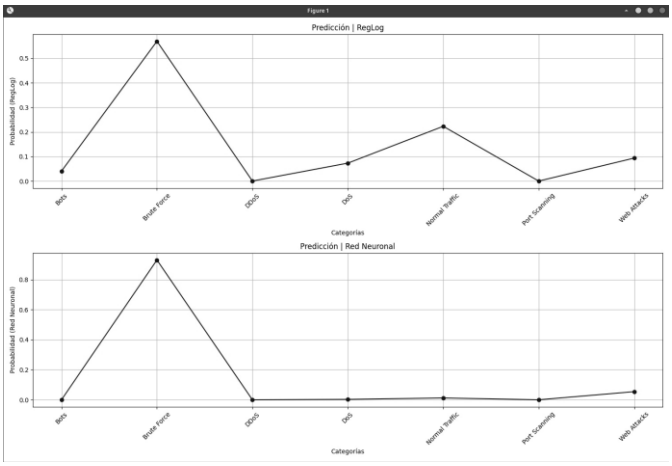

Fig. 2. Gráfico de probabilidades de clasificación por cada modelo. La gráfica presenta las probabilidades de pertenencia de cada clase según los modelos entrenados, permitiendo interpretar cuantitativamente la confianza de las predicciones.

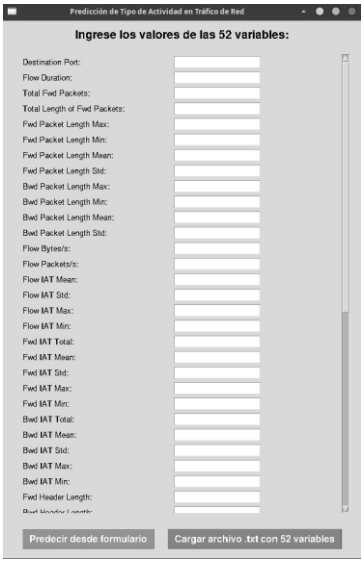

Fig. 3 Interfaz gráfica para ingreso de variables y ejecución de predicciones. La figura muestra el formulario desplazable para ingresar las 52 variables y los botones dedicados para ejecutar las predicciones.

IV. Resultados

A. Resultados de Regresión Logística

El modelo de Regresión Logística fue optimizado mediante búsqueda de hiperparámetros, alcanzando los siguientes valores óptimos: un parámetro de regularización C de 100, un peso de clases balanceado para contrarrestar el desbalance en el conjunto de datos, un número máximo de iteraciones de 100, la utilización de la penalización L2 y el solucionador de tipo Newton-CG. La validación cruzada de 5 folds arrojó un score promedio de 0.908, indicando una capacidad de generalización robusta sobre distintas particiones del conjunto de datos.

La **Tabla V** presenta las métricas de evaluación obtenidas sobre el conjunto de prueba, incluyendo precision, recall, F1-score, soporte absoluto y relativo. Se observa un desempeño altamente satisfactorio para las clases DDoS y Port Scanning, con F1-scores superiores a 0.98, mientras que la clase Normal Traffic muestra un desempeño inferior, con recall de 0.664 y F1-score de 0.734. Esto evidencia que el modelo lineal enfrenta dificultades para separar tráfico legítimo de ataques con patrones menos evidentes.

TABLA V
Métricas de evaluación detalladas para Regresión Logística

Clase	Precision	Recall	F1-score	Support
Bots	0.849	0.897	0.872	389
Brute Force	0.840	0.985	0.907	389
DDoS	0.961	1.000	0.980	390
DoS	0.965	0.921	0.942	390
Normal Traffic	0.820	0.664	0.734	390
Port Scanning	0.980	0.997	0.989	390
Web Attacks	0.946	0.897	0.921	390
Accuracy	0.909			
Accuracy		0.909		
Macro F1-score		0.906		
Weighted F1-score		0.906		

El análisis de las métricas revela que la Regresión Logística logra un desempeño equilibrado en la mayoría de las clases, pero la menor capacidad de discriminación se observa en la clase Normal Traffic, indicando que los patrones de tráfico legítimo presentan una superposición parcial con ataques menos intensivos. Esto se refleja en un F1-score macro de 0.906, ligeramente inferior al weighted F1 (0.909), lo que confirma que la contribución de clases más difíciles reduce la eficiencia promedio global.

