GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA INTELIGENTE DE DETECCIÓN DE CIBERATAQUES MEDIANTE MACHINE LEARNING

Autor: Blanca Manrique Sanz

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid – 2026

Declaración de originalidad

Declaro bajo mi responsabilidad que el Proyecto presentado con el título **SISTEMA INTELIGENTE DE DETECCIÓN DE CIBERATAQUES MEDIANTE MACHINE LEARNING** en la ETS de Ingeniería – ICAI de la Universidad Pontificia Comillas en el curso académico **2025-2026** es de mi autoría y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Uso de Inteligencia Artificial[1]

Declaro bajo mi responsabilidad que (indicar la opción correcta):

☐ No he utilizado Inteligencia Artificial en la elaboración del presente documento.

☒ He utilizado Inteligencia Artificial en la elaboración del presente documento y/o del Anexo B siempre en las condiciones permitidas por la Universidad Pontificia Comillas, es decir, aplicando el Nivel 2 de la Escala de Evaluación de Perkins et al. (2024): "*La IA puede utilizarse para actividades previas a la tarea, como la lluvia de ideas, la descripción y la investigación inicial. Este nivel se centra en el uso de la IA para la planificación, las síntesis y la generación de ideas, pero las evaluaciones deben hacer hincapié en la capacidad de desarrollar y refinar estas ideas de forma independiente*". En concreto, la Inteligencia Artificial ha sido empleada para:

La Inteligencia Artificial se ha empleado como herramienta de apoyo en la fase inicial de planificación y estructuración de las ideas del proyecto. En particular, se ha utilizado para organizar preliminarmente los aspectos que se debían considerar en la selección de algoritmos de Machine Learning, explorar posibles alternativas y sintetizar sus principales características.

Firmado (alumno): BLANCA MANRIQUE SANZ
Fecha: 04/06/2026

[1] Esta declaración se refiere al uso de la Inteligencia Artificial generativa para realizar los documentos del Proyecto (Anexo B y Memoria). No aplica a Proyectos donde, por su naturaleza, deban emplear inteligencia artificial como parte de los mismos (aplicación de técnicas de aprendizaje automático, redes neuronales, análisis de datos…)

/Autorización para la entrega del Proyecto

El Director del Proyecto	El co-Director del Proyecto (si aplica)
7121631 4B ATILANO RAMIRO Signed by: 71216314B ATILANO RAMIRO FERNÁNDEZ-PACHECO Date: 2026-06-05 10:18:37 CEST	
Fdo:	Fdo:
Fecha:	Fecha:

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

SISTEMA INTELIGENTE DE DETECCIÓN DE CIBERATAQUES MEDIANTE MACHINE LEARNING

Autor: Blanca Manrique Sanz

Director: Atilano Fernández-Pacheco Sánchez-Migallón

Madrid - 2026

Agradecimientos

A mis padres,
Gracias por todo el apoyo a lo largo de estos años, que no siempre ha sido fácil. Habéis estado ahí en los momentos más difíciles, sin dudar nunca, y habéis sido el pilar fundamental que me ha permitido llegar hasta aquí. Gracias por vuestra paciencia, vuestro cariño y por creer en mí incluso cuando yo misma dudaba.

A mi hermana,
Sin ella no habría llegado hasta aquí. Ha sido parte fundamental desde el principio, acompañándome, ayudándome y aconsejándome en todo momento. Mi principal apoyo en cada paso, cada acierto o error. Gracias por estar ahí siempre, incluso cuando ni yo me aguanto.

A mis abuelos,
A Alejandro, Luis y Urbana, que me dejasteis durante el camino para cuidarme desde el cielo y me dais la fuerza que necesito cuando siento que no puedo. Y a mi abuela Isabel, que me acompaña en cada paso que doy. Los cuatro habéis sido siempre un referente para mí.

A mi tía Angelines,
Que siempre ha estado ahí durante todos estos años para todo. Llamando después de cada examen, cuidándome en todo momento y asegurándose de que todo iba en orden. Sin ti hubiese sido todo mucho más difícil. Gracias.

A mi tío Miguel,
Que no ha podido verlo, pero tío, ya puedo decir que soy ingeniera, con todas las letras. Espero que estés orgulloso. Va por ti.

A mis amigos,
A todos, por ser apoyo, confidentes, compañeros de biblioteca, de fiesta, de reír, llorar y, sobre todo, por no dejarme tirar la toalla en muchas ocasiones. Especialmente a Arantxa, Paula, Ale y Miguel. Gracias por ser casa aun estando fuera de ella.

SISTEMA INTELIGENTE DE DETECCIÓN DE CIBERATAQUES MEDIANTE MACHINE LEARNING

Autor: Manrique Sanz, Blanca.
Director: Fernández-Pacheco Sánchez-Migallón, Atilano.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El presente Trabajo de Fin de Grado desarrolla un sistema inteligente de detección de ciberataques basado en técnicas de Machine Learning aplicado al análisis de tráfico de red. Para ello, se han implementado y comparado diferentes modelos supervisados y no supervisados utilizando datasets especializados en ciberseguridad. Además, se ha desarrollado una aplicación web interactiva capaz de monitorizar tráfico de red y visualizar alertas de seguridad en tiempo real. Los resultados obtenidos muestran una elevada capacidad de detección de ataques DDoS, alcanzando métricas cercanas al 100% en precisión y F1-Score.

Palabras clave: Ciberseguridad, Machine Learning, Detección de intrusiones, DDoS, Inteligencia Artificial, Tráfico de red

1. Introducción

El incremento de ataques informáticos en redes empresariales y sistemas conectados ha provocado la necesidad de desarrollar mecanismos de detección cada vez más avanzados y automatizados. Entre las amenazas más frecuentes destacan los ataques distribuidos de denegación de servicio (DDoS), capaces de saturar sistemas y comprometer la disponibilidad de servicios críticos. [1]

Tradicionalmente, los Sistemas de Detección de Intrusiones (IDS) se han basado en reglas y firmas predefinidas. Sin embargo, estos sistemas presentan limitaciones importantes frente a amenazas nuevas o patrones de ataque desconocidos. En este contexto, las técnicas de Machine Learning han adquirido una gran relevancia dentro del ámbito de la ciberseguridad, permitiendo identificar comportamientos anómalos mediante el análisis inteligente del tráfico de red.

El objetivo principal de este proyecto consiste en diseñar e implementar un sistema inteligente de detección de ciberataques utilizando técnicas de aprendizaje automático. Para ello, se han desarrollado diferentes modelos supervisados y no supervisados capaces de clasificar tráfico benigno y tráfico malicioso a partir de datasets especializados en detección de intrusiones. [2]

Además del entrenamiento y evaluación de modelos, el proyecto incluye el desarrollo de una aplicación web interactiva para la visualización y monitorización de tráfico de red, permitiendo representar gráficamente los resultados obtenidos y generar alertas automáticas ante posibles situaciones de riesgo.

2. Definición del Proyecto

El proyecto desarrollado aborda el problema de detección inteligente de intrusiones mediante técnicas de inteligencia artificial aplicadas al análisis de tráfico de red. Para

ello, se ha diseñado una arquitectura completa que abarca todas las fases necesarias en un sistema de análisis de datos basado en Machine Learning:

- Carga y exploración del dataset.
- Limpieza y tratamiento de datos.
- Transformación y normalización de variables.
- Entrenamiento de modelos supervisados.
- Entrenamiento de modelos no supervisados.
- Evaluación de resultados.
- Visualización y monitorización mediante dashboard web.

Durante el desarrollo se han utilizado datasets especializados en ciberseguridad, principalmente CSE-CIC-IDS2018, debido a la gran variedad de ataques y registros disponibles. [3]

Los modelos supervisados implementados han sido: [4]

- Regresión Logística.
- Árbol de decisión.
- Random Forest.
- Gradient Boosting.

Por otro lado, en la parte de aprendizaje no supervisado se han utilizado técnicas de clustering y reducción de dimensionalidad:

- PCA
- K-Means
- Clustering jerárquico

Finalmente, se ha desarrollado una aplicación web interactiva mediante Streamlit capaz de analizar tráfico de red, representar métricas de seguridad en tiempo real y generar alertas automáticas ante comportamientos anómalos.

3. Descripción del sistema desarrollado

La arquitectura desarrollada integra diferentes módulos de procesamiento y análisis de datos orientados a la detección de amenazas en tráfico de red.

En primer lugar, los datos son cargados y procesados mediante técnicas de limpieza y normalización, eliminando valores nulos, infinitos y variables irrelevantes. Posteriormente, los datos son utilizados para entrenar diferentes modelos de Machine Learning encargados de clasificar el tráfico como benigno o malicioso.

El sistema incorpora además técnicas de reducción de dimensionalidad mediante PCA para facilitar la representación visual del tráfico de red y permitir identificar agrupaciones y anomalías dentro de los datos.

Como parte final del proyecto, se ha desarrollado un dashboard web interactivo capaz de visualizar métricas de seguridad, representar gráficamente el tráfico analizado y generar alertas automáticas cuando el porcentaje de tráfico malicioso supera determinados umbrales. [5]

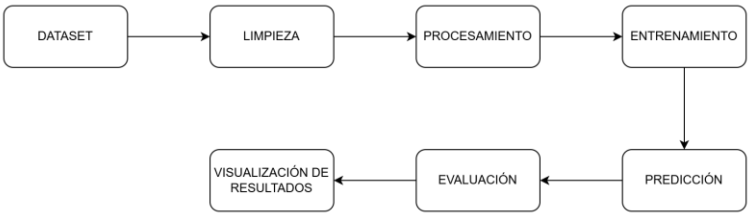


Ilustración 1. Arquitectura general del sistema inteligente de detección de ciberataques desarrollado en el proyecto.

Los modelos supervisados implementados han mostrado un rendimiento muy elevado en la detección de ataques DDoS, obteniendo métricas cercanas al 100% en accuracy, precisión y F1-Score.

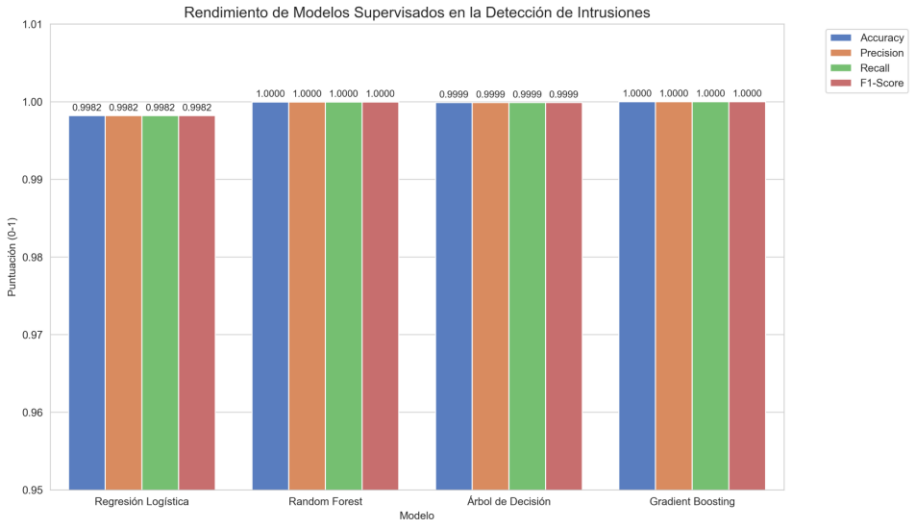


Ilustración 2. Comparación de los modelos supervisados implementados para la detección de intrusiones.

La aplicación web desarrollada permite monitorizar el tráfico de red y visualizar los resultados obtenidos mediante gráficas, tablas y métricas estadísticas.

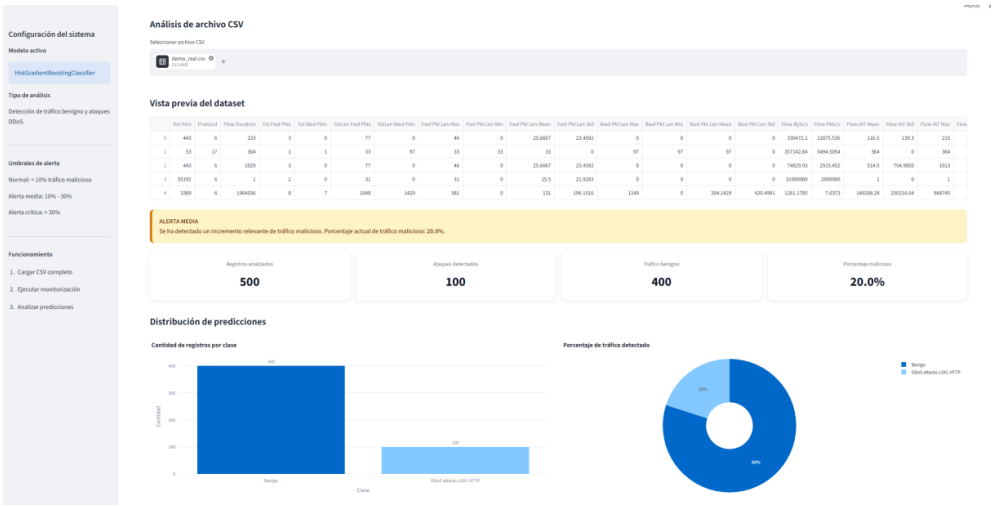


Ilustración 3. Dashboard web desarrollado para la monitorización y visualización de tráfico de red y detección de ataques.

El sistema incorpora además un módulo de alertas automáticas que informa al usuario cuando se detecta un porcentaje elevado de tráfico malicioso.

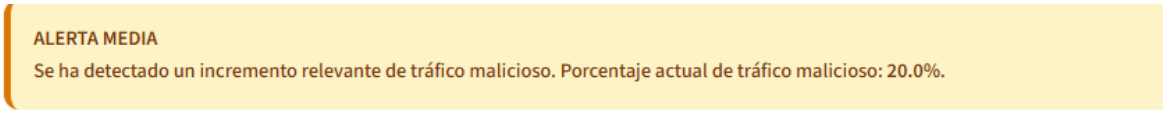


Ilustración 4. Sistema automático de alertas implementado para la detección de tráfico malicioso.

4. Resultados

Los resultados obtenidos durante el desarrollo del proyecto muestran una elevada eficacia de las técnicas de Machine Learning aplicadas a la detección de intrusiones.

Los modelos supervisados alcanzan métricas extremadamente elevadas, obteniendo valores próximos al 100% en accuracy, precisión, recall y F1-Score. Entre todos los modelos implementados, Gradient Boosting y Random Forest destacan como los más robustos y precisos.

Por otro lado, los modelos no supervisados permiten identificar agrupaciones automáticas dentro del tráfico de red y detectar anomalías sin necesidad de utilizar etiquetas previas.

Las técnicas de reducción de dimensionalidad mediante PCA han facilitado la representación visual del tráfico de red y la interpretación de patrones anómalos dentro de los datos.

Finalmente, la aplicación web desarrollada demuestra la viabilidad de integrar modelos de Machine Learning en sistemas interactivos de monitorización y análisis de tráfico de red en tiempo real.

5. Conclusiones

El desarrollo de este proyecto ha permitido demostrar que las técnicas de Machine Learning constituyen una herramienta altamente eficaz para la detección inteligente de ciberataques en tráfico de red.

Los resultados obtenidos validan la capacidad de los modelos supervisados para clasificar tráfico benigno y tráfico malicioso con una elevada precisión, mientras que las técnicas no supervisadas permiten complementar el análisis mediante la detección automática de agrupaciones y anomalías.

Además, el desarrollo del dashboard web aporta una capa adicional de visualización y monitorización que mejora la interpretación de resultados y facilita la detección temprana de amenazas.

Como líneas futuras de trabajo, sería interesante integrar captura real de paquetes de red, desplegar el sistema en entornos empresariales y explorar modelos de Deep Learning para mejorar la capacidad de detección frente a ataques más complejos.

6. Referencias

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, y K.-Y. Tung, «Intrusion detection system: A comprehensive review», *J. Netw. Comput. Appl.*, vol. 36, n.º 1, pp. 16-24, ene. 2013, doi: 10.1016/j.jnca.2012.09.004.

[2] K. A. Scarfone y P. M. Mell, «Guide to Intrusion Detection and Prevention Systems (IDPS)», National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[3] «IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) ». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv

[4] A. L. Buczak y E. Guven, «A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection», *IEEE Commun. Surv. Tutor.*, vol. 18, n.º 2, pp. 1153-1176, 2016, doi: 10.1109/COMST.2015.2494502.

[5] «Streamlit • A faster way to build and share data apps». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://streamlit.io/

INTRUSION DETECTION SYSTEM USING MACHINE LEARNING

Author: Manrique Sanz, Blanca.
Supervisor: Fernández-Pacheco Sánchez-Migallón, Atilano.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

The present Final Degree Project develops an intelligent cyberattack detection system based on Machine Learning techniques applied to network traffic analysis. To achieve this, different supervised and unsupervised models have been implemented and compared using specialized cybersecurity datasets. In addition, an interactive web application capable of monitoring network traffic and displaying real-time security alerts has been developed. The obtained results show a high capability for detecting DDoS attacks, reaching metrics close to 100% in precision and F1-Score.

Keywords: Cybersecurity, Machine Learning, Intrusion Detection, DDoS, Artificial Intelligence, Network Traffic Analysis

1. Introduction

The increase in cyberattacks targeting enterprise networks and connected systems has created the need for increasingly advanced and automated detection mechanisms. Among the most common threats are Distributed Denial of Service (DDoS) attacks, capable of saturating systems and compromising the availability of critical services.[1]

Traditionally, Intrusion Detection Systems (IDS) have been based on predefined rules and signatures. However, these systems present important limitations when dealing with new threats or previously unknown attack patterns. In this context, Machine Learning techniques have gained great relevance within the field of cybersecurity, enabling the identification of anomalous behaviors through intelligent network traffic analysis.

The main objective of this project is to design and implement an intelligent cyberattack detection system using Machine Learning techniques. For this purpose, different supervised and unsupervised models capable of classifying benign and malicious traffic have been developed using specialized intrusion detection datasets. [2]

In addition to model training and evaluation, the project includes the development of an interactive web application for network traffic visualization and monitoring, allowing graphical representation of the obtained results and the generation of automatic alerts in potentially dangerous situations.

2. Project Definition

The developed project addresses the problem of intelligent intrusion detection through artificial intelligence techniques applied to network traffic analysis. To achieve this, a complete architecture covering all the necessary stages of a Machine Learning-based data analysis system has been designed:

- Dataset loading and exploration.

- Data cleaning and preprocessing.
- Variable transformation and normalization.
- Supervised model training.
- Unsupervised model training.
- Results evaluation.
- Visualization and monitoring through a web dashboard.