La Figura 4 muestra la matriz de confusión correspondiente al conjunto de prueba. Se evidencia que las principales confusiones ocurren entre Normal Traffic y Bots, así como entre Normal Traffic y Brute Force. Adicionalmente, se observa que una fracción de instancias de Normal Traffic son clasificadas erróneamente como Web Attacks, y que Web Attacks presentan también confusiones hacia la clase Brute Force. Estos patrones sugieren que el modelo enfrenta dificultades para discriminar tráfico legítimo de ataques de fuerza bruta y de ciertos ataques web. Una explicación plausible es que Bots y Brute Force comparten volúmenes de tráfico relativamente bajos, distribuciones temporales que no difieren drásticamente de conexiones legítimas y secuencias repetitivas que pueden asemejarse a accesos normales automatizados. De manera similar, los ataques web tienden a imitar solicitudes HTTP válidas, lo que facilita su confusión con tráfico normal y, en menor medida, con intentos de fuerza bruta que también se apoyan en múltiples solicitudes válidas pero repetitivas. En contraste, las clases con huellas más claras, como DDoS y Port Scanning, se caracterizan por tasas de paquetes anómalamente altas y patrones estructurados que permiten una separación mucho más nítida, obteniendo resultados de clasificación casi perfectos.

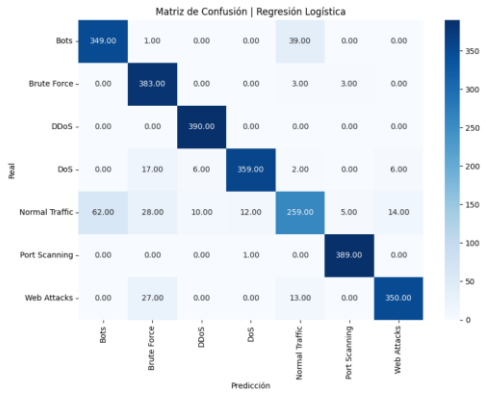

Fig. 4 Matriz de confusión para Regresión Logística. La figura ilustra las predicciones frente a las etiquetas reales, destacando las principales confusiones entre clases.

Desde un enfoque estadístico, la estabilidad del modelo se observa mediante los scores consistentes de validación cruzada (0.908 ± 0.012), indicando que la Regresión Logística generaliza adecuadamente a diferentes particiones de datos, aunque su naturaleza lineal limita la

separación de clases complejas o con solapamiento de características. En particular, los valores de recall inferiores para Normal Traffic sugieren que modelos más flexibles, capaces de capturar relaciones no lineales, podrían mejorar la discriminación en escenarios de ataques sutiles o tráfico legítimo con patrones mixtos.

B. Resultados de Redes Neuronales Densas

Con el propósito de evaluar la capacidad de las redes neuronales densas en la clasificación de tráfico de red, se diseñaron y entrenaron tres arquitecturas con distinta profundidad y número de neuronas (RN1, RN2 y RN3). El entrenamiento se realizó en condiciones homogéneas, utilizando la función de pérdida categorical cross-entropy, el optimizador Adam, y aplicando un particionamiento estratificado de los datos para preservar la proporción de clases en entrenamiento y prueba.

1) RN1: Dos capas densas (20 y 10 neuronas)

El modelo RN1, compuesto por dos capas ocultas con 20 y 10 neuronas respectivamente, se entrenó durante 50 épocas con un tamaño de lote de 500 y una tasa de aprendizaje inicial de 0.01. La **Tabla VI** resume su desempeño sobre el conjunto de prueba.

TABLA VI
MÉTRICAS DE EVALUACIÓN PARA RN1

Clase	Precision	Recall	F1-score	Support
Bots	0.969	0.967	0.968	389
Brute Force	0.959	0.972	0.966	389
DDoS	0.992	0.997	0.995	390
DoS	0.974	0.967	0.970	390
Normal Traffic	0.944	0.913	0.928	390
Port Scanning	0.975	0.997	0.986	390
Web Attacks	0.959	0.962	0.960	390
Accuracy			0.968	
Macro F1-score			0.968	
Weighted F1-score			0.968	

El modelo logró un accuracy de 0.968 y un macro F1-score de 0.968, constituyéndose en la mejor arquitectura entre las evaluadas. La robustez del desempeño en casi todas las clases indica que RN1 logra un equilibrio óptimo entre complejidad y capacidad de generalización. Las dificultades principales se observan en Normal Traffic, donde el F1-score es ligeramente menor, lo que refleja que aún existen confusiones con clases cercanas en términos de patrones de comportamiento como Bots y Brute Force.

2) RN2: Cuatro capas densas (20, 30, 20 y 10 neuronas)

La arquitectura RN2 incrementó la profundidad, incorporando cuatro capas ocultas de 20, 30, 20 y 10 neuronas. El entrenamiento se realizó durante 30 épocas con un tamaño de lote de 1000. La **Tabla VII** presenta los resultados.

TABLA VII
MÉTRICAS DE EVALUACIÓN PARA RN2

Clase	Precision	Recall	F1-score	Support
Bots	0.939	0.995	0.966	389
Brute Force	0.943	0.979	0.960	389
DDoS	0.985	0.992	0.989	390
DoS	0.944	0.946	0.945	390
Normal Traffic	0.960	0.862	0.908	390
Port Scanning	0.982	0.997	0.990	390
Web Attacks	0.979	0.959	0.969	390
Accuracy			0.962	
Macro F1-score			0.961	
Weighted F1-score			0.961	

RN2 alcanza una precisión de 0.962, un F1 macro de 0.961 y un F1 ponderado de 0.961. Resulta relevante observar que la alta precisión obtenida para la clase Normal Traffic (0.960), combinada con un recall relativamente bajo (0.862), indica que el modelo realiza una clasificación conservadora para esta categoría, etiquetando como Normal Traffic únicamente cuando existe alta certeza y generando así un número notable de falsos negativos. Por otro lado, la clase DoS evidencia una disminución relativa en su rendimiento (F1 ≈ 0.945) en comparación con RN1. Estos patrones sugieren que la mayor profundidad de la red y el uso de un batch de gran tamaño (1000) han modificado la dinámica de optimización: la menor cantidad de actualizaciones por época asociada al batch grande, junto con la mayor capacidad de representación de la red, podrían haber conducido a una configuración de parámetros con propiedades de generalización subóptimas para ciertas regiones del espacio de características. En términos prácticos, RN2 presenta un comportamiento más conservador en algunas clases y un recall menos equilibrado.

3) RN3: Una capa densa (20 neuronas)

RN3 fue entrenada durante 30 épocas con batch size 1000. La Tabla 4 contiene las métricas suministradas para esta arquitectura.

TABLA VIII
MÉTRICAS DE EVALUACIÓN PARA RN3

Clase	Precision	Recall	F1-score	Support
Bots	0.960	0.985	0.972	389
Brute Force	0.903	0.979	0.940	389
DDoS	0.963	1.000	0.981	390
DoS	0.975	0.905	0.939	390
Normal Traffic	0.924	0.846	0.884	390
Port Scanning	0.977	0.995	0.986	390
Web Attacks	0.953	0.944	0.948	390
Accuracy			0.951	
Macro F1-score			0.950	
Weighted F1-score			0.950	

RN3 alcanza una precisión de 0.9510 y un F1 macro de 0.950. El patrón más relevante corresponde a la baja precisión observada para la clase Brute Force (0.903) junto con un recall muy alto (0.979), lo que refleja un elevado número de falsos positivos asignados a esta categoría, indicando que RN3 tiende a sobreasignar la etiqueta Brute Force cuando la representación de las instancias es insuficiente para discriminar correctamente otras clases. Asimismo, el F1 reducido para Normal Traffic (0.884) confirma la menor capacidad discriminativa de esta arquitectura más simple.

4) Comparación de las Arquitecturas de Red

Considerando de manera integral todas las métricas de desempeño obtenidas, se evidencia que la arquitectura RN1 alcanza el rendimiento óptimo entre las tres redes evaluadas. La mejora de RN1 respecto a RN2 en macro F1 es de 0.0066 (0.66 puntos porcentuales) y respecto a RN3 de 0.0176 (1.76 puntos porcentuales). Aunque las diferencias parecen

pequeñas, son relevantes en aplicaciones de seguridad, donde incluso fracciones mínimas de errores pueden tener impacto operativo.

Estas diferencias pueden explicarse por varios factores metodológicos. RN1 equilibra capacidad de representación y control de varianza: tiene complejidad suficiente para modelar relaciones no lineales sin inducir sobreajuste, a diferencia de RN2, cuya mayor profundidad puede generar variabilidad en la generalización para clases parcialmente solapadas como Normal Traffic y DoS. Además, RN1 se benefició de un tamaño de batch menor (500) y más épocas (50), lo que permitió actualizaciones más frecuentes y convergencia más precisa, mientras que RN2 y RN3, con batch 1000 y 30 épocas, muestran una convergencia más "conservadora", afectando recall en ciertas clases.