During the development process, specialized cybersecurity datasets have been used, mainly CSE-CIC-IDS2018, due to the wide variety of attacks and available records. [3]

The implemented supervised models are: [4]

- Logistic Regression.
- Decision Tree.
- Random Forest.
- Gradient Boosting.

On the other hand, in the unsupervised learning stage, clustering and dimensionality reduction techniques have been used:

- PCA
- K-Means
- Hierarchical Clustering

Finally, an interactive web application using Streamlit has been developed, capable of analyzing network traffic, representing real-time security metrics, and generating automatic alerts in anomalous situations.

3. Description of the developed system

The developed architecture integrates different data processing and analysis modules aimed at detecting threats in network traffic.

First, the data is loaded and processed through cleaning and normalization techniques, removing null values, infinite values, and irrelevant variables. Subsequently, the data is used to train different Machine Learning models responsible for classifying traffic as benign or malicious.

The system also incorporates dimensionality reduction techniques through PCA in order to facilitate the visual representation of network traffic and allow the identification of clusters and anomalies within the data.

As a final part of the project, an interactive web dashboard has been developed, capable of visualizing security metrics, graphically representing the analyzed traffic and generating automatic alerts when the percentage of malicious traffic exceeds certain thresholds. [5]

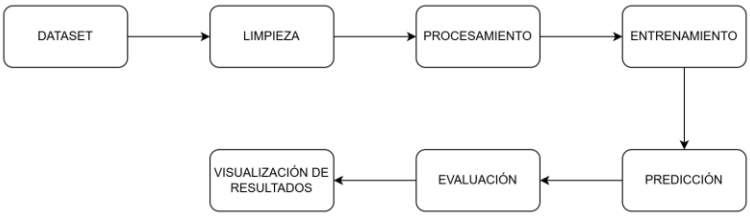


Figure 1. General architecture of the intelligent cyberattack detection system developed in the project.

The implemented supervised models have shown very high performance in detecting DDoS attacks, obtaining metrics close to 100% in accuracy, precision, and F1-Score.

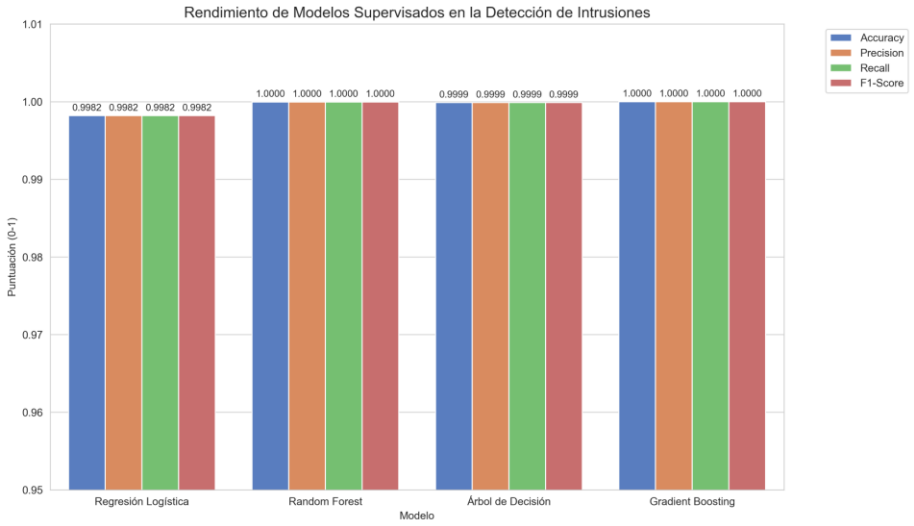


Figure 2. Comparison of the supervised models implemented for intrusion detection.

The developed web application allows network traffic monitoring and visualization of the obtained results through graphs, tables, and statistical metrics.

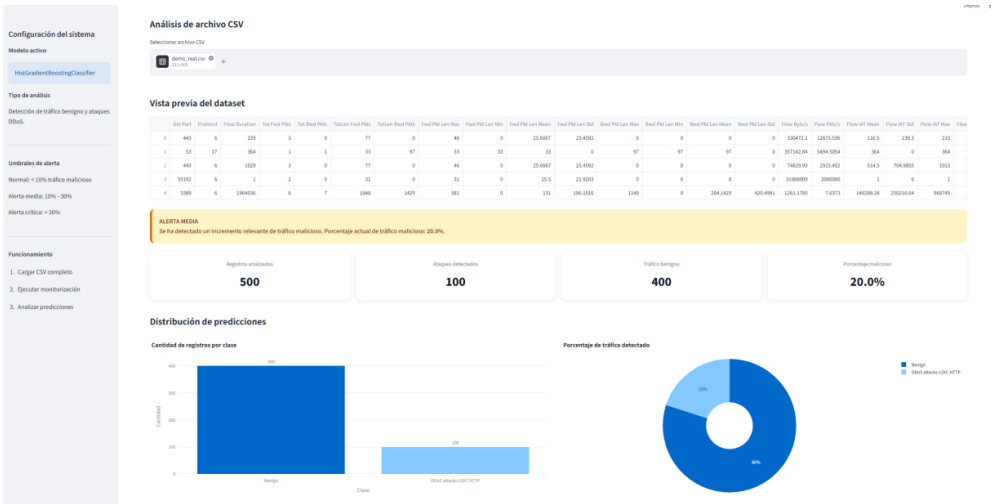


Figure 3. Web dashboard developed for network traffic monitoring and attack detection visualization.

The system also incorporates an automatic alert module that informs the user whenever a high percentage of malicious traffic is detected.

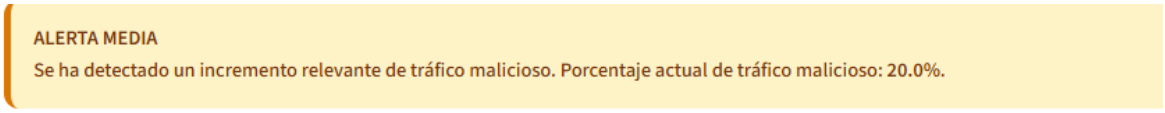


Figure 4. Automatic alert system implemented for malicious traffic detection.

4. Results

The results obtained during the development of the project demonstrate the high effectiveness of Machine Learning techniques applied to intrusion detection.

The supervised models achieve extremely high metrics, obtaining values close to 100% in accuracy, precision, recall and F1-Score. Among all the implemented models, Gradient Boosting and Random Forest stand out as the most robust and accurate.

On the other hand, the unsupervised models make it possible to identify automatic groupings within network traffic and detect anomalies without the need for previous labels.

Dimensionality reduction techniques through PCA have facilitated the visual representation of network traffic and the interpretation of anomalous patterns within the data.

Finally, the developed web application demonstrates the feasibility of integrating Machine Learning models into interactive real-time network traffic monitoring and analysis systems.

5. Conclusions

The development of this project has demonstrated that Machine Learning techniques constitute a highly effective tool for intelligent cyberattack detection in network traffic.

The obtained results validate the ability of supervised models to classify benign and malicious traffic with high precision, while unsupervised techniques complement the analysis through the automatic detection of clusters and anomalies.

In addition, the development of the web dashboard provides an additional layer of visualization and monitoring that improves result interpretation and facilitates early threat detection.

As future lines of work, it would be interesting to integrate real network packet capture, deploy the system in enterprise environments, and explore Deep Learning models to improve detection capabilities against more complex attacks.

6. References

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, y K.-Y. Tung, «Intrusion detection system: A comprehensive review», *J. Netw. Comput. Appl.*, vol. 36, n.º 1, pp. 16-24, ene. 2013, doi: 10.1016/j.jnca.2012.09.004.

[2] K. A. Scarfone y P. M. Mell, «Guide to Intrusion Detection and Prevention Systems (IDPS)», National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[3] «IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) ». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv

[4] A. L. Buczak y E. Guven, «A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection», *IEEE Commun. Surv. Tutor.*, vol. 18, n.º 2, pp. 1153-1176, 2016, doi: 10.1109/COMST.2015.2494502.

[5] «Streamlit • A faster way to build and share data apps». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://streamlit.io/

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

En el contexto digital actual, las organizaciones empresariales dependen cada vez más de la conectividad y el intercambio continuo de información a través de redes locales e infraestructuras en la nube. Esta interconexión ha ido creciendo y, con ella, ha aumentado significativamente la exposición a ciberamenazas, generando un entorno en el que la ciberseguridad se ha convertido en un elemento crítico para garantizar la continuidad del negocio y la protección de los datos.

Además, la transformación digital ha impulsado la adopción de tecnologías como el *cloud computing*, el Internet de las Cosas (IoT) y los sistemas distribuidos, lo que incrementa aún más la superficie de ataque. En este contexto, cualquier vulnerabilidad puede ser explotada, comprometiendo información sensible o interrumpiendo servicios esenciales. Por ello, resulta imprescindible contar con mecanismos de detección eficaces que permitan detectar las amenazas de forma temprana.

Los ciberataques dirigidos a entornos empresariales se han diversificado y sofisticado en los últimos años. Desde ataques de denegación de servicio que saturan los servidores corporativos, hasta intentos de fuerza bruta, escaneos de puertos para detectar vulnerabilidades o inyecciones maliciosas, las empresas se enfrentan a amenazas cada vez más complejas y difíciles de detectar mediante métodos tradicionales. Muchos de estos ataques, además, están en constante evolución, lo que hace más difícil su identificación basada únicamente en firmas o reglas predefinidas.

En este aspecto, los sistemas clásicos de detección de intrusiones (IDS) presentan ciertas limitaciones, especialmente frente ataques desconocidos o patrones de comportamiento no registrados previamente. A esto se le suma el enorme volumen de datos que circula por las redes, lo que hace que un análisis manual sea inviable y aumenta la probabilidad de que actividades maliciosas pasen desapercibidas.

Ante esta problemática, el uso de técnicas de Machine Learning se ha consolidado como una alternativa prometedora. Estos modelos permiten analizar grandes cantidades de tráfico de red y detectar comportamientos anómalos de forma automática, adaptándose a nuevas amenazas sin necesidad de reglas detalladas. Gracias a su capacidad de aprendizaje, pueden identificar patrones complejos que resultarían difíciles de detectar mediante métodos más convencionales.

Este proyecto propone el desarrollo de un sistema inteligente de detección de ciberataques enfocado en redes empresariales, utilizando modelos de Machine Learning aplicados al tráfico de red. La propuesta combina algoritmos de clasificación y detección de anomalías con herramientas de visualización de datos, con el objetivo de ofrecer una solución no solo eficiente, sino también comprensible para el usuario.

Para ello, se van a emplear datasets públicos, ampliamente utilizados en la investigación científica de ciberseguridad. Estos conjuntos de datos incluyen tanto tráfico legitimo como diferentes tipos de ataques, lo que permite entrenar y evaluar modelos con gran realismo.

El desarrollo del proyecto abarca distintas fases, incluyendo el procesamiento de los datos, la selección y entrenamiento de modelos, la evaluación de su rendimiento y la interpretación de los resultados. Asimismo, se busca analizar las ventajas y limitaciones de cada enfoque, con el fin de identificar la solución más adecuada para este tipo de entornos.

En definitiva, este trabajo pretende aportar una visión práctica sobre el uso de Machine Learning en la detección de ciberataques, explorando su potencial como herramienta para mejorar la seguridad en redes empresariales y contribuyendo al desarrollo de soluciones más avanzadas en el ámbito de la ciberseguridad.

1.1 MOTIVACIÓN DEL PROYECTO

Continuando con lo expuesto en la introducción, el análisis de los diferentes métodos de detección de ciberataques pone de manifiesto que muchas de las soluciones existentes presentan limitaciones importantes en entornos reales. Aunque los sistemas tradicionales de

detección de intrusiones han sido durante años una pieza fundamental dentro de las arquitecturas de seguridad empresarial, el crecimiento exponencial del tráfico de red y la sofisticación de las amenazas actuales han reducido notablemente su eficacia frente a determinados escenarios.

Uno de los principales inconvenientes de los sistemas IDS tradicionales basados en firmas está en su carácter reactivo. Estos mecanismos dependen de bases de datos que contienen patrones de ataques previamente conocidos, lo que implica que únicamente pueden detectar amenazas que ya han sido identificadas con anterioridad. Como consecuencia, los comportamientos anómalos no registrados previamente pueden pasar desapercibidos, comprometiendo la seguridad de la infraestructura sin generar alertas tempranas.

A este problema se le suma el enorme volumen de información que circula por las redes empresariales. El incremento de dispositivos conectados, servicios cloud, aplicaciones distribuidas y sistemas IoT genera cantidades masivas de tráfico que resultan imposibles de analizar manualmente de forma eficiente. Incluso utilizando herramientas automatizadas tradicionales, la cantidad de eventos generados puede derivar en un elevado número de falsos positivos, dificultando el trabajo de los equipos de seguridad y reduciendo la capacidad de respuesta ante incidentes reales.

Por otro lado, muchas soluciones comerciales avanzadas presentan importantes barreras económicas y técnicas para pequeñas y medianas organizaciones. Los sistemas de monitorización y análisis de seguridad más sofisticados requieren infraestructuras complejas, personal altamente cualificado y elevados costes de mantenimiento y actualización. Esto provoca que numerosas empresas continúen utilizando mecanismos de protección insuficientes frente a amenazas demasiado modernas.

En este contexto, el uso de técnicas de Machine Learning representa una alternativa especialmente prometedora. Los modelos de aprendizaje automático permiten analizar grandes volúmenes de datos de forma automática, identificar relaciones complejas entre variables y detectar patrones de comportamiento anómalo sin necesidad de definir manualmente reglas específicas para cada situación.

Además, el aprendizaje automático ofrece la posibilidad de combinar distintos enfoques de detección integrando modelos supervisados y no supervisados capaces de abordar el problema desde diferentes perspectivas. Mientras que los algoritmos supervisados permiten clasificar tráfico conocido con altos niveles de precisión, las técnicas no supervisadas facilitan la identificación de anomalías y comportamientos inesperados en la red.

La motivación principal de este Trabajo Fin de Grado surge precisamente del interés para analizar el potencial real de estas técnicas aplicadas en el ámbito de la ciberseguridad. Más concretamente, se busca estudiar cómo diferentes modelos de Machine Learning pueden emplearse para detectar tráfico malicioso dentro de redes empresariales, evaluando tanto su capacidad de clasificación como su comportamiento frente a distintos tipos de ataques.

Además, este trabajo nace también de la creciente que la inteligencia artificial y el análisis de datos están adquiriendo dentro del ámbito de la ingeniería y la ciberseguridad. La integración de ambos campos representa una de las líneas de investigación con mayor proyección actualmente y constituye una oportunidad interesante para desarrollar soluciones eficientes, adaptativas y escalables frente a las amenazas digitales del futuro.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

2.1 PYTHON

Python ha sido el lenguaje de programación principal utilizado durante el desarrollo del proyecto debido a su versatilidad, facilidad de uso y amplia disponibilidad de herramientas y librerías orientadas al análisis de datos y al aprendizaje automático.[6]

Su amplio abanico de librerías especializadas permite trabajar de forma eficiente en todas las fases del proyecto, desde la parte del procesamiento de datos hasta el entrenamiento y evaluación de los modelos.

Además, Python se ha consolidado como uno de los lenguajes más utilizados en la investigación y el desarrollo del ámbito de la inteligencia artificial y la ciberseguridad, esto facilita el acceso a documentación recursos y herramientas avanzadas

Figura 1. Logo de Python

2.1.1 LIBRERÍAS DE ANÁLISIS Y PROCESAMIENTO DE DATOS

2.1.1.1 Pandas

La librería Pandas se ha utilizado para la carga, manipulación y procesamiento de los datasets utilizados durante el proyecto.

Debido a sus estructuras de datos tipo DataFrame, Pandas permite trabajar de forma eficiente con grandes volúmenes de información, facilitando tareas como la limpieza de datos, el tratamiento de los valores nulos, filtrado y selección de variables, la agrupación de los registros y el análisis estadístico preliminar.

Esta librería ha resultado especialmente útil en las fases de exploración y procesado de los datasets.

Figura 2. Logo de pandas

2.1.1.2 NumPy

NumPy se ha empleado para realizar operaciones matemáticas y para el manejo de estructuras multidimensionales de datos.

Esta librería proporciona soporte para arrays de alto rendimiento y funciones optimizadas para cálculo numérico. Resulta especialmente útil en operaciones relacionadas con transformación de variables, cálculos estadísticos, matrices y preparación de datos para modelos de Machine Learning.

Además, muchas librerías de inteligencia artificial y análisis de datos utilizadas están construidas sobre NumPy.

Figura 3. Logo de NumPy

2.1.2 LIBRERÍAS DE MACHINE LEARNING

2.1.2.1 Scikit-learn

La librería Scikit-Learn se ha utilizado como herramienta principal para la implementación de los modelos de Machine Learning desarrollados durante el proyecto. [7]

Esta librería tiene algoritmos optimizados y fáciles de utilizar para tareas de clasificación, regresión, clustering, reducción de dimensionalidad y evaluación de modelos.

Figura 4. Logo de scikit-learn

2.1.2.2 Joblib

La librería joblib Se ha utilizado principalmente para la serialización y almacenamiento de los modelos entrenados.

Gracias a esta librería ha sido posible guardar modelos de Machine Learning ya entrenados, así como transformaciones aplicadas a los datos, permitiendo reutilizar los posteriormente sin necesidad de repetir el proceso completo del entrenamiento.

Figura 5. Logo de JobLib

2.1.3 LIBRERÍAS DE VISUALIZACIÓN DE DATOS

2.1.3.1 Matplotlib

La librería Matplotlib Se ha utilizado para generar representaciones gráficas orientadas al análisis y visualización de los resultados obtenidos.

Mediante esta herramienta se han elaborado gráficos relacionados con la distribución de clases, matrices de confusión, comparativas métricas, importancia de variables y representación de clusters.

Figura 6. Logo de matplotlib

2.1.3.2 Seaborn

Seaborn Se ha utilizado como complemento de Matplotlib para crear visualizaciones y estadísticas más avanzadas y con una representación visual más clara.

Esta librería ha facilitado la elaboración de mapas de calor, gráficos de distribución, gráficas comparativas y representaciones y estadísticas complejas.

Figura 7. Logo de seaborn

2.2 *ENTORNO DE DESARROLLO*

2.2.1 VISUAL STUDIO CODE

Visual Studio Code es un editor de código abierto desarrollado por Microsoft gratuito. Se ha sido utilizado como entorno de desarrollo principal para la organización y edición del código fuente del proyecto. Este editor proporciona funcionalidades orientadas al desarrollo en Python, entre las que destacan: resaltado de sintaxis, depuración de código, integración con notebooks, contra el de versiones y gestión de extensiones.[8]

Figura 8. Logo de Visual Studio Code

2.2.2 JUPYTER NOTEBOOK

Jupyter Notebook se ha utilizado como entorno interactivo para el desarrollo experimental y el análisis de los modelos de Machine Learning. Su ventaja principal es que puede combinar código, resultados, visualizaciones y documentación dentro de un mismo entorno de trabajo. Esto permite realizar pruebas iterativas, analizar resultados de forma visual y documentar cada etapa del proceso de manera estructurada.

Figura 9. Logo de Jupyter

2.3 CONTROL DE VERSIONES

2.3.1 GITHUB

Se ha utilizado GitHub Para controlar las versiones y el almacenamiento del código a lo largo del desarrollo del proyecto. El uso de esta plataforma permite mantener un seguimiento organizado de los cambios realizados, facilitando la gestión de distintas versiones. Además, proporciona una forma segura de almacenar notebooks, scripts y resultados obtenidos durante las diferentes fases.

Figura 10. Logo de GitHub

Capítulo 3. ESTADO DE LA CUESTIÓN

La creciente digitalización de los entornos empresariales y el aumento continuo de las amenazas informáticas han convertido la ciberseguridad en uno de los principales desafíos tecnológicos actuales. Las organizaciones modernas dependen de infraestructuras interconectadas que generan grandes volúmenes de tráfico de red y manejan información sensible de forma constante. Como consecuencia, la necesidad de detectar actividades maliciosas de manera rápida y precisa se ha vuelto fundamental para garantizar la seguridad y continuidad de los servicios.

Tradicionalmente, la detección de intrusiones se ha basado en sistemas fundamentados en firmas y reglas predefinidas capaces de identificar ataques conocidos. Sin embargo, la sofisticación creciente de las amenazas actuales, junto con la aparición de variantes desconocidas y de ataques de día cero, ha impulsado el desarrollo de nuevas técnicas basadas en inteligencia artificial y aprendizaje automático.

Actualmente, las investigaciones en este ámbito se centran en el desarrollo de sistemas inteligentes capaces de analizar grandes volúmenes de tráfico de red, detectar comportamientos anómalos y adaptarse dinámicamente a nuevos escenarios de ataque. Estos sistemas combinan técnicas tradicionales de detección con algoritmos avanzados de Machine Learning, análisis estadístico y procesamiento masivo de datos.

3.1 SISTEMAS TRADICIONALES DE DETECCIÓN DE INTRUSIONES (IDS)

Los sistemas tradicionales de detección de intrusiones, conocidos como IDS, constituyen una de las tecnologías más utilizadas históricamente en ciberseguridad para identificar actividades sospechosas dentro de una red o sistema informático. [1]

Su funcionamiento se basa en comparar el tráfico de red capturado con un conjunto de firmas conocidas que representan patrones de ataques que han sido previamente identificados. Cada firma codifica las características específicas de un ataque concreto, como consecuencia de bytes en la carga útil de los paquetes, flags TCP anómalos o patrones de acceso a los recursos del sistema [9].

Cuando el comportamiento observado coincide con alguna de estas reglas, el sistema genera una alerta indicando la posible existencia de una intrusión. Este enfoque presenta importantes ventajas:

- Elevada velocidad de detección
- Bajo coste computacional
- Facilidad de interpretación
- Gran eficacia frente a amenazas previamente conocidas

Entre las herramientas más utilizadas destacan:

- Snort: desarrollado originalmente por Cisco, ampliamente empleado en entornos empresariales y académicos.
- Suricata: orientado a redes de alto rendimiento y capaz de realizar procesamiento multihilo e inspección profunda de paquetes.
- Zeek IDS: enfocado al análisis avanzado del tráfico de red y generación de eventos y seguridad.

3.1.1 TIPOS DE SISTEMAS IDS

Los sistemas IDS pueden clasificarse según el tipo de información analizada y su ubicación dentro de la infraestructura.

- Network Intrusion Detection System (NIDS): Estos sistemas monitorizan el tráfico que circula por la red con el objetivo de identificar patrones de comportamiento malicioso. Normalmente se despliegan en Puntos estratégicos como routers, switches o firewalls.

Su principal ventaja es la capacidad de supervisar varios dispositivos de forma simultánea desde un único punto de análisis.

- Host Intrusion Detection System (HIDS): Estos sistemas se ejecutan directamente sobre equipos individuales y analizan eventos locales como: accesos al sistema, ejecución de procesos, modificación de archivos o registros y actividad del sistema operativo.

  Este enfoque permite detectar amenazas que podrían no ser visibles desde el tráfico de red.

3.1.2 LIMITACIONES DE LOS IDS TRADICIONALES

A pesar de que son muy utilizados, los IDS basados en firmas presentan muchas limitaciones estructurales. Su principal inconveniente es la incapacidad para detectar ataques desconocidos o variantes nuevas de amenazas existentes, ya que su funcionamiento depende de los patrones maliciosos que hayan sido previamente identificados y almacenados en bases de firmas actualizadas.[2]

Además, el mantenimiento continuo de estas bases supone un coste operativo considerable, especialmente en infraestructuras de gran tamaño. Otro problema relevante aparece en entornos con tráfico cifrado o elevados volúmenes de datos, donde la eficacia de los sistemas puede verse reducida.

Como respuesta a estas limitaciones surgen los sistemas IPS (Intrusion Prevention Systems), Capaces no solo de detectar actividades sospechosas, sino también de bloquear las automáticamente en tiempo real. Sin embargo, aunque amplíen las capacidades de los IDS tradicionales, continúan presentando dificultades para identificar amenazas no registradas.

En este contexto, el presente proyecto propone una aproximación basada en técnicas de Machine Learning con el objetivo de superar algunas de las limitaciones de los IDS tradicionales. A diferencia de los sistemas basados únicamente en firmas, los modelos de aprendizaje automático permiten analizar patrones de comportamiento en el tráfico de red e

identificar anomalías que podrían estar asociadas a ataques desconocidos o variantes no registradas previamente.

Además, se incorporan modelos supervisados y no supervisados capaces de adaptarse a diferentes escenarios de tráfico y mejorar la capacidad de detección frente a amenazas emergentes. Junto con ello, el sistema desarrollado incluye una herramienta de visualización de resultados que facilita la interpretación de las alertas y métricas obtenidas, proporcionando un apoyo más eficiente para el análisis de seguridad en entornos empresariales.

3.2 ENFOQUES BASADOS EN MACHINE LEARNING

Con el objetivo de superar las limitaciones de los sistemas tradicionales, se desarrollaron sistemas inteligentes de detección de intrusiones basados en técnicas de Machine Learning.

Estos sistemas permiten analizar automáticamente grandes cantidades de tráfico de red y aprender patrones complejos de comportamiento, siendo capaces de detectar anomalías incluso ante ataques no observados previamente. [4]

A diferencia de los sistemas basados únicamente en firmas, los modelos de aprendizaje automático adaptarse dinámicamente a cambios de comportamiento en el tráfico. Actualmente, los enfoques más utilizados pueden clasificarse en dos grandes grupos: modelos supervisados y modelos no supervisados

3.2.1 MODELOS SUPERVISADOS

Los modelos supervisados son aquellos que se entrenan con datos previamente etiquetados, es decir, en lo que se conoce si cada conexión pertenece a tráfico normal o a un ataque. El objetivo de estos es construir un modelo capaz de predecir la clase o el comportamiento de nuevas observaciones [10], [11].

.

3.2.1.1 Regresión Logística

La regresión logística es 1 de los algoritmos supervisados más utilizados en tareas de clasificación binaria debido a su simplicidad y su interpretabilidad. Su funcionamiento se basa en el uso de una función logística capaz de estimar la probabilidad de pertenencia de una observación a una clase.

En ciberseguridad, este modelo se utiliza frecuentemente para distinguir entre tráfico benigno y tráfico malicioso a partir de características estadísticas extraídas de los flujos de red. Aunque presenta limitaciones frente a relaciones altamente no lineales, sigue siendo muy utilizado como modelo base debido a su rapidez de entrenamiento y bajo coste computacional.[12]

3.2.1.2 Árbol de decisión

Los árboles de decisión clasifican los datos mediante una estructura jerárquica basada en reglas construidas a partir de umbrales sobre las distintas variables del dataset. Cada nodo representa una decisión sobre una característica concreta y las ramas generan particiones de los datos hasta alcanzar una clasificación final.

Su principal ventaja es la facilidad de interpretación, ya que permite visualizar claramente el proceso de decisión seguido por el modelo. No obstante, puede presentar problemas de sobreajuste cuando el árbol alcanza una complejidad muy alta.

3.2.1.3 Random Forest

Random Forest es un algoritmo basado en múltiples árboles de decisión entrenados sobre subconjuntos aleatorios de datos y variables. El modelo combina las predicciones de todos los árboles para obtener una clasificación final más rígida y estable.[13]

Este enfoque permite reducir de forma significativa el sobreajuste presente en los árboles individuales y mejorar la capacidad de generalización. Actualmente, Random Forest es uno de los algoritmos más utilizados en sistemas de detección de intrusiones basados en Machine

Learning debido a su elevada precisión y su buen rendimiento en datasets complejos y desbalanceados.

3.2.1.4 Gradient Boosting

Gradient Boosting es un modelo avanzado basado en el entrenamiento secuencial de múltiples árboles débiles. Cada árbol intenta corregir los errores cometidos por las iteraciones anteriores, mejorando progresivamente la capacidad predictiva del sistema.

Este algoritmo suele ofrecer resultados muy precisos en tareas de clasificación complejas, aunque requiere mayores recursos computacionales y un ajuste más cuidadoso de hiperparámetros.

3.2.2 MODELOS NO SUPERVISADOS

Estos modelos se aplican cuando no se dispone de etiquetas o si son poco fiables. Su objetivo es descubrir estructuras o patrones anómalos dentro de los datos [10].

En el ámbito de la ciberseguridad, esos algoritmos resultan especialmente útiles para detectar comportamientos anómalos y amenazas desconocidas.

3.2.2.1 PCA

El análisis de componentes principales es una técnica para reducir la dimensión que transforma el conjunto original de variables en un número reducido de componentes principales no correlacionadas.

El objetivo de PCA consiste en conservar la mayor cantidad posible de información utilizando menos dimensiones. En sistemas de detección de intrusiones, esta técnica permite: reducir complejidad computacional, eliminar redundancia entre variables y facilitar la visualización del tráfico de red.

3.2.2.2 K-Means

K-Means es uno de los algoritmos de clustering más utilizados en aprendizaje no supervisado. Su funcionamiento se basa en dividir los datos en grupos o clusters según la similitud existente entre observaciones.

En ciberseguridad, K-Means se emplea frecuentemente para: identificar patrones de comportamiento, segmentar tráfico de red y detectar anomalías alejadas del comportamiento normal.

3.2.2.3 Clustering

El clustering jerárquico constituye agrupaciones de datos mediante relaciones de proximidad entre las observaciones. A diferencia de K-Means, no requiere definir previamente el número de grupos y permite representar las relaciones mediante estructuras jerárquicas conocidas como dendrogramas.

3.3 SOLUCIONES HÍBRIDAS Y SISTEMAS RECIENTES

En la actualidad, predominan las soluciones híbridas, que combinan detección basada en firmas con análisis inteligente de comportamiento [14]. Este enfoque permite aprovechar la rapidez y precisión de los sistemas basados en firmas, junto con la capacidad adaptativa de las modelos inteligentes. Entre las soluciones modernas más utilizadas destacan:

- Wazuh
- Elastic Security
- Splunk
- IBM QRadar
- Microsoft Sentinel

Estas plataformas integran correlación avanzada de eventos, análisis estadístico y detección de anomalías mediante inteligencia artificial. Estas soluciones se apoyan en técnicas de Big

Data Analytics, esto les permite procesar grandes volúmenes de registros y flujos de red en tiempo real.

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

El incremento constante de ciberataques en entornos empresariales ha convertido la ciberseguridad en una necesidad fundamental para garantizar la protección de los sistemas de información y los datos sensibles. Los sistemas tradicionales de detección de intrusiones presentan limitaciones frente a amenazas avanzadas y ataques desconocidos, especialmente en escenarios con grandes volúmenes de tráfico y técnicas de ataque cada vez más sofisticadas.

En este contexto, las técnicas de Machine Learning ofrecen nuevas posibilidades para mejorar la capacidad de detección mediante el análisis automático de patrones y anomalías en el tráfico de red. Este proyecto surge con el propósito de estudiar e implementar modelos inteligentes capaces de identificar comportamientos maliciosos y contribuir al desarrollo de soluciones más adaptables y eficientes en el ámbito de la ciberseguridad.

4.2 METODOLOGÍA

El desarrollo el proyecto se ha organizado en distintas fases, desde el análisis inicial hasta la implementación y evaluación de los resultados.

La metodología seguida adopta un enfoque secuencial e incremental, comenzando por la revisión bibliográfica y el estudio del Estado del arte en sistemas de detección de intrusiones y técnicas de Machine Learning aplicadas a la ciberseguridad. Posteriormente, se ha realizado la selección, limpieza y preparación de los datasets utilizados para el entrenamiento y validación de los modelos.

A continuación, se han desarrollado e implementado modelos supervisados y no supervisados con el objetivo de detectar patrones anómalos de actividades maliciosas en el

tráfico de red. Finalmente, se ha llevado a cabo la evaluación comparativa de los modelos obtenidos y el diseño de un sistema de visualización que facilite la interpretación de los resultados.

Cada una de estas fases incluye tareas específicas orientadas a garantizar un desarrollo progresivo y estructurado del proyecto.

4.3 PLANIFICACIÓN

La planificación temporal del proyecto se ha organizado a lo largo del curso académico, distribuyendo las diferentes tareas según las fases de análisis, preparación de datos, desarrollo de modelos, validación y visualización de resultados.

A continuación, se muestra la organización temporal del trabajo, estructurada por meses y semanas, permitiendo realizar un seguimiento progresivo de las actividades desarrolladas durante el proyecto.

TAREAS		OCTUBRE				NOVIEMBRE				DICIEMBRE				ENERO				FEBRERO				MARZO				ABRIL				MAYO			
		S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
ANALISIS INICIAL Y PLANIFICACIÓN	Revisión bibliográfica																																
	Clasificación de enfoques																																
	Definición del alcance y objetivos																																
	Elaboración del Anexo B																																
SELECCIÓN Y PREPARACIÓN DE DATOS	Identificación y descarga de datasets																																
	Limpieza y procesamiento																																
	Análisis exploratorio de datos																																
	Selección de características																																
DESARROLLO DE MODELOS DE DETECCIÓN	Entrenamiento modelos supervisados																																

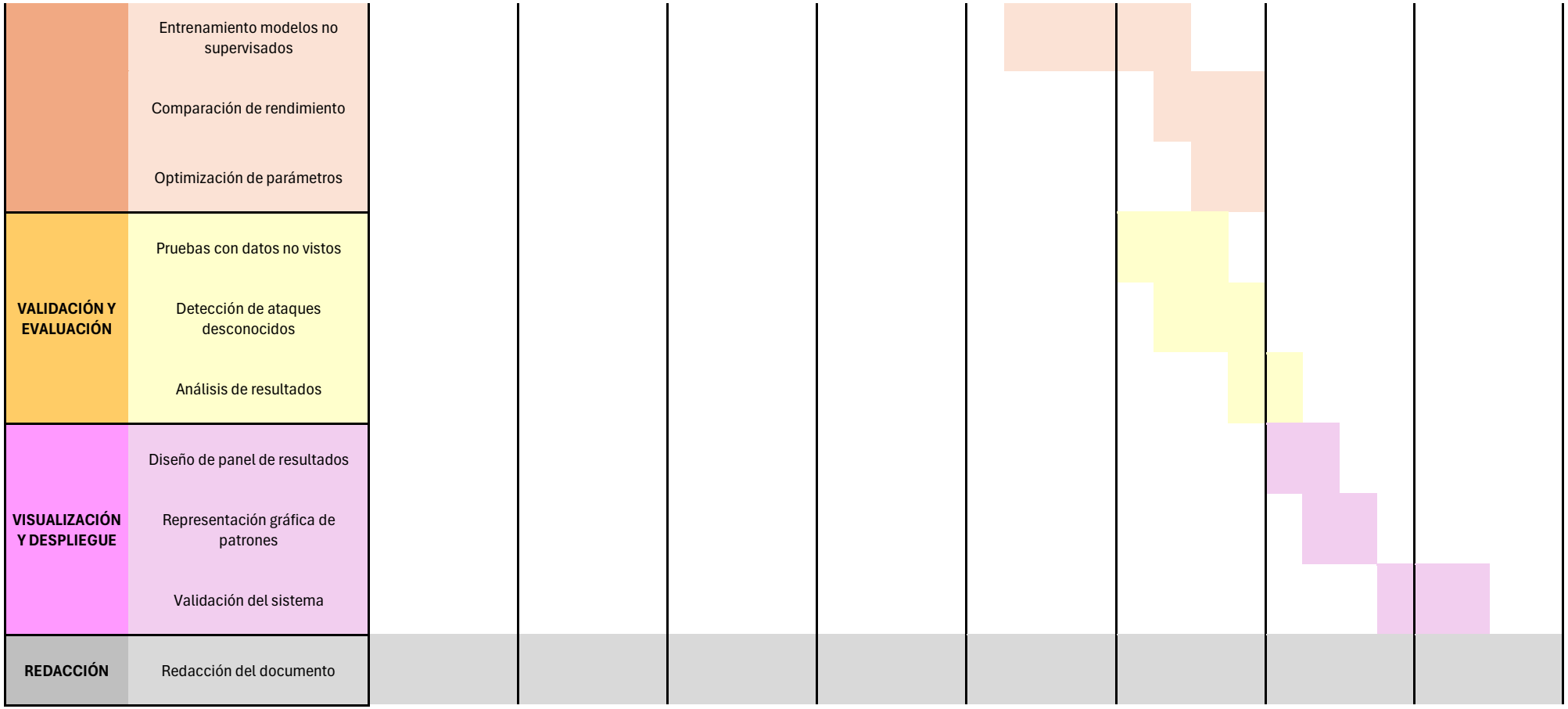


Figura 11. Diagrama de planificación

4.4 OBJETIVOS

El objetivo general de este proyecto es diseñar y evaluar un sistema inteligente de detección de ciberataques en entornos empresariales, basado en técnicas de Machine Learning, capaz de identificar tanto amenazas como patrones anómalos que indicen comportamientos maliciosos. El sistema busca integrar y aplicar modelos supervisados y no supervisados dentro de un flujo de análisis, detección y visualización. Para poder alcanzar este propósito, se han definido objetivos específicos:

- Analizar y preparar los datos para la detección de intrusiones: estudiar las técnicas actuales de detección basadas en firmas, reglas y aprendizaje automático, así como seleccionar y procesar conjuntos de datos públicos para su utilización en el entrenamiento de modelos.
- Desarrollar y evaluar modelos de detección: implementar modelos supervisados y no supervisados, comparando su rendimiento para identificar la solución más adecuada según distintos escenarios de tráfico y tipos de ataque.
- Diseñar un sistema de visualización de resultados: crear un panel gráfico que permita representar e interpretar de forma clara los resultados obtenidos por los modelos desarrollados.