En cuanto al comportamiento por clase, todas las arquitecturas identifican con alta fiabilidad las clases con características distintivas (DDoS, Port Scanning), mientras que las clases con patrones solapados (Normal Traffic, Bots, Brute Force, Web Attacks) son más sensibles a la arquitectura y la dinámica de entrenamiento, como se refleja en la matriz de confusión.

Para confirmar la capacidad de generalización de RN1, se realizó una validación cruzada K-Fold con $K = 5$, obteniéndose un accuracy promedio de 0.9416 (desviación estándar $\approx$ 0.028) y un macro F1 promedio de 0.940. Por otro lado, el intervalo de confianza (IC) del 95% para el accuracy medio, calculado como [0.907, 0.976], indica que con un 95% de confianza, la media verdadera del desempeño de RN1 sobre todo el conjunto de datos se encuentra dentro de este rango. Esto significa que, aunque el accuracy observado en la partición de prueba específica fue 0.9416, el rendimiento real del modelo podría variar ligeramente dependiendo de la partición de datos utilizada. La estrechez del intervalo sugiere que RN1 mantiene un desempeño consistente y robusto a través de distintas particiones estratificadas del conjunto de datos, confirmando que las métricas obtenidas en el conjunto de prueba reflejan un comportamiento confiable del modelo y no dependen de una división particular de los datos.

La Figura 5 presenta la matriz de confusión correspondiente al modelo RN1 evaluado sobre el conjunto de prueba. Se observa que la diagonal principal muestra un desempeño notablemente sólido para la mayoría de las clases, confirmando la alta precisión del modelo. Las clases DDoS y Port Scanning alcanzan las métricas más elevadas, reflejando su separación clara en el espacio de características y su fácil discriminación por la red. En contraste, la clase Normal Traffic presenta el desempeño relativo más bajo, siendo confundida principalmente con Bots, lo que indica cierto solapamiento en patrones de tráfico legítimo y automatizado. Asimismo, Web Attacks muestra confusiones menores con Brute Force, mientras que Bots es ocasionalmente clasificado como Normal Traffic. El resto de las clases evidencia errores residuales mínimos, consolidando que RN1 logra una clasificación consistente y robusta para la mayoría de los tipos de ataque.

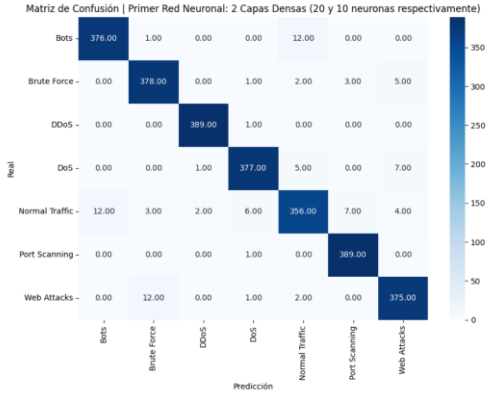

Fig. 5 Matriz de confusión para RN1. La figura muestra las predicciones frente a las etiquetas reales.

C. *Comparación entre Regresión Logística y RN1*

En términos generales, RN1 superó a la Regresión Logística en todas las métricas de desempeño, con un macro F1-score de 0.968 frente a 0.909 de la regresión logística, lo que representa una mejora absoluta de aproximadamente 5.9 puntos porcentuales. Esta diferencia, aunque puede parecer modesta, es significativa en aplicaciones de ciberseguridad, donde incluso pequeñas mejoras en la detección pueden reducir el riesgo operativo y minimizar la ocurrencia de falsos negativos.

En cuanto a la exactitud global, RN1 alcanzó un 0.968 en el conjunto de prueba, comparado con 0.909 de la Regresión Logística, confirmando que la red neuronal logra una clasificación más consistente y generalizable. Desde la perspectiva de la precisión por clase, ambas arquitecturas identifican con alta efectividad ataques con patrones característicos claros, como DDoS y Port Scanning, sin embargo, RN1 presenta una ventaja notable en la clasificación de clases con solapamiento semántico o estadístico, como Normal Traffic, Bots y Web Attacks. Por ejemplo, Normal Traffic obtuvo un recall de 0.913 en RN1 frente a 0.664 en Regresión Logística, reflejando la capacidad de RN1 de modelar relaciones no lineales y detectar patrones complejos que los modelos lineales no capturan completamente.