4.5 ESTIMACIÓN ECONÓMICA

A la hora de desarrollar un proyecto tecnológico orientado a la detección inteligente de ciberataques, es necesario realizar una estimación económica que permita analizar la viabilidad del sistema desarrollado.

Para ello, se han considerado los costes asociados al software utilizado, los recursos materiales empleados durante el desarrollo y el capital humano necesario para la realización del proyecto.

4.5.1 COSTES DE SOFTWARE

Para el desarrollo del proyecto se han utilizado principalmente herramientas gratuitas y de código abierto, lo que reduce considerablemente el coste asociado a las licencias de software.

HERRAMIENTA	*FUNCIÓN PRINCIPAL*	*COSTE*
Python	Desarrollo del sistema	Gratuito
Visual Studio Code	Entorno de desarrollo	Gratuito
Jupyter Notebook	Experimentación y análisis	Gratuito
Scikit-Learn	Machine Learning	Gratuito
Pandas	Procesamiento de datos	Gratuito
NumPy	Cálculo numérico	Gratuito
Matplotlib	Visualización de datos	Gratuito
Seaborn	Visualización estadística	Gratuito

Tabla 1. Software utilizado durante el proyecto

Aunque el software utilizado no supone un coste directo, si debe considerarse el consumo de recursos computacionales durante el procesamiento de grandes volúmenes de datos y el entrenamiento de modelos.

4.5.2 RECURSOS MATERIALES

En cuanto a los recursos materiales, el desarrollo del proyecto se ha realizado utilizando un ordenador portátil personal y un monitor auxiliar para facilitar el trabajo de programación, análisis de datos y entrenamiento de modelos

4.5.2.1 Ordenador portátil

El desarrollo principal del proyecto se ha realizado utilizando un ordenador portátil equipado con procesador Intel Core i5 de última generación, 16 GB de memoria RAM y Sistema operativo Windows.

Dicho equipo ha permitido ejecutar las tareas de procesamiento de datos, entrenamiento de modelos y análisis de resultados necesarios para el desarrollo del sistema de detección de intrusiones.

Considerando el precio medio actual de un portátil con esas características, estimado en aproximadamente 1.100€, y una vida útil de 5 años, el coste proporcional asociado al tiempo de utilización durante el desarrollo del TFG (9 meses) se estima en:

$$\frac{1100 \times 9}{60} = 165€$$

4.5.2.2 Monitor auxiliar

Para mejorar la productividad y facilitar el análisis simultánea de código, resultados y documentación, se ha utilizado un monitor auxiliar HP de 27 pulgadas.

El coste aproximado de este tipo de monitor se sitúa en torno a 250€. Considerando al igual que el ordenador portátil una vida útil estimada de 5 años y el tiempo de uso asociado al proyecto, el coste proporcional es de:

$$\frac{250 \times 9}{60} = 37.5€$$

RECURSO	*COSTE APROXIMADO*	*COSTE EN EL PROYECTO*
Ordenador portátil	1.100€	165€
Monitor	250€	38€

Tabla 2. Recursos materiales utilizados

4.5.3 CAPITAL HUMANO

El capital humano representa el principal coste asociado al desarrollo el proyecto, debido al tiempo necesario para la investigación bibliográfica, el análisis y procesamiento de datasets, implementación de los modelos de Machine Learning, entrenamiento y validación, análisis de resultados, elaboración de gráficas y redacción de la memoria final.

Se estima que el desarrollo completo del proyecto ha requerido aproximadamente 450h de trabajo. Tomando como referencia el salario medio actual de un desarrollador junior en España, estimado en 18€/h el coste asciende a:

$$450 \times 18 = 8100€$$

Además, debe considerarse la supervisión académica y revisión técnica realizada durante el desarrollo del proyecto. Estimando unas 50 horas de dedicación y un coste medio de 30€/hora para labores de dirección y supervisión, el coste es de:

$$50 \times 30 = 1500€$$

CONCEPTO	*HORAS ESTIMADAS*	*COSTE POR HORA*	*COSTE TOTAL*
Desarrollo del proyecto	450 h	18 €/h	8.100€
Supervision y dirección	50 h	30 €/h	1.500€

Tabla 3. Estimación de capital humano

4.5.4 COSTE TOTAL ESTIMADO

A continuación, se muestra un resumen económico de los costes asociados al desarrollo del proyecto.

CONCEPTO	*COSTE*
Software utilizado	0 €
Ordenador portátil	165 €
Monitor auxiliar	38 €
Desarrollo del proyecto	8.100 €
Supervisión y dirección	1.500 €
TOTAL	9.803 €

Tabla 4. Resumen económico del proyecto

4.5.5 COSTES OPERATIVOS DEL SISTEMA

En un escenario real de implantación empresarial, el sistema desarrollado podría generar una serie de costes operativos asociados a su mantenimiento y funcionamiento continuo. Entre ellos destacan los costes de infraestructura computacional, almacenamiento de datos, actualización de modelos de Machine Learning y supervisión técnica del sistema.

En caso de desplegarse en entornos cloud, también deberían considerarse gastos derivados del uso de servidores, procesamiento de datos en tiempo real y escalabilidad de la infraestructura. Además, el sistema requeriría tareas periódicas de mantenimiento para adaptar los modelos a nuevas amenazas y garantizar la precisión de la detección frente a ataques emergentes.

Aunque el presente proyecto se ha desarrollado en un entorno académico y experimental, la utilización de herramientas open source permite reducir considerablemente los costes operativos frente a soluciones comerciales tradicionales de ciberseguridad.

4.5.6 POSIBLE MODELO DE MONETIZACIÓN

El sistema desarrollado presenta potencial de aplicación en entornos empresariales, especialmente en pequeñas y medianas empresas que no disponen de soluciones avanzadas de detección de intrusiones basadas en inteligencia artificial.

Una posible estrategia de monetización consistiría en ofrecer el sistema como una solución Software as a Service (SaaS), mediante suscripción mensual, incluyendo funcionalidades de monitorización, detección automática de amenazas y visualización de resultados en tiempo real.

Asimismo, podrían ofrecerse diferentes niveles de servicio en función del tamaño de la empresa, volumen de tráfico monitorizado características avanzadas de análisis y generación de alertas. Otra posible vía de explotación sería la integración del sistema como complemento de plataformas de ciberseguridad ya existentes o su adaptación personalizada para organizaciones concretas.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

5.1 ARQUITECTURA GENERAL DEL SISTEMA

El sistema desarrollado en este proyecto tiene como objetivo la detección inteligente de ciberataques en tráfico de red mediante técnicas de Machine Learning. Para ello, se ha diseñado una arquitectura basada en diferentes fases de procesamiento y análisis de datos, comenzando desde la carga y preparación del dataset Hasta la implementación final de un dashboard interactivo para la monitorización del tráfico de red.

La arquitectura propuesta con vida modelo supervisados y no supervisados junto con herramientas de visualización y monitorización, permitiéndonos solo entrenar y evaluar modelos de clasificación, sino también representar gráficamente los resultados y simular un entorno de detección en tiempo real.

El flujo general del sistema se divide en las siguientes etapas:

1. Carga y exploración del dataset
2. Limpieza y procesamiento de datos
3. Transformación y normalización de variables
4. Entrenamiento de modelos supervisados
5. Entrenamiento de modelos no supervisados
6. Evaluación y comparación de resultados
7. Visualización
8. Desarrollo del dashboard web
9. Monitorización y visualización en tiempo real
10. Sistema de alertas y notificaciones.

5.1.1 FLUJO GENERAL DEL SISTEMA

El funcionamiento global del sistema desarrollado puede representarse mediante el siguiente flujo de trabajo:

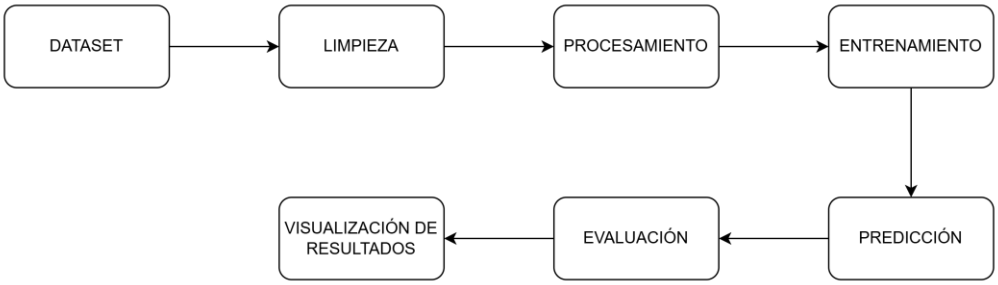


Figura 12. Diagrama del flujo de trabajo

En primer lugar, se carga el tráfico de red contenido en el dataset. Posteriormente, se ha realizado una fase de limpieza orientada a eliminar valores nulos, infinitos y registros inconsistentes.

A continuación, se han transformado las variables categóricas y las características numéricas normalizadas para facilitar el entrenamiento de modelos. Una vez procesados los datos, se entrenan diferentes algoritmos supervisados y no supervisados para detectar comportamientos anómalos y clasificar el tráfico de red.

Finalmente, los resultados obtenidos se han evaluado mediante distintas métricas y se han representado gráficamente para facilitar su interpretación.

5.2 PIPELINE DE PROCESAMIENTO DE DATOS

El pipeline del procesamiento es una de las partes fundamentales del sistema desarrollado, ya que la calidad del preprocesamiento influye directamente en el rendimiento de los modelos de Machine Learning.

5.3 *ESTUDIO Y SELECCIÓN DE DATASETS*

Se va a describir el estudio realizado comparando los distintos conjuntos de datos utilizados en investigación sobre detección de intrusiones. El objetivo de este análisis fue seleccionar el dataset más adecuado para el desarrollo experimental del proyecto, teniendo en cuenta factores como el realismo del tráfico de red, la variedad de ataques, el volumen de datos y la complejidad de las características disponibles.

5.3.1 KDD CUP-1999 – INTRUSION DETECTION DATASET

5.3.1.1 Descripción general del dataset

El dataset KDD Cup 1999 es uno de los conjuntos de datos clásicos más utilizados en investigación sobre detección de intrusiones en redes. Fue desarrollado a partir de tráfico de red simulador en un entorno militar y contiene tanto conexiones legítimas como distintos tipos de ataques informáticos.

Cada registro representa una conexión de red descrita mediante 41 características relacionadas con protocolos, servicios, estado de la conexión y métricas de tráfico. La última columna corresponde a la variable objetivo que clasifica cada conexión como tráfico normal o como uno de los distintos tipos de ataque.

Los ataques incluidos en el dataset se agrupan principalmente las siguientes categorías:

- DoS (Denial of Service)
- Probe
- R2L (Remote to Local)
- U2R (User to Root)

Este dataset se ha utilizado ampliamente como benchmark para evaluar modelos de Machine Learning aplicados a tareas y clasificación y detección de intrusiones. Sin embargo, presenta ciertas limitaciones relacionadas con el bajo realismo del tráfico y un elevado de desbalanceo entre las clases.

CARACTERISTICA	*VALOR*
Año de publicación	1999
Tipos de datos	Conexiones de red
Número de registros	494.021
Número de atributos	41 + etiqueta
Variables categóricas	4
Variables numéricas	38
Valores nulos	No
Valores infinitos	No
Tipos principales de ataque	DoS, Probe, R2L, U2R
Nivel de realismo	Bajo
Complejidad computacional	Baja

Tabla 5. Características generales del dataset KDD Cup 1999

5.3.1.2 Análisis exploratorio del dataset

Tras la carga del conjunto de datos, se realizó un análisis exploratorio inicial para estudiar la estructura de las variables, los tipos de datos y la distribución de las clases.

El dataset contiene un total de 42 columnas, de las cuales 41 corresponden a variables predictoras y una a la etiqueta de clasificación. Las variables se distribuyen de la siguiente forma:

- 23 variables enteras `(int64)`
- 15 variables continuas `(float64)`
- 4 variables categóricas `(string)`

Además, se comprobó que el conjunto de datos no presenta valores nulos de ninguna de las variables, lo que simplifica considerablemente el proceso de limpieza inicial.

TIPO DE VARIABLE	*CANTIDAD*
Variables enteras `(int64)`	23
Variables continuas `(float64)`	15
Variables categóricas `(str)`	4
Total, variables predictoras	41

Tabla 6. Distribución de tipos de variables en KDD Cup 1999

El análisis estadístico mediante `df.describe()` permitió identificar importantes diferencias de escala entre variables. Algunas características presentan valores muy pequeños mientras que otras alcanzan magnitudes considerablemente elevadas, lo que evidencia la necesidad de aplicar técnicas de normalización antes del entrenamiento de los modelos.

Además, se observó una elevada dispersión de múltiples variables, así como la presencia de valores atípicos (outliers), especialmente en métricas relacionadas con tráfico y número de conexiones.

MÉTRICA ANALIZADA	*OBSERVACIÓN*
Media `(mean)`	Variables en escalas muy diferentes
Desviación típica `(std)`	Alta variabilidad en algunas características

Valores máximos `(max)`	Presencia de posibles outliers
Percentiles (25%, 50%, 75%)	Muchas variables concentradas en 0
Distribución general	Datos altamente dispersos

Tabla 7. Principales observaciones del análisis estadístico

5.3.1.3 Distribución de clases

Uno de los aspectos más relevantes observados durante el análisis del dataset es el fuerte desbalanceo entre clases.

Las clases correspondientes a los ataques smurf y neptune representan aproximadamente el 78% del conjunto de datos, mientras que el tráfico normal supone únicamente cerca del 20%. El resto de los ataques aparecen con frecuencias muy reducidas.

Este comportamiento es especialmente importante desde el punto de vista de Machine Learning, ya que puede provocar que los modelos aprendan patrones sesgados hacia las clases mayoritarias, reduciendo su capacidad de detección sobre ataques menos frecuentes.

CLASE	*NÚMERO DE REGISTROS*	*PORCENTAJE APROXIMADO*
smurf	~ 280000	~57%
neptune	~107000	~22%
normal	~97000	~20%
back	~2200	<1%
satan	~1500	<1%

ipsweep	~1200	<1%
portsweep	~1000	<1%

Tabla 8. Clases más frecuentes en KDD Cup 1999

5.3.1.4 Limitaciones del dataset

A pesar de su relevancia histórica, KDD Cup 1999 presenta importantes limitaciones que dificultan su utilización en escenarios modernos de detección de intrusiones:

- Elevado desbalanceo entre clases
- Existencia de numerosos registros redundantes y duplicados
- Bajo nivel de realismo respecto a redes actuales
- Tráfico generado en entornos simulados
- Escasa representación de amenazas modernas

Por estas razones, aunque el dataset resulta útil para realizar estudios iniciales y comparar algoritmos clásicos, no representa adecuadamente las características del tráfico de red contemporáneo.

5.3.2 NSL-KDD

5.3.2.1 Descripción general del dataset

El dataset NSL-KDD surge como una versión mejorada del conocido KDD Cup 1999, desarrollado específicamente para corregir algunas de las principales limitaciones presentes en dicho conjunto de datos. Su objetivo principal es proporcionar un entorno más equilibrado y adecuado para la investigación y evolución de modelos de detección de intrusiones basados en Machine Learning.

Al igual que KDD Cup 1999, cada registro del dataset representa una conexión de red descrita mediante 41 características relacionadas con protocolos, servicios, estados de

conexión y métricas de tráfico. La última columna corresponde a la etiqueta de clasificación, indicando si el tráfico es normal o pertenece a alguna categoría de ataque.

Entre los principales ataques incluidos en el dataset destacan:

- DoS (Denial of Service)
- Probe
- R2L (Remote to Local)
- U2R (User to Root)

Una de las principales ventajas de NSL-KDD es que incorpora conjuntos separados de entrenamiento y prueba, permitiendo evaluar la capacidad de generalización de los modelos frente a tráfico no visto previamente.

CARACTERISTICA	*VALOR*
Año de publicación	2009
Tipos de datos	Conexiones de red
Número de registros (Train)	125.973
Número de registros (Test)	22.544
Número de atributos	41 + etiqueta
Variables categóricas	4
Variables numéricas	39
Valores nulos	No
Valores infinitos	No

Separación Train/Test	Sí
Nivel de realismo	Medio

Tabla 9. Características generales del dataset NSL-KDD

5.3.2.2 Mejoras respecto a KDD Cup 1999

El dataset NSL-KDD fue diseñado para solucionar varios de los problemas estructurales presentes en KDD Cup 1999, especialmente aquellos relacionados con el aprendizaje sesgado de los modelos.

Entre las principales mejoras introducidas están:

- Eliminación de registros duplicados
- Reducción parcial del desbalanceo entre clases
- Evaluación más justa de los algoritmos
- Mayor capacidad de generalización de los modelos

Estas modificaciones convierten a NSL-KDD en un benchmark más adecuado para comparar algoritmos de clasificación en tareas de detección de intrusiones.

PROBLEMA EN KDD99	*MEJORA INTRODUCIDA EN NSL-KDD*
Gran cantidad de duplicados	Eliminación de registros redundantes
Modelos sobreajustados	Mejor capacidad de generalización
Alto sesgo entre clases	Distribución más equilibrada
Evaluación poco realista	Separación train/test más robusto

Tabla 10. Principales mejoras de NSL-KDD respecto a KDD99

5.3.2.3 Análisis exploratorio del dataset

Tras la carga de los conjuntos KDDTrain+ y KDDTest+, Se realizó un análisis exploratorio de las variables y de la estructura general de los datos.

El conjunto de entrenamiento contiene 125973 registros y 43 columnas, mientras que el conjunto de prueba contiene 22544 registros. Una de las columnas adicionales corresponde al nivel de dificultad de clasificación de cada instancia, aunque esta variable no se ha utilizado durante el entrenamiento los módulos.

El análisis de tipo de datos muestra la siguiente distribución:

- 24 variables enteras `(int64)`
- 15 variables continuas `(float64)`
- 4 variables categóricas `(string)`

Además, se verificó que todas las variables presentan valores válidos y que no existen nulos en el dataset.

TIPO DE VARIABLE	*CANTIDAD*
Variables enteras `(int64)`	23
Variables continuas `(float64)`	15
Variables categóricas `(str)`	4
Total, variables predictoras	41

Tabla 11. Distribución de variables en NSL-KDD

El análisis estadístico mediante `df.describe()` permitió Observar diferencias significativas de escala entre variables, así como una elevada dispersión en múltiples características.