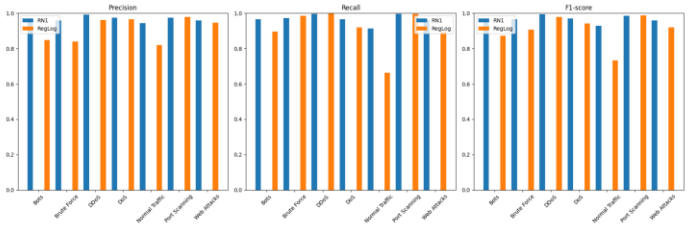

Fig. 6 Comparación de métricas por clase entre Regresión Logística y RN1. La figura grafica las métricas de precisión, recall y F1-score para cada clase, mostrando claramente la superioridad de RN1 en clases con solapamiento parcial y la consistencia de ambos modelos en clases de señal fuerte como DDoS y Port Scanning.

Asimismo, el comportamiento de los modelos frente a las confusiones evidencia diferencias relevantes. En la Regresión Logística, las principales confusiones se producen entre Normal Traffic y Bots, así como entre Normal Traffic y Web Attacks, con una tendencia menor a confundir Brute Force con otras clases. RN1, en contraste, mantiene la diagonal de la matriz de confusión más pronunciada, indicando menor error de clasificación, aunque persiste cierta confusión leve entre Normal Traffic y Bots, y entre Web Attacks y Brute Force. Este patrón refleja que incluso arquitecturas profundas pueden verse afectadas por clases parcialmente solapadas, aunque con una magnitud significativamente inferior a la observada en modelos lineales.

Desde un punto de vista operativo, al evaluar ambos modelos en un entorno productivo, se observó que, aunque generalmente ambos predicen la misma clase final, RN1 asigna mayor probabilidad a la clase predicha y menores probabilidades a las clases alternativas. En cambio, la Regresión Logística distribuye probabilidades relativamente más altas a otras clases, incluso cuando la predicción final coincide, indicando menor confianza en la decisión. Este comportamiento refuerza la ventaja de RN1 en términos de certeza de clasificación y confiabilidad frente a patrones ambiguos de tráfico.

En conclusión, RN1 se confirma como el modelo más adecuado para la clasificación de tráfico de red en este conjunto de datos, debido a su superioridad cuantitativa en métricas globales y por clase, a la reducción de confusiones en clases críticas y parcialmente solapadas, y a la mayor certeza en la asignación de probabilidades. La Regresión Logística, aunque presenta desempeño competitivo en ciertas clases linealmente separables, queda limitada frente a patrones complejos, subrayando la ventaja de modelos de aprendizaje profundo en escenarios de ciberseguridad de alta demanda.

V. CONCLUSIONES

El estudio comparativo entre modelos de clasificación lineales y redes neuronales densas evidencia la superioridad cuantitativa de la arquitectura RN1 sobre la Regresión Logística en la detección de tráfico de red malicioso. RN1 alcanzó un accuracy de 0.968 y un macro F1-score de 0.968, frente a 0.909 y 0.909 para la Regresión Logística, respectivamente, representando una mejora absoluta de aproximadamente 5.9 puntos porcentuales. Estas diferencias, aunque numéricamente moderadas, son estadística y operativamente relevantes, dado que incluso pequeñas mejoras en la clasificación reducen significativamente la ocurrencia de falsos negativos en entornos de ciberseguridad.

El análisis por clase demuestra que RN1 mantiene un desempeño robusto para clases parcialmente solapadas, como Normal Traffic, Bots y Web Attacks, mientras que ambas arquitecturas clasifican con alta precisión las clases de señales claras, como DDoS y Port Scanning. Asimismo, RN1 muestra una distribución de probabilidades más concentrada en la clase predicha, reflejando mayor certeza en la decisión, mientras que la Regresión Logística asigna probabilidades relativamente más altas a clases alternativas, lo que evidencia un nivel de confianza menor en sus predicciones.