Muchas variables presentan una gran concentración de valores cercanos a cero, mientras que otras contienen valores máximos muy elevados, indicando la posible presencia de outliers y distribuciones altamente asimétricas.

MÉTRICA ANALIZADA	*OBSERVACIÓN*
Media `(mean)`	Variables en escalas muy diferentes
Desviación típica `(std)`	Alta dispersión en varios atributos
Valores máximos `(max)`	Presencia de posibles outliers
Percentiles (25%, 50%, 75%)	Muchas variables concentradas en 0
Distribución general	Datos dispersos y desbalanceados

Tabla 12. Observaciones del análisis estadístico de NSL-KDD

5.3.2.4 Distribución de clases

El análisis de la variable objetivo permitió identificar nuevamente un importante desbalance entre clases, aunque menos notorio que en KDD Cup 1999.

En el conjunto de entrenamiento, el tráfico normal representa aproximadamente el 53% de los registros, mientras que ataques como neptune constituyen cerca del 33%. El resto de las categorías aparecen con frecuencias considerablemente menores.

Además, El conjunto de prueba incorpora algunos ataques que no están presentes en entrenamiento, permitiendo así evaluar la capacidad del modelo para enfrentarse a amenazas no vistas previamente y simulando un escenario más cercano a entornos reales de ciberseguridad.

CLASE	*PORCENTAJE APROXIMADO*

normal	~53%
neptune	~33%
satan	~3%
ipsweep	~2%
portsweep	~2%
Otros ataques	<1%

Tabla 13. Clases más frecuentes en NSL-KDD (Train)

5.3.2.5 Limitaciones del dataset

A pesar representar una mejora significativa respecto a KDD Cup 1999, el dataset NSL-KDD continúa teniendo ciertas limitaciones:

- Tráfico generado en entornos simulados
- Escasa representación de amenazas modernas
- Número reducido de características respecto a datasets actuales
- Persistencia de desbalanceo
- Nivel de realismo inferior al de datasets recientes basados en flujos de red

Por estas razones, aunque NSL-KDD sigue siendo ampliamente utilizado en investigación académica, los datasets más recientes ofrecen escenarios considerablemente más complejos y representativos de redes corporativos modernas.

5.3.3 CICIDS2017

5.3.3.1 Descripción general del dataset

El dataset CICIDS2017 constituye uno de los conjuntos de datos modernos más utilizados en investigación sobre detección de intrusiones en redes. Fue desarrollado por el *Canadian*

Institute for Cybersecurity con el objetivo de proporcionar un entorno de tráfico mucho más realista que los datasets clásicos utilizados tradicionalmente en ciberseguridad.

A diferencia de KDD Cup 1999 o NSL-KDD, este conjunto de datos se basa en tráfico de red generado en un entorno corporativo controlado que simula el comportamiento real de usuarios y servicios dentro de una infraestructura empresarial.

Cada registro del dataset representa un flujo de red completo y no una conexión individual, lo que supone un enfoque más actual en sistemas de detección de intrusiones. Los flujos de red fueron extraídos automáticamente mediante la herramienta CICFlowMeter, incorporando un elevado número de características relacionadas con:

- Duración del flujo
- Número de paquetes enviados y recibidos
- Estadísticas de tamaño de paquetes
- Tasas de transmisión
- Métricas temporales
- Medidas estadísticas del comportamiento del tráfico

La variable objetivo se encuentra en la columna Label, encargada de clasificar cada flujo cómo tráfico benigno o como uno de los distintos tipos de ataque.

Entre las amenazas incluidos destacan:

- Ataques DoS y DDoS
- Escaneos de red (PortScan)
- Ataques de fuerza bruta
- Inyecciones web
- Botnets
- Otros ataques representativos de amenazas actuales.

Gracias a estas características, CICIDS2017 presenta un nivel de realismo y complejidad considerablemente superior al de datasets clásicos, convirtiéndose en uno de los benchmarks

más utilizados actualmente en investigaciones sobre Machine Learning aplicado a la ciberseguridad.

CARACTERISTICA	*VALOR*
Año de publicación	2017
Tipos de datos	Flujos de red
Número de registros	1.894.275
Número de atributos	78 + etiqueta
Variables categóricas	1
Variables numéricas	78
Valores nulos	Sí
Valores infinitos	Sí
Nivel de realismo	Alto
Complejidad computacional	Alta

Tabla 14. Características generales del dataset CICIDS2017

5.3.3.2 Análisis exploratorio del dataset

Tras la carga del dataset, se realizó un análisis exploratorio para estudiar la estructura de las variables, el tipo de datos y las posibles inconsistencias presentes en el conjunto.

El dataset contiene un total de 79 columnas, de las cuales 78 corresponden a variables proyectoras y una a la etiqueta de clasificación. La distribución de tipos de datos es la siguiente:

- 54 variables enteras `(int64)`
- 24 variables continuas `(float64)`
- 1 variables categóricas `(string)`

Además, se observó que muchos nombres de columnas presentan espacios y formatos extensos, lo que requirió una normalización posterior durante la fase de procesado.

TIPO DE VARIABLE	*CANTIDAD*
Variables enteras `(int64)`	54
Variables continuas `(float64)`	24
Variables categóricas `(str)`	1
Total, variables predictoras	78

Tabla 15. Distribución de variables en CICIDS2017

Uno de los aspectos más relevantes destacados durante el análisis exploratorio fue la presencia de valores nulos e infinitos en determinadas variables estadísticas.

En total, el dataset contiene:

- 1277 valores nulos
- 2787 valores infinitos

Aunque estas cantidades representan un porcentaje reducido respecto al volumen de los datos, su tratamiento resulta imprescindible, ya que los modelos de Machine Learning no pueden trabajar correctamente con valores infinitos e indefinidos.

PROBLEMA DETECTADO	*CANTIDAD*

Valores nulos `(NaN)`	1277
Valores infinitos `(Inf)`	2787
Columnas con nombres extensos	Si
Variables altamente desbalanceadas	Si

Tabla 16. Problemas detectados en CICIDS2017

5.3.3.3 Distribución de clases

El análisis de la variable objetivo permitió identificar un fuerte desbalanceo entre clases, situación habitual en datasets de ciberseguridad.

La clase correspondiente al tráfico Benigno representa aproximadamente el 80% del conjunto de datos, mientras que el resto de los ataques constituyen cerca del 20% restante. Algunos ataques presentan un número muy elevado de registros, Dos Hulk o PortScan, Mientras que otros aparecen con frecuencias extremadamente reducidas.

Este comportamiento supone un desafío importante para los modelos de clasificación, Ya que puede provocar un aprendizaje sesgado hacia las clases mayoritarias.

CLASE	*NÚMERO DE REGISTROS*	*PORCENTAJE APROXIMADO*
BENIGN	~ 1.500.000	~80%
Dos Hulk	~230.000	~12%
PortScan	~160.000	~8%
DDoS	~128.000	~7%

FTP-Patator	~8.000	<1%
SSH-Patator	~6.000	<1%
DoS slowloris	~5.000	<1%
Heartbleed	11	Muy bajo

Tabla 17. Clases más frecuentes en CICIDS2017

5.3.3.4 Ventajas y limitaciones del dataset

El dataset CICIDS2017 presenta importantes ventajas respecto a conjuntos clásicos como KDD Cup 1999 o NSL-KDD:

- Tráfico mucho más realista
- Ataques modernos
- Mayor número de características
- Representación mediante flujos de red
- Escenarios cercanos a redes corporativas reales

Sin embargo, también introduce nuevos desafíos relacionados con el volumen de los datos, la presencia de valores nulos e infinitos, el fuerte desbalanceo de clases y la alta complejidad computacional necesaria para el entrenamiento de modelos.

Estas características convierten a este dataset en un entorno mucho más representativo es de problemas reales de ciberseguridad, aunque también significativamente más exigente desde el punto de vista del procesamiento y análisis de datos.

5.3.4 CSE-CIC-IDS2018

5.3.4.1 Descripción general del dataset

El dataset CSE-CIC-IDS2018 Representa una evolución directa del CICIDS2017 y constituye 1 de los conjuntos de datos más avanzados y realistas disponibles actualmente para investigación en detección de intrusiones mediante Machine Learning.

Este dataset fue desarrollado conjuntamente por el *Canadian Institute for Cybersecurity (CIC) y el Communications Security Establishment (CSE)* de Canadá, Con el objetivo de proporcionar un entorno experimental capaz de reproducir escenarios de red corporativos mucho más cercanos a situaciones reales.

A diferencia de datasets clásicos basados en conexiones individuales, CSE-CIC-IDS2018 trabaja con flujos completos de red, incorporando una gran variedad de características extraídas automáticamente mediante la herramienta CICFlowMeter.

Las variables incluidas describen aspectos relacionados con duración de los flujos, número de paquetes enviados y recibidos, métricas de transmisión, estadísticas de tamaña de paquetes, comportamiento temporal del tráfico y características estadísticas avanzadas de la comunicación en red.

La variable objetivo clasifica cada flujo como tráfico Benigno como 1 de los distintos tipos de ataque incluidos en el dataset.

Entre las amenazas presentes destacan:

- Ataques Dos y DDoS
- Ataques de fuerza bruta
- Ataques de aplicaciones web
- Escaneos y reconocimiento de red
- Actividades maliciosas avanzadas

Gracias a su elevado nivel de realismo, su gran volumen de datos y la complejidad de sus características, este dataset se ha consolidado como uno de los benchmarks más utilizados en investigaciones recientes sobre sistemas IDS basados en aprendizaje automático.

CARACTERISTICA	*VALOR*
Año de publicación	2018
Tipos de datos	Flujos de red
Número de registros	1.048.575
Número de atributos	79 + etiqueta
Variables categóricas	2
Variables numéricas	78
Valores nulos	Sí
Valores infinitos	Sí
Nivel de realismo	Muy alto
Complejidad computacional	Muy alta

Tabla 18. Características generales del dataset CSE-CIC-IDS2018

5.3.4.2 Análisis exploratorio del dataset

Tras la carga del conjunto de datos, se realizó un análisis exploratorio para estudiar la estructura de las variables, La distribución de clases y los posibles problemas presentes en el dataset.

El conjunto tiene un total de 80 columnas, de las cuales 79 corresponden a variables productoras y una a la variable objetivo. La distribución de tipos de datos es la siguiente:

- 54 variables enteras `(int64)`
- 24 variables continuas `(float64)`
- 2 variables categóricas `(string)`

Además, se detectó la presencia de valores nulos en determinadas métricas asociadas principalmente a tasas de transmisión y estadísticas temporales.

TIPO DE VARIABLE	*CANTIDAD*
Variables enteras `(int64)`	54
Variables continuas `(float64)`	24
Variables categóricas `(str)`	2
Total, variables predictoras	79

Tabla 19. Distribución de variables en CSE-CIC-IDS2018

Otro aspecto relevante observado durante el análisis fue la complejidad estructural del dataset. El gran número de características, junto con la elevada dimensionalidad y el tamaño del conjunto de datos, supone un escenario más exigente desde el punto de vista computacional.

Además, se identificaron problemas habituales en entornos reales de ciberseguridad como:

- Valores nulos
- Presencia de datos desbalanceados
- Características altamente dispersas
- Métricas con escalas muy diferentes

Estas condiciones hacen necesario aplicar técnicas de limpieza normalización y procesado antes de un entrenamiento en los modelos.

PROBLEMA DETECTADO	*PRESENCIA*
Valores nulos `(NaN)`	Sí
Valores infinitos `(Inf)`	Sí
Datos desbalanceados	Si
Variables en distintas escalas	Si
Alta dimensionalidad	Si

Tabla 20. Problemas detectados en CSE-CIC-IDS2018

5.3.4.3 Distribución de clases

El análisis de la variable objetivo muestra que el tráfico Benigno representa aproximadamente el 64% del conjunto de datos, mientras que los ataques constituyen cerca del 36% restante.

Aunque sigue existiendo desbalance entre clases, resulta más moderado que en datasets anteriores como KDD Cup 1999 o CICIDS2017, Proporcionando un escenario más equilibrado para el entrenamiento y evaluación de modelos de clasificación.

Además, el dataset incorpora una mayor variedad de amenazas modernas y escenarios de ataque complejos, que permiten analizar el comportamiento de los algoritmos frente a tráfico considerablemente más realista.

TIPO DE TRÁFICO	*PORCENTAJE APROXIMAADO*

Tráfico benigno	~64%
Tráfico malicioso	~36%

Tabla 21. Distribución aproximada de clases en CSE-CIC-IDS2018

5.3.4.4 Justificación de la selección del dataset

Tras el análisis comparativo de los distintos conjuntos de datos estudiados, se seleccionó el dataset CSE-CIC-IDS2018 para el desarrollo experimental del proyecto.

La elección de este dataset se fundamenta principalmente en los siguientes aspectos:

- Mayor nivel de realismo respecto a redes corporativas y actuales
- Presencia de amenazas modernas y escenarios complejos
- Mayor volumen y diversidad de datos
- Representación mediante flujos completos de red
- Mayor complejidad estructural y computacional
- Escenario más adecuado para evaluar modelos avanzados de Machine Learning

A diferencia de datasets como KDD Cup 1999 o NSL-KDD, CSE-CIC-IDS2018 Reproduce situaciones mucho más cercanas en torno a reales y ciberseguridad, incorporando además problemas habituales presentes en sistemas reales, como tráfico desbalanceado, valores nulos y características altamente variables.

Por todo ello, este dataset se consideró el más adecuado para entrenar, evaluar y comparar los modelos implementados en el proyecto, permitiendo analizar su comportamiento bajo condiciones más exigentes y representativas de escenarios reales.

5.4 CARGA DEL DATASET

Para el desarrollo experimental se ha utilizado el dataset CSE-CIC-IDS2018, compuesto por tráfico de red legítimo y varios tipos de ataques representativos de entornos empresariales reales.

Los archivos CSV se han cargado utilizando la librería Pandas, pudiendo así trabajar de forma estructurada mediante DataFrames.

```
ruta_csv = "../raw/*.csv"
archivos = glob.glob(ruta_csv)

for archivo in archivos:
    df_temp = pd.read_csv(archivo)
    df_temp.columns = df_temp.columns.str.strip()
    lista_df.append(df_temp)

df = pd.concat(lista_df, ignore_index=True)
```

Tras la carga inicial, se ha realizado una inspección de la estructura del conjunto de datos, analizando el número de registros, variables disponibles y tipos de datos presentes.

5.4.1 LIMPIEZA Y TRATAMIENTO DE DATOS

Una vez cargado el dataset, se ha realizado un análisis preliminar para detectar posibles problemas presentes en los datos.

```
df["Label"] = df["Label"].str.strip()
df = df[df["Label"] != "Label"]
columnas_sobrantes = ["Flow ID", "Src IP", "Dst IP", "Timestamp"]
df = df.drop(columns=columnas_sobrantes, errors="ignore")
```

Durante esa etapa se han identificado valores nulos, valores infinitos, valores redundantes y características con distribuciones altamente dispersas. Los registros que contenían valores inconsistentes han sido eliminados o transformados con el objetivo de garantizar la estabilidad de los algoritmos. Además, se han eliminado columnas no relevantes para el entrenamiento de modelos ya que no aportaban información útil para la clasificación.

5.4.2 ELIMINACIÓN DE VALORES NULOS E INFINITOS

Este procedimiento ha permitido obtener un conjunto de datos más consistente y adecuado para las fases posteriores de entrenamiento y evaluación.

```
df = df.replace([np.inf, -np.inf], np.nan)
nulos_antes = df.isna().sum().sum()
df = df.dropna()
```

5.4.3 NORMALIZACIÓN DE VARIABLES

Las características numéricas en el dataset tienen escalas muy diferentes entre sí. Algunas variables contienen valores extremadamente elevados, mientras que otras tienen rangos considerablemente menores.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Para evitar que determinadas variables dominaran el entrenamiento de los modelos, se ha aplicado un proceso de normalización utilizando StandardScaler. La normalización ha permitido centrar las variables alrededor de media cero y desviación típica unitaria, mejorando la estabilidad y el rendimiento de los algoritmos.

5.4.4 DIVISION TRAIN/TEST

Una vez completado el preprocesamiento, el conjunto de datos se ha dividido en subconjuntos de entrenamiento y prueba.

La división utilizada ha sido de:

- 80% para entrenamiento
- 20% para evaluación

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
```

```
    random_state=42,
    stratify=y
)
```

Esta separación ha permitido entrenar los modelos utilizando una parte del dataset y posteriormente evaluar su capacidad de generalización sobre los datos no vistos previamente. A continuación, con el objetivo de optimizar el flujo de trabajo y evitar repetir las fases de procesamiento en cada ejecución, los conjuntos de entrenamiento y prueba han sido almacenados localmente junto con el scaler utilizado durante la normalización.

```
X_train_final.to_csv("../procesados/X_train.csv", index=False)
X_test_final.to_csv("../procesados/X_test.csv", index=False)
y_train.to_csv("../procesados/y_train.csv", index=False)
y_test.to_csv("../procesados/y_test.csv", index=False)
joblib.dump(scaler, "../procesados/scaler.pkl")
```

5.5 *IMPLEMENTACIÓN DE MODELOS SUPERVISADOS*

Con el objetivo de detectar y clasificar distintos tipos de ciberataques presentes en el tráfico de red, se han implementado varios modelos supervisados. Todos los algoritmos han sido entrenados utilizando el mismo conjunto de datos procesados, permitiendo así realizar una comparación de su comportamiento y rendimiento.

Los algoritmos se han seleccionado buscando combinar modelos sencillos e interpretables con otros más complejos y robustos, permitiendo analizar diferentes enfoques de clasificación sobre un mismo problema.

5.5.1 REGRESIÓN LOGÍSTICA

la regresión logística se ha utilizado como modelo base de clasificación supervisada debido a su simplicidad y gran velocidad de entrenamiento. Este algoritmo permite establecer una primera aproximación al problema de clasificación, es especialmente útil para comparar el comportamiento de modelos más complejos.

```
lr = LogisticRegression(
    max_iter=500,
    random_state=42,
    solver="lbfgs",
```

```
    n_jobs=-1
)
lr.fit(X_train, y_train)
```

A pesar de ser un modelo lineal, ofrece buenos resultados en tareas de clasificación binaria cuando los datos presentan patrones relativamente separables. El entrenamiento del modelo es realizado utilizando los datos previamente normalizados.

5.5.2 ÁRBOL DE DECISIÓN

El modelo basado en árboles de decisión ha sido implementado por su capacidad para generar reglas jerárquicas de clasificación fácilmente interpretables. Este algoritmo divide progresivamente el espacio de características utilizando diferentes condiciones lógicas, permitiendo separar las distintas clases presentes en el dataset. Una de las principales ventajas de este modelo es la facilidad para interpretar el proceso de decisión seguido durante la clasificación del tráfico de red.

```
dt = DecisionTreeClassifier(
    max_depth=10,
    random_state=42
)
dt.fit(X_train, y_train)
```

5.5.3 RANDOM FOREST

Random Forest ha sido uno de los modelos implementados más importantes durante el desarrollo del proyecto debido a su robustez y capacidad de generalización. Este algoritmo combina múltiples árboles de decisión entrenados sobre subconjuntos aleatorios del dataset, reduciendo el riesgo de sobreajuste y mejorando la estabilidad del modelo final. Además, Random Forest permite calcular la importancia relativa de las variables utilizadas durante el proceso de clasificación, facilitando la interpretación del comportamiento del modelo.

```
rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=20,
    n_jobs=-1,
    random_state=42
```

```
)
rf.fit(X_train, y_train)
```

5.5.4 GRADIENT BOOSTING

Este modelo ha sido implementado como alternativa avanzada basada en técnicas de boosting. Este algoritmo entrena múltiples modelos de forma secuencial, corrigiendo progresivamente los errores cometidos por las iteraciones anteriores. Su funcionamiento permite construir modelos altamente precisos incluso en datasets complejos y de gran tamaño como el utilizado en este proyecto.

```
hgb = HistGradientBoostingClassifier(
    max_iter=100,
    max_depth=10,
    random_state=42,
    verbose=1
)
hgb.fit(X_train, y_train)
```

5.5.5 COMPARACIÓN INICIAL DE MODELOS SUPERVISADOS

Una vez implementados todos los modelos supervisados, se ha realizado una primera comparación preliminar orientada a analizar diferencias de comportamiento, complejidad y capacidad de generalización.

Durante esta fase se ha observado que los modelos basados en ensamblado, especialmente Random Forest y HistGradientBoosting, ofrecen un comportamiento más robusto frente a la complejidad del tráfico de red. Por otro lado, la regresión logística es un modelo más sencillo con una elevada velocidad entrenamiento, aunque con más limitaciones frente a relaciones no lineales complejas.

5.6 IMPLEMENTACIÓN DE MODELOS NO SUPERVISADOS

Además de los algoritmos supervisados utilizados para clasificar el tráfico de red, se han implementado técnicas de aprendizaje no supervisado orientadas a detectar patrones ocultos y agrupaciones presentes en los datos sin necesidad de utilizar etiquetas previas.

Este tipo de algoritmos resultan especialmente útiles en entornos de ciberseguridad, donde pueden existir ataques de desconocidos o comportamientos anómalos que no hayan sido previamente identificados.

5.6.1 PCA

La técnica PCA ha sido utilizada para reducir la dimensionalidad del dataset y facilitar la visualización de los datos en espacios bidimensionales. El dataset empleado tiene un número de características muy grande, lo que dificulta su representación gráfica directa. PCA permite transformar las variables originales en un conjunto reducido de componentes principales que conservan gran parte de la información presente en los datos.

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_test)
```

5.6.2 K-MEANS

el algoritmo K-Means ha sido implementado con el objetivo de agrupar automáticamente registros similares del tráfico de red utilizando técnicas de clustering. Este algoritmo divide los datos en distintos grupos o clusters basándose en la similitud entre registros, sin utilizar información previa sobre las etiquetas reales del tráfico. Para facilitar el proceso de agrupamiento y mejorar la representación visual el algoritmo ha sido aplicado sobre los datos previamente reducidos mediante PCA.

```
n_clusters = 5
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=10)
X_sample = X_test.sample(50000, random_state=42)
clusters = kmeans.predict(X_sample)
```

5.7 PERSISTENCIA Y ALMACENAMIENTO DE MODELOS

Con el objetivo de optimizar el flujo de trabajo y evitar repetir continuamente el entrenamiento los modelos, todos los algoritmos entrenados se han almacenado localmente utilizando la librería Joblib. Este procedimiento permite reutilizar posteriormente los

modelos ya entrenados durante las fases de evaluación, comparación y visualización de resultados.

5.8 *DESARROLLO DEL DASHBOARD WEB*

Con el objetivo de complementar el sistema de detección desarrollado y facilitar la interpretación de los resultados obtenidos, se ha implementado un dashboard web interactivo orientado a la monitorización y análisis del tráfico de red.

La aplicación web permite cargar conjuntos de datos en formato CSV, procesarlos automáticamente mediante el modelo entrenado y visualizar de forma gráfica el comportamiento del tráfico analizado. Además, el sistema incorpora funcionalidades de monitorización en tiempo real, representación estadística de resultados y generación de alertas automáticas ante posibles situaciones de riesgo.

El dashboard ha sido desarrollado utilizando la librería Streamlit de Python, lo que permite crear interfaces web interactivas de forma sencilla e integrarlas directamente con los modelos de Machine Learning entrenados durante el proyecto.

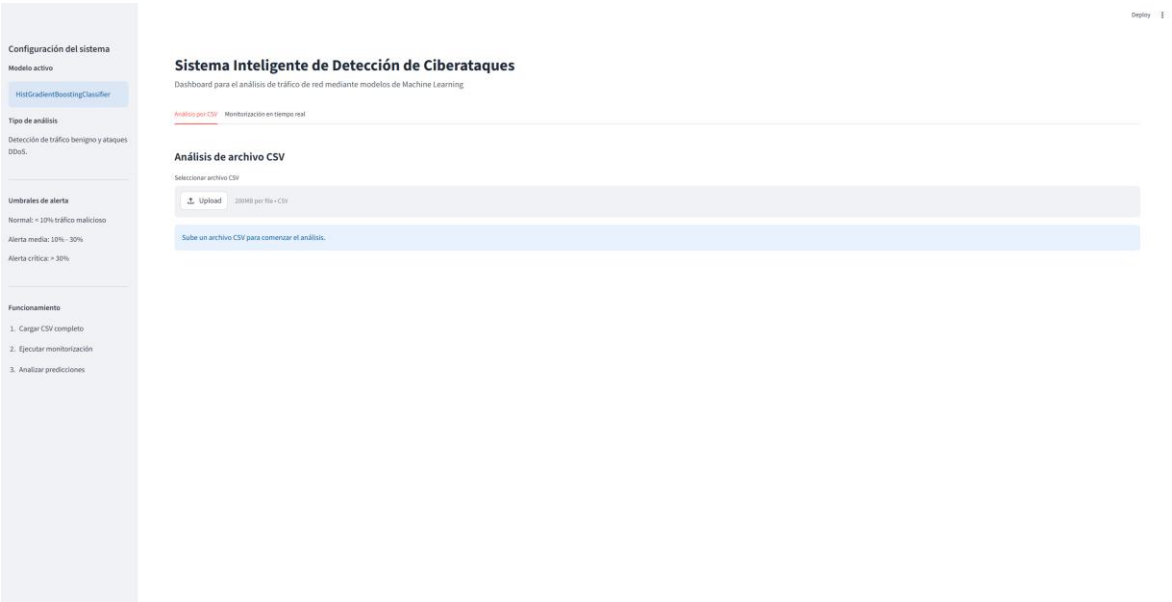


Figura 13. Visualización general de la web

5.8.1 ARQUITECTURA DE LA APLICACIÓN WEB

La arquitectura de la aplicación web se basa en una estructura modular compuesta por distintas etapas de procesamiento y visualización. En primer lugar, la aplicación carga automáticamente el modelo HistGradientBoosting Seleccionado como modelo final tras la comparación experimental, junto con el scaler utilizado durante la fase de normalización de datos. Posteriormente, el usuario puede cargar un archivo CSV con tráfico de red para su análisis.

Una vez cargado el dataset, el sistema realiza automáticamente el preprocesamiento de los datos y ejecuta las predicciones utilizando el modelo de clasificación seleccionado. Los resultados obtenidos se almacenan temporalmente y se representan mediante diferentes elementos gráficos e indicadores estadísticos dentro del dashboard.

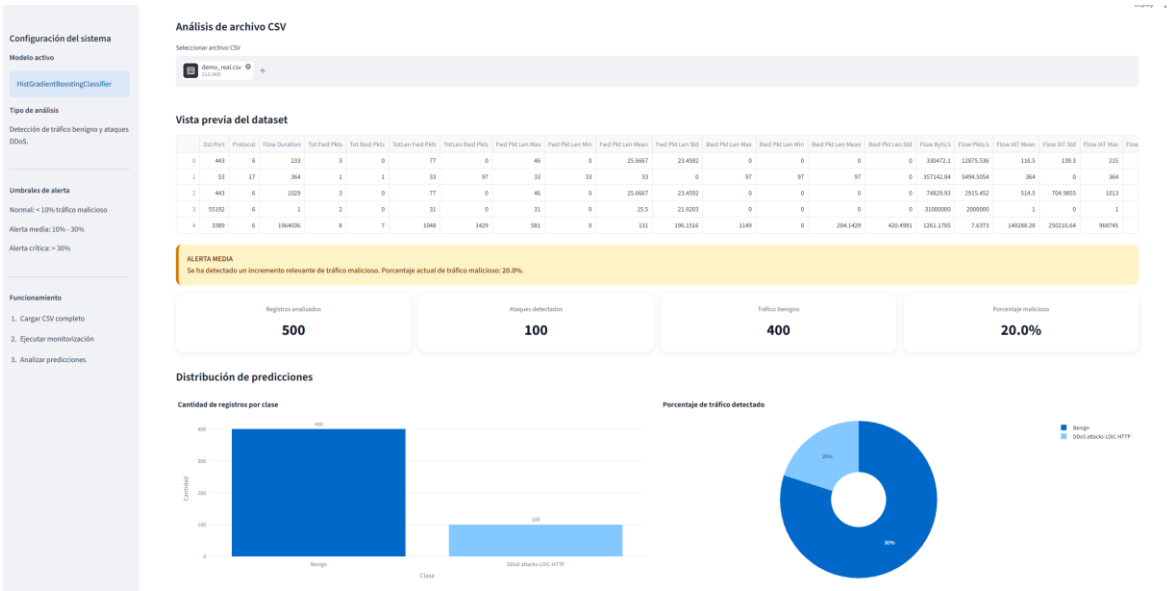


Figura 14. Análisis con archivo CSV

La arquitectura implementada permite además incorporar simulaciones de tráfico en tiempo real mediante actualizaciones automáticas de los datos procesados.

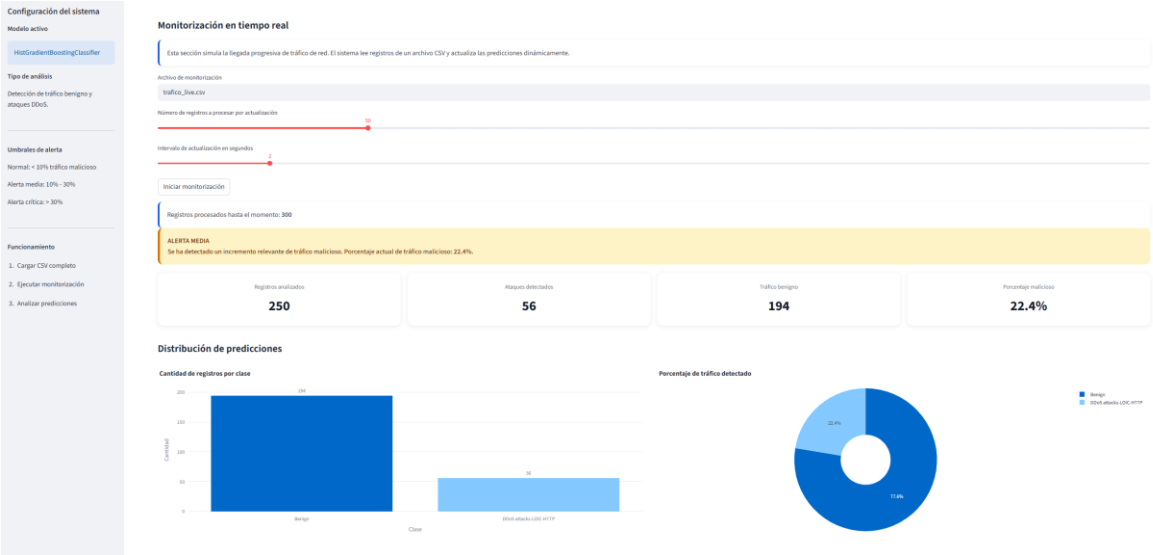


Figura 15. Monitorización en tiempo real

5.8.2 VISUALIZACIÓN DE MÉTRICAS Y RESULTADOS

El dashboard desarrollado incorpora diferentes componentes gráficos destinados a facilitar la interpretación de los resultados obtenidos durante el análisis del tráfico de red.

Entre las principales visualizaciones implementadas destacan:

- Métricas generales del análisis realizado
- Número total de registros procesados
- Cantidad de tráfico Benigno detectado
- Número de ataques identificados
- Porcentaje de tráfico malicioso
- Gráfico de barras para comparar la distribución de predicciones
- Gráficas circulares para representar porcentajes de tráfico Benigno y malicioso
- Tablas dinámicas con los resultados generados por el modelo

Estas representaciones permiten interpretar rápidamente el comportamiento del sistema y facilitan la detección visual de posibles anomalías en el tráfico analizado.

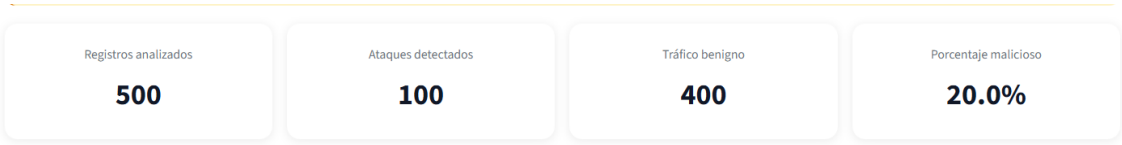


Figura 16. Indicadores estadísticos mostrados por el dashboard web tras el análisis del tráfico de red mediante análisis de CSV

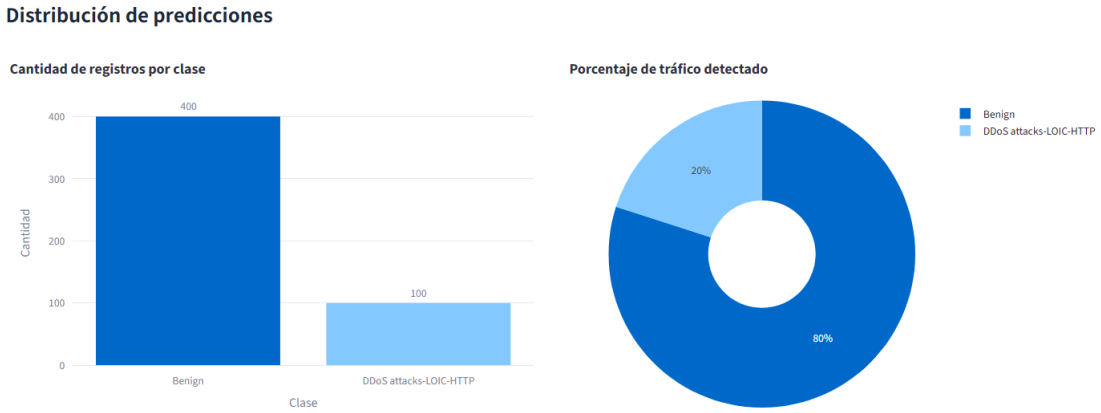
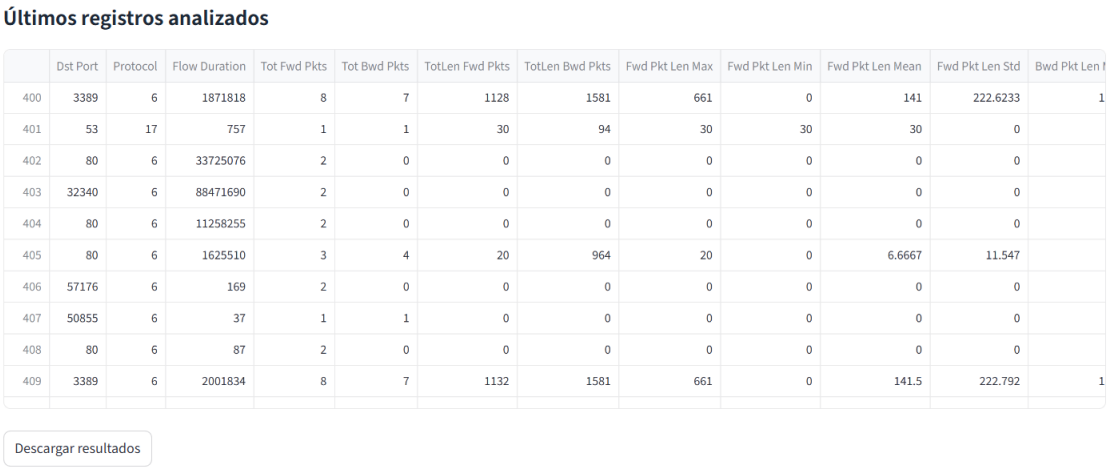


Figura 17. Representación gráfica de la distribución de tráfico benigno y tráfico malicioso detectado por el sistema mediante análisis por CSV



	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	Fwd Pkt Len Std	Bwd Pkt Len M
400	3389	6	1871818	8	7	1128	1581	661	0	141	222.6233	1
401	53	17	757	1	1	30	94	30	30	30	0	
402	80	6	33725076	2	0	0	0	0	0	0	0	
403	32340	6	88471690	2	0	0	0	0	0	0	0	
404	80	6	11258255	2	0	0	0	0	0	0	0	
405	80	6	1625510	3	4	20	964	20	0	6.6667	11.547	
406	57176	6	169	2	0	0	0	0	0	0	0	
407	50855	6	37	1	1	0	0	0	0	0	0	
408	80	6	87	2	0	0	0	0	0	0	0	
409	3389	6	2001834	8	7	1132	1581	661	0	141.5	222.792	1

Figura 18. Tabla de registros analizados mostrada por el dashboard durante la monitorización del tráfico de red mediante análisis por CSV

5.8.3 SISTEMA DE MONITORIZACIÓN EN TIEMPO REAL

Además del análisis estadístico mediante archivos CSV, el sistema incorpora una funcionalidad de monitorización en tiempo real basada en simulaciones de tráfico de red.

Esta funcionalidad permite actualizar automáticamente las predicciones realizadas por el modelo conforme se reciben nuevos registros de tráfico, simulando el comportamiento de un entorno de red real. Para ello, la aplicación procesa continuamente pequeños bloques de datos y se actualiza dinámicamente las gráficas y métricas mostradas en pantalla.

La monitorización en tiempo real permite observar la evolución del tráfico analizado y detectar rápidamente posibles incrementos de actividad maliciosa.