La evaluación de generalización mediante validación cruzada K-Fold (k=5) para RN1 confirmó la estabilidad del modelo, con un accuracy promedio de 0.942, macro F1 promedio de 0.940 y un intervalo de confianza al 95% para el accuracy de [0.907, 0.976]. Esto asegura que los resultados observados en el conjunto de prueba no dependen de particiones específicas y reflejan un comportamiento consistente frente a variaciones de los datos.

En síntesis, la arquitectura RN1 de dos capas densas se establece como la opción más eficiente y confiable para la clasificación de tráfico de red en este conjunto de datos, combinando alta precisión, robustez frente a confusiones y capacidad de generalización, lo que resalta la ventaja de modelos de aprendizaje profundo de complejidad moderada frente a modelos lineales en escenarios de ciberseguridad de alta criticidad.

APÉNDICE

Apéndice A – Descripción de características del conjunto de datos

Para facilitar la reproducibilidad y el análisis detallado, este apéndice proporciona un esquema completo con la descripción formal de cada característica, su tipo de dato y rango típico de valores. La tabla completa está disponible en el siguiente enlace:

Tabla A.1: Tabla de características del conjunto de datos

Apéndice B – Repositorio del proyecto

El código fuente, scripts de preprocesamiento y modelos entrenados utilizados en este estudio se encuentran disponibles en el repositorio del proyecto. Esto permite a otros investigadores replicar los experimentos y explorar extensiones del trabajo.

Repositorio B.1: Repositorio del proyecto

Apéndice C – Recolección y caracterización y del conjunto de datos

Este apartado expone de manera detallada el proceso de recolección y caracterización del conjunto de datos CIC-IDS2017, empleado en el presente estudio.

Documento C.1: Recolección y caracterización del conjunto de datos CIC-IDS2017

REFERENCES

[1] I. Sharafaldin, A. H. Lashkari, y A. A. Ghorbani, "CICIDS2017: Intrusion Detection Evaluation Dataset," Canadian Institute for Cybersecurity, University of New Brunswick, 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[2] "Security Concept," ScienceDirect Topics. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/security-concept

[3] M. A. Elhadi y M. A. Ghorbani, "Classification and Importance of Intrusion Detection System," ResearchGate, 2025. [Online]. Available: https://www.researchgate.net/publication/340655192_Classification_and_Importance_of_Intrusion_Detection_System

[4] "Network Forensics: A Short Guide to Digital Evidence Recovery from Computer Networks," Forensic Focus, 15-Mar-2025. [Online]. Available: https://www.forensicfocus.com/guides/network-forensics-a-short-guide-to-digital-evidence-recovery-from-computer-networks/

[5] D. Herzalla, W. T. Lunardi y M. A. Lopez, "TII-SSRC-23 Dataset: Typological Exploration of Diverse Traffic Patterns for Intrusion Detection," arXiv, 14-Sep-2023. [Online]. Available: https://arxiv.org/abs/2310.10661

[6] İ. Yazıcı, "A survey of applications of artificial intelligence and machine learning in cybersecurity," Computers & Security, vol. 114, p. 102525, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2215098623001337

[7] T. Saranya, "Performance Analysis of Machine Learning Algorithms in Intrusion Detection Systems," Procedia Computer Science, vol. 171, pp. 1289–1296, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920311121

[8] M. Sharafaldin, A. H. Lashkari y A. A. Ghorbani, "CSE-CIC-IDS2018: Intrusion Detection Evaluation Dataset," Canadian Institute for Cybersecurity, University of New Brunswick, 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[9] E. A. C. Ribeiro, "CICIDS2017: Cleaned & Preprocessed," Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/ericanacletoribeiro/cicids2017-cleaned-and-preprocessed?resource=download

[10] P. Jairu y P. Pankaj, "A Supervised Machine Learning Approach to Network Intrusion Detection on CICIDS-2017 Dataset," Culminating Projects in Information Assurance, vol. 117, 2021. [Online]. Available: https://repository.stcloudstate.edu/msia_etds/117

[11] Z. Xu y Y. Liu, "Robust Anomaly Detection in Network Traffic: Evaluating Machine Learning Models on CICIDS2017," 2025. [Online]. Available: https://arxiv.org/abs/2506.19877

[12] S. S. Panwar, P. S. Negi, L. S. Panwar y Y. P. Raiwan, "Implementation of Machine Learning Algorithms on CICIDS-2017 Dataset for Intrusion Detection Using WEKA," Blue Eyes Intelligence Engineering & Sciences Publication, 2019, doi: 10.35940/ijrte.C4587.098319