Figura 19. Indicadores estadísticos mostrados por el dashboard web tras el análisis del tráfico de red mediante monitorización en tiempo real

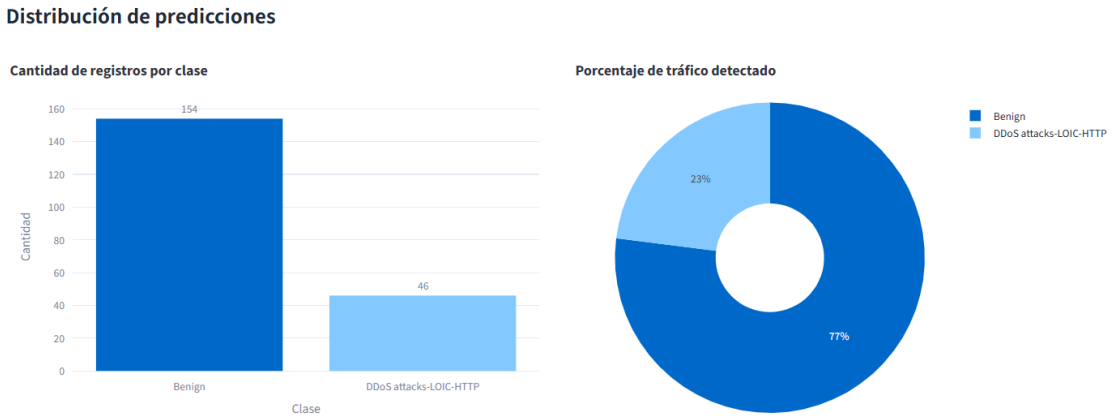


Figura 20. Representación gráfica de la distribución de tráfico benigno y tráfico malicioso detectado por el sistema mediante monitorización en tiempo real

Últimos registros analizados

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	Fwd Pkt Len Std	Bwd Pkt Len M
220	0	0	72626856	9	0	0	0	0	0	0	0	
221	53	17	289	1	1	51	67	51	51	51	0	
222	80	6	34581520	2	0	0	0	0	0	0	0	
223	53	17	108645	2	2	66	246	33	33	33	0	
224	53	17	317	1	1	41	73	41	41	41	0	
225	3389	6	1691904	8	7	1148	1581	677	0	143.5	228.1297	1
226	3389	6	1102824	8	7	1148	1581	677	0	143.5	228.1297	1
227	443	6	118012340	4	2	148	252	74	0	37	42.7239	
228	53	17	420	1	1	42	58	42	42	42	0	
229	53	17	60810	2	2	74	268	37	37	37	0	

Figura 21. Tabla de registros analizados mostrada por el dashboard durante la monitorización del tráfico de red mediante monitorización en tiempo real

5.8.4 SISTEMA DE ALERTAS

Con el objetivo de mejorar la capacidad de detección temprana de amenazas, el dashboard incorpora un sistema básico de alertas automáticas basado en el porcentaje de tráfico malicioso detectado.

El sistema clasifica el nivel de riesgo en diferentes categorías:

- Estado normal
- Nivel de alerta moderado
- Nivel crítico

Umbrales de alerta

Normal: < 10% tráfico malicioso

Alerta media: 10% - 30%

Alerta crítica: > 30%

Figura 22. Umbrales de alerta implementados en el dashboard para la detección de tráfico malicioso.

Cuando el porcentaje de tráfico identificado como malicioso supera determinados umbrales, la aplicación muestra automáticamente mensajes visuales de advertencia dentro del panel de

monitorización. Este mecanismo permite representar de forma sencilla situaciones potencialmente peligrosas y aproxima al comportamiento del sistema al funcionamiento real de plataformas modernas de monitorización de ciberseguridad.

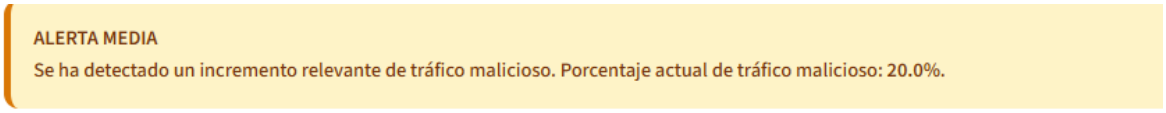


Figura 23. Alerta automática generada por el sistema ante un incremento de tráfico malicioso detectado.

5.8.5 DESPLIEGUE Y ACCESO REMOTO DE LA APLICACIÓN

El dashboard web Desarrollado durante el proyecto ha sido desplegado utilizando la plataforma Streamlit Community Cloud, permitiendo el acceso remoto a la aplicación desde cualquier navegador web sin necesidad de instalación local.

El despliegue en la nube facilita la monitorización y análisis del tráfico de red de forma centralizada, proporcionando una interfaz accesible y multiplataforma para la visualización de resultados y detección de posibles ciberataques.

La aplicación integra los modelos de Machine Learning previamente entrenados y permite tanto el análisis mediante archivos CSV como la monitorización dinámica del tráfico en tiempo real. Además, el sistema incorpora métricas visuales, gráficos interactivos y mecanismos automáticos de alerta cuando se detectan porcentajes elevados de tráfico malicioso.

Para el despliegue se ha utilizado GitHub como repositorio de almacenamiento del proyecto y Streamlit Community Cloud como plataforma de publicación de la aplicación web.

La aplicación puede ejecutarse localmente mediante el siguiente comando:

```
streamlit run app.py
```

Asimismo, la aplicación ha sido desplegada en la nube y puede accederse mediante la url indicada en el anexo.

Capítulo 6. ANÁLISIS DE RESULTADOS

6.1 MÉTRICAS DE EVALUACIÓN

6.1.1 ACCURACY

La métrica accuracy, representa el porcentaje total de predicciones clasificadas respecto al número total de registros evaluados. Esta métrica permite obtener una visión general del rendimiento del modelo.

Sin embargo, en conjuntos de datos desbalanceados, como ocurre habitualmente en ciberseguridad, puede resultar insuficiente por sí sola, ya que un modelo puede obtener valores elevados clasificando únicamente de forma correcta la clase mayoritaria.

6.1.2 PRECISIÓN

La métrica precisión, mide la proporción de predicciones correctas positivas respecto al total de predicciones positivas realizadas por el modelo. En sistemas de detección de intrusiones, una precisión elevada permite reducir el número de falsas alarmas, evitando clasificar tráfico legítimo como tráfico malicioso.

6.1.3 RECALL

La métrica recall, indica la capacidad del modelo para detectar los ataques reales presentes en el conjunto de datos. En el ámbito de la ciberseguridad, esta métrica resulta especialmente importante ya que un valor bajo implica que determinados ataques podrían pasar desapercibidos.

6.1.4 F1-SCORE

El F1-Score combina las métricas precisión y recall mediante su media, proporcionando una medida más equilibrada del rendimiento del modelo.

Esta métrica resulta especialmente útil en datasets desbalanceados, donde es necesario evaluar de forma simultánea la capacidad de detección y la reducción de falsos positivos.

6.1.5 MATRIZ DE CONFUSIÓN

Las matrices de confusión permiten representar gráficamente el comportamiento de los modelos de clasificación, mostrando las predicciones correctas e incorrectas realizadas para cada clase. Este tipo de representación facilita el análisis detallado del número de falsos positivos y falsos negativos detectado por cada algoritmo.

6.2 RESULTADOS DE MODELOS SUPERVISADOS

6.2.1 RESULTADOS DE REGRESIÓN LOGÍSTICA

La regresión logística se utilizado como modelo base de clasificación supervisada debido a su simplicidad computacional y elevada interpretabilidad.

Los resultados obtenidos muestran un rendimiento muy elevado, alcanzando valores superiores a 99% en accuracy y F1-score. No obstante, comparada con modelos más complejos, presenta un error de clasificación mayor, especialmente en la detección de ataques.

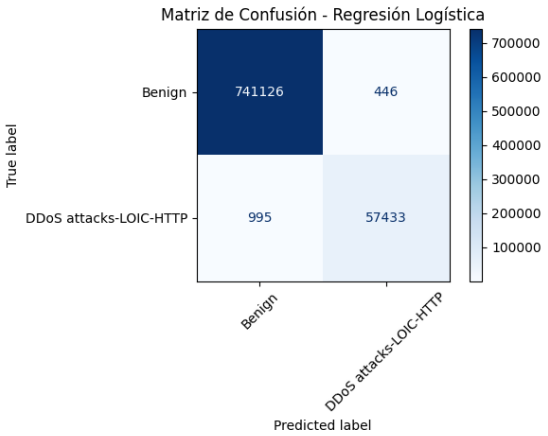


Figura 24. Matriz de confusión Regresión Logística.

Tal y como puede observarse en la Figura 24, el modelo clasifica correctamente la gran mayoría de los registros, aunque todavía aparecen algunos falsos positivos y falsos negativos.

6.2.2 RESULTADOS DE ÁRBOL DE DECISIÓN

El modelo basado en árbol de decisión ha obtenido un rendimiento superior respecto a regresión logística, reduciendo considerablemente el número de errores de clasificación.

Además de una elevada precisión, este modelo ofrece una gran capacidad de interpretabilidad, permitiendo visualizar de forma jerárquica el proceso de toma de decisiones utilizado durante la clasificación.

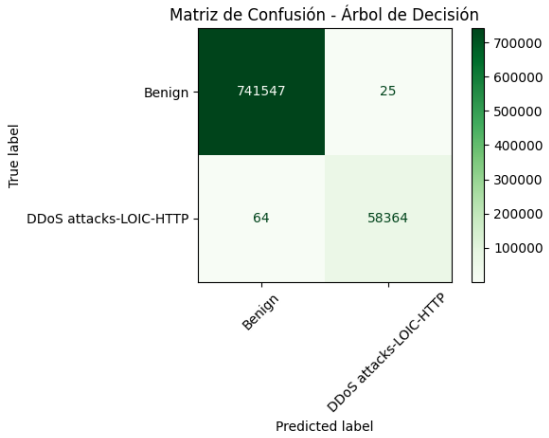


Figura 25. Matriz de confusión del Árbol de Decisión.

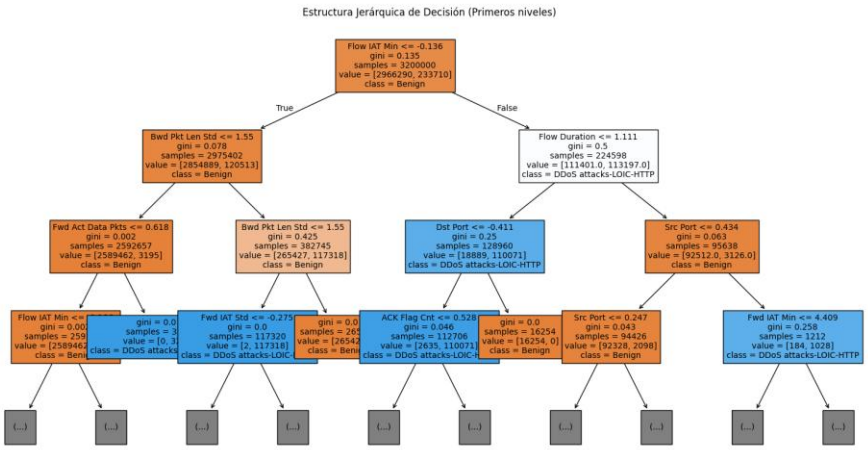


Figura 26. Estructura jerárquica simplificada del Árbol de Decisión.

La estructura mostrada permite identificar qué características del tráfico de red tienen una mayor relevancia en la separación de tráfico benigno y malicioso.

6.2.3 RESULTADOS DE RANDOM FOREST

Random Forest ha sido uno de los modelos con mejores resultados durante la fase experimental. Gracias al uso combinado de múltiples árboles de decisión, el modelo ha conseguido una elevada capacidad de generalización y una reducción del sobreajuste.

La matriz de confusión obtenida muestra un número muy reducido de errores de clasificación.

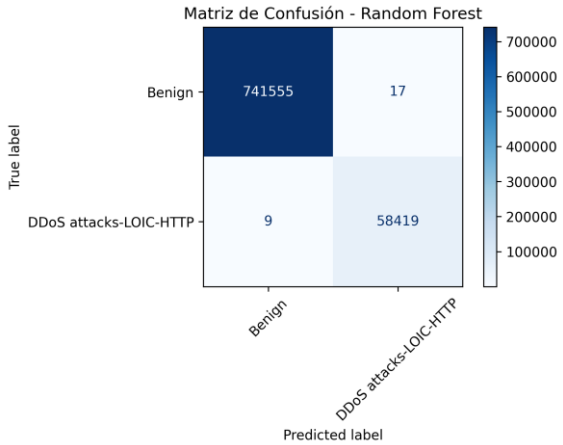


Figura 27. Matriz de confusión del modelo Random Forest.

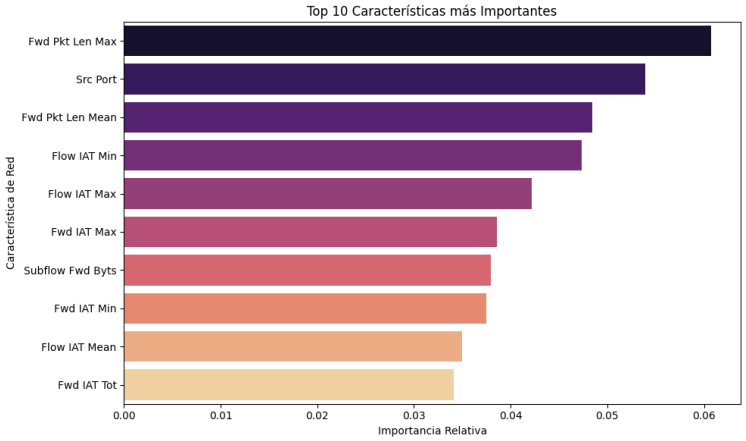


Figura 28. Variables más relevantes según Random Forest.

El análisis de importancia y variables indica que determinadas características relacionadas con tamaños de paquetes, tiempos de transmisión y estadísticas temporales influyen notablemente en la detección de ataques DDoS.

6.2.4 RESULTADOS DE GRADIENT BOOSTING

El modelo HistGradientBoosting Obtenido el mejor rendimiento global de todos los algoritmos supervisados implementados en este proyecto. Los resultados muestran una capacidad de clasificación prácticamente perfecta, Alcanzado valores cercanos al 100% tanto en accuracy como en F1-score.

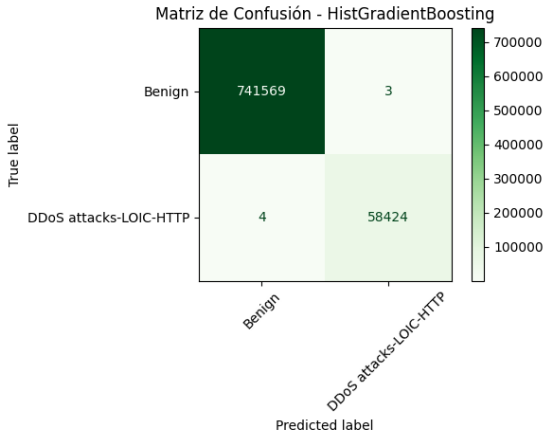


Figura 29. Matriz de confusión del modelo HistGradientBoosting.

Tal y como se observa en la figura, el número de errores de clasificación es prácticamente nulo.

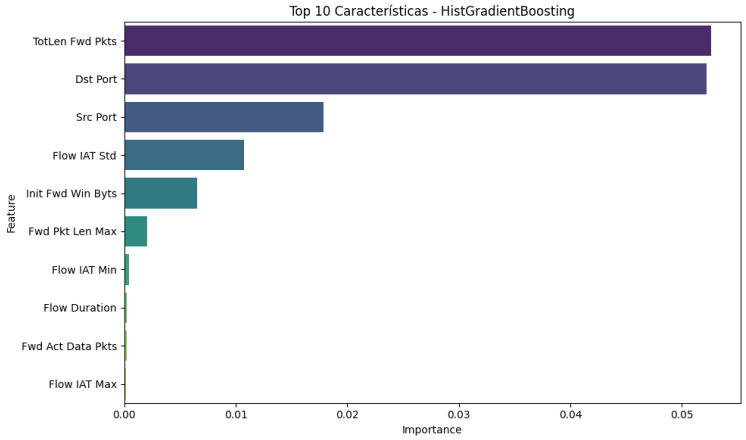


Figura 30. Variables más relevantes según HistGradientBoosting.

6.3 *RESULTADOS DE MODELOS NO SUPERVISADOS*

6.3.1 RESULTADOS DE K-MEANS

El algoritmo K-Means Ha sido utilizado para analizar si el tráfico de red podía agruparse automáticamente sin utilizar etiquetas previas durante el entrenamiento. Los resultados

muestran que determinadas agrupaciones generadas por el algoritmo coinciden parcialmente con las clases reales del conjunto de datos.

Con el objetivo de representar gráficamente los grupos detectados, se ha aplicado previamente una reducción de dimensionalidad mediante PCA (Principal Component Analysis), proyectando la información del dataset sobre dos componentes principales. Esto permite visualizar de forma más clara la distribución espacial del tráfico de red y el comportamiento del algoritmo de clustering.

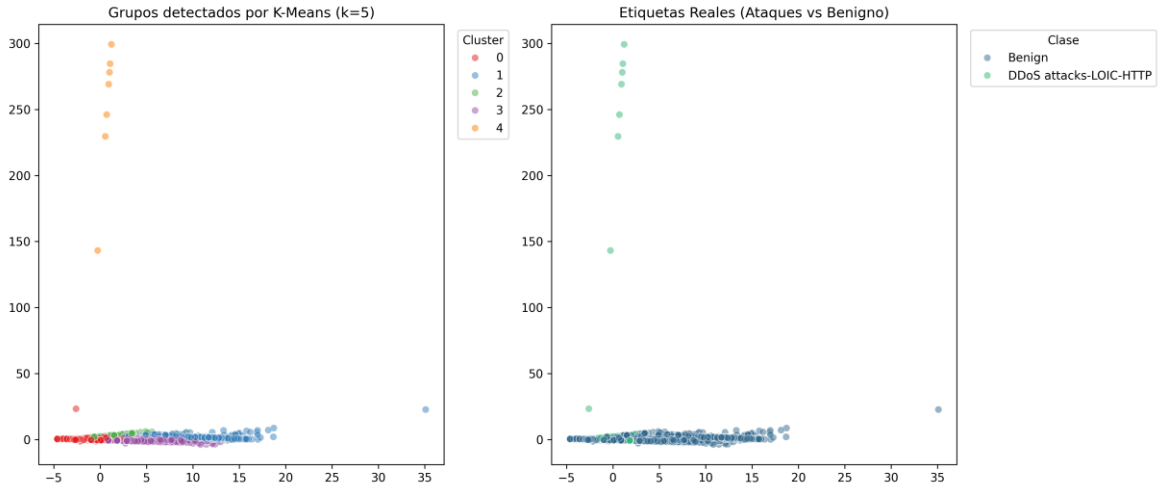


Figura 31. Comparación entre clusters detectados por K-Means y etiquetas reales.

Tal y como puede observarse en la Figura 31, El algoritmo consigue identificar ciertos patrones diferenciados dentro del tráfico de red. Algunos clusters generados por K-Means presentan una correspondencia parcial con registros asociados a DDoS, mientras que otros agrupan principalmente tráfico benigno.

Sin embargo, también se aprecia una importante superposición entre diferentes grupos, especialmente en las regiones donde existe una mayor densidad de tráfico legítimo. Este comportamiento muestra una de las principales limitaciones de los métodos no supervisados: la ausencia de etiquetas durante el entrenamiento a dificultar la separación exacta entre clases complejas o muy similares.

Además, el fuerte balanceo existente en el dataset hoy influye directamente sobre la formación de clusters, provocando que la mayoría de las agrupaciones se concentren alrededor de los patrones dominantes del tráfico benigno.

Con el objetivo de complementar el análisis visual, también se ha representado el tráfico de redes utilizando PCA y diferenciando las etiquetas reales del conjunto de datos.

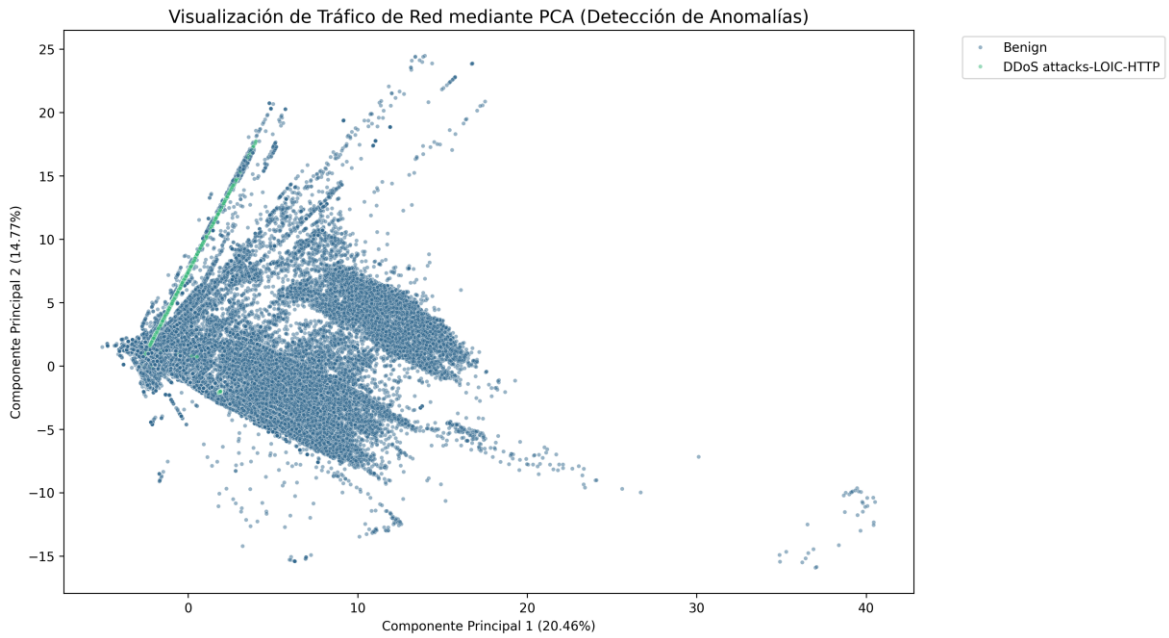


Figura 32. Representación PCA del tráfico de red y detección de anomalías.

La representación muestra que la mayor parte del tráfico benigna se concentra en regiones densas y compactas, mientras que determinados ataques aparecen como agrupaciones más aisladas dentro del espacio proyectado. Esto confirma que existen diferencias estadísticas relevantes entre ambos tipos de tráfico, lo que favorece la utilización de técnicas de Machine Learning para la detección automática de intrusiones.

En conjunto, los resultados obtenidos permiten concluir que K-Means resulta útil como herramienta exploratoria para identificar estructuras internas y posibles anomalías en grandes volúmenes de tráfico de red. No obstante, el rendimiento obtenido sigue siendo inferior al alcanzado por los modelos supervisados, ya que estos disponen de información

etiquetada durante el entrenamiento y consiguen una separación mucho más precisa entre tráfico legítimo y malicioso.

6.3.2 RESULTADOS DE CLUSTERING JERÁRQUICO

Con el Objetivo de complementar el análisis no supervisado realizado mediante K-Means, se ha aplicado un algoritmo de clustering jerárquico sobre una muestra representativa del conjunto de datos. Este enfoque permite analizar las relaciones de similitud existentes entre los distintos registros de tráfico de red sin necesidad de utilizar etiquetas previas.

El clustering jerárquico organiza progresivamente los datos en forma de árbol, agrupando aquellos registros con características similares en función de la distancia entre ellos. Para representar visualmente este proceso se generó un dendrograma, mostrado en la Figura 33.

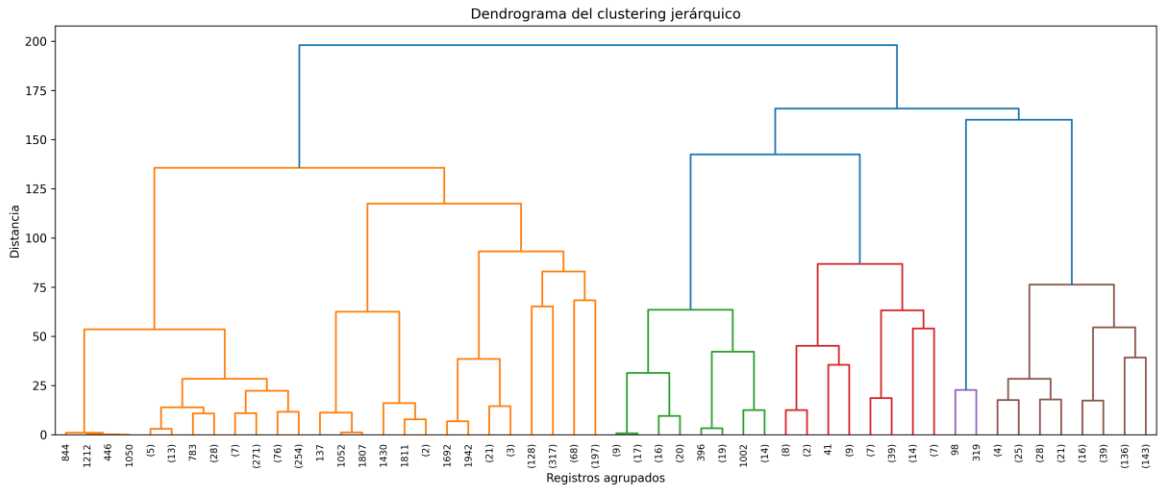


Figura 33. Dendrograma obtenido mediante clustering jerárquico

En el dendrograma puede observarse cómo diferentes grupos de registros se van fusionando progresivamente a medida que aumenta la distancia de agrupación. Las ramas inferiores representan registros muy similares entre sí, mientras que las uniones situadas en niveles superiores indican agrupaciones más generales y menos homogéneas.

Los resultados obtenidos muestran que existen patrones diferenciados dentro del tráfico analizado, lo que confirma la presencia de estructuras internas en los datos capaces de

separar distintos comportamientos de red. Este comportamiento resulta coherente con la existencia de tráfico benigno y tráfico asociado a ataques DDoS dentro del dataset utilizado.

Aunque el clustering jerárquico no ha alcanzado el nivel de precisión de los modelos supervisados, sí ha permitido realizar una exploración visual y estructural muy útil del conjunto de datos. Además, este tipo de técnicas puede resultar especialmente interesante en escenarios reales donde no se dispone de etiquetas previas o se desea detectar comportamientos anómalos desconocidos.

Sin embargo, una de las principales limitaciones de este enfoque es su elevado coste computacional cuando se trabaja con grandes volúmenes de información. Por este motivo, en este proyecto se utiliza únicamente una muestra reducida del dataset original para generar el dendrograma y facilitar la interpretación visual de los resultados.

6.4 COMPARATIVA GLOBAL DE MODELOS

6.4.1 COMPARACIÓN DE MÉTRICAS

Con el objetivo de comparar el rendimiento global de los diferentes modelos supervisados e implementados, se analizarán las métricas de accuracy, precisión, recall y F1-score obtenidas durante la fase de evaluación.

MODELO	*ACCURACY*	*PRECISION*	*RECALL*	*F1-SCORE*
Regresión logística	0.9982	0.9982	0.9982	0.9982
Random Forest	1.0000	1.0000	1.0000	1.0000
Árbol de decisión	0.9999	0.9999	0.9999	0.9999
Gradient Boosting	1.0000	1.0000	1.0000	1.0000

Tabla 22. Resumen de métricas obtenidas por los modelos supervisados

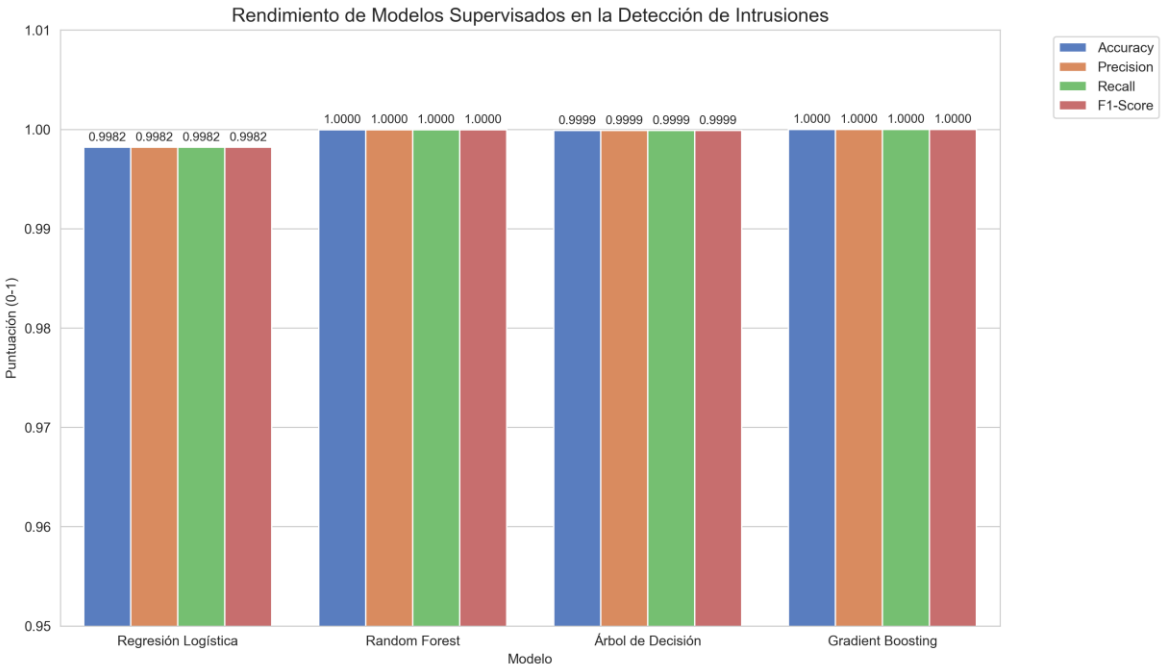


Figura 34. Comparativa de métricas de rendimiento entre los modelos supervisados.

Los resultados muestran que todos los modelos alcanzan valores muy elevados en las métricas evaluadas. No obstante, los modelos basados en ensamblado, especialmente Random Forest y Gradient Boosting, presentan el mejor comportamiento global.

La regresión logística obtiene un rendimiento ligeramente inferior al respecto al resto de modelos, aunque mantiene resultados muy competitivos considerando su menor complejidad computacional. Estos resultados reflejan que las características extraídas del dataset contienen información suficientemente representativa para conseguir distinguir entre tráfico benigno y tráfico malicioso

6.4.2 COMPARACIÓN COMPUTACIONAL

Además de las métricas de clasificación, también se ha analizado el comportamiento computacional de los modelos implementados. La Figura muestra una comparación relativa del coste computacional asociado a cada uno de los modelos supervisados implementados.

Esta comparación considera aspectos como el tiempo de entrenamiento, la complejidad del algoritmo y la cantidad de operaciones necesarias durante el procesamiento de los datos.

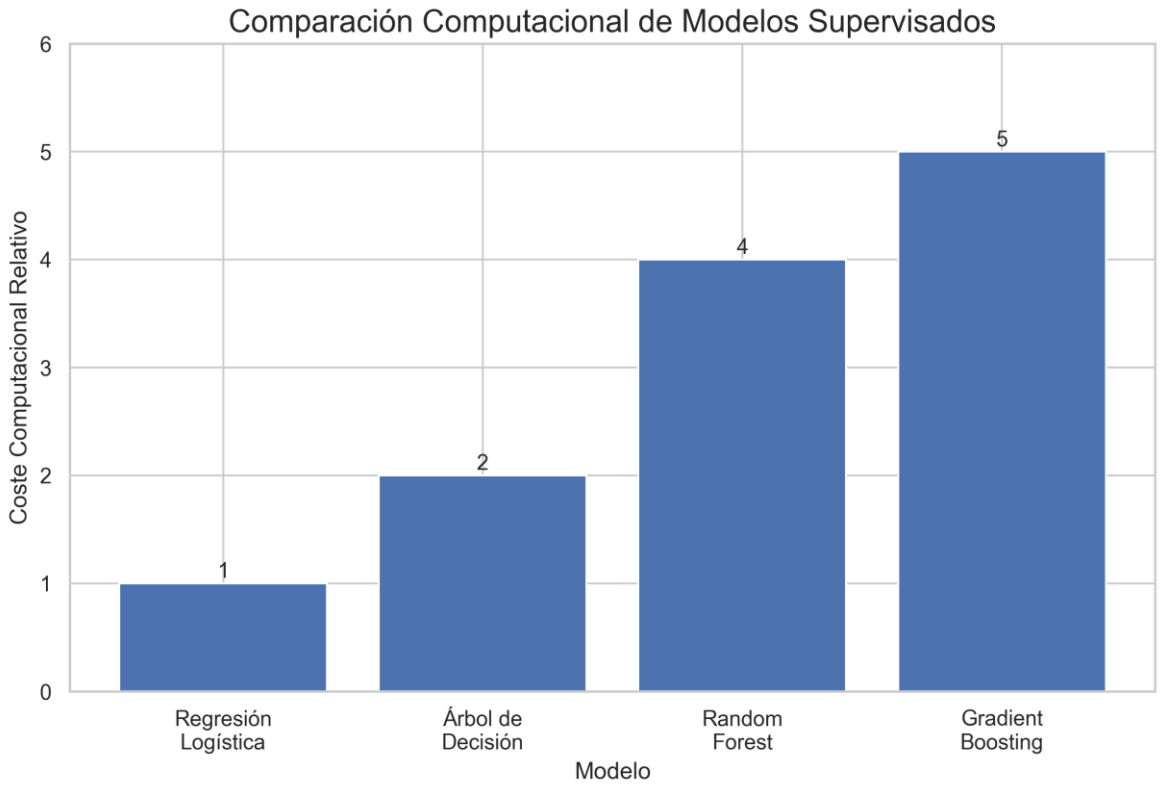


Figura 35. Comparación relativa del coste computacional de los modelos supervisados.

Tal y como se observa, la regresión logística presenta el menor coste computacional debido a la simplicidad de su estructura matemática. Por otro lado, los modelos basados en ensamblados, como Random Forest y Gradient Boosting, requieren una mayor capacidad de procesamiento debido al entrenamiento de múltiples árboles de decisión.

En particular, Gradient Boosting ha sido el modelo con mayor coste computacional relativo, aunque también ha obtenido los mejores resultados en términos de precisión y capacidad de clasificación.

6.4.3 VENTAJAS Y LIMITACIONES DE CADA ENFOQUE

Los modelos supervisados implementados en este proyecto han demostrado una elevada capacidad para detectar ataques DDoS gracias al uso de datos previamente etiquetados durante el entrenamiento. Entre sus principales ventajas destacan la alta precisión obtenida, la facilidad para evaluar su rendimiento mediante métricas cuantitativas y su gran capacidad de generalización cuando se dispone de conjuntos de datos representativos.

En particular, los algoritmos basados en ensamblados, como Random Forest y Gradient Boosting, ofrecen un muy buen comportamiento en la clasificación del tráfico de red, reduciendo considerablemente el número de falsos positivos y falsos negativos. Además, los árboles de decisión permiten una interpretación más visual e intuitiva del proceso de clasificación.

Sin embargo, los modelos supervisados también presentan ciertas limitaciones. Su rendimiento depende directamente de la calidad de los datos utilizados durante el entrenamiento y requieren grandes cantidades de información etiquetada para alcanzar buenos resultados. Además, algunos modelos más complejos presentan un mayor coste computacional y pueden resultar más difíciles de interpretar.

Por otro lado, los métodos no supervisados, como K-Means y el clustering jerárquico, permiten detectar patrones y agrupaciones sin necesidad de etiquetas previas. Esto supone una ventaja importante en escenarios reales donde pueden aparecer ataques desconocidos o comportamientos anómalos no registrados anteriormente.

No obstante, estos enfoques presentan una menor precisión de comparación con los modelos supervisados y su interpretación puede resultar más compleja. Además, algoritmos como el clustering jerárquico tienen un elevado coste computacional cuando se aplican sobre volúmenes de datos, lo que dificulta su escalabilidad en entornos reales de alta demanda.

ENFOQUE	*VENTAJAS*	*LIMITACIONES*

Modelos supervisados	Alta precisión y evaluación clara mediante métricas	Necesitan datos etiquetados
Regresión Logística	Bajo coste computacional e interpretación sencilla	Menor capacidad ante relaciones no lineales
Árbol de decisión	Fácil interpretación visual	Riesgo de sobreajuste
Random Forest	Alta precisión y robustez	Mayor coste computacional
Gradient Boosting	Mejor rendimiento global	Mayor complejidad de entrenamiento
Modelos no supervisados	Detectan agrupaciones sin etiquetas previas	Menor precisión e interpretación más compleja
K-Means	Rápido y útil para exploración	Requiere definir número de clusters
Clustering jerárquico	Permite visualizar relaciones mediante dendrograma	Poco escalable con grandes volúmenes de datos

Tabla 23. Ventajas y limitaciones de los enfoques utilizados

6.5 DISCUSIÓN DE RESULTADOS

Los resultados obtenidos a lo largo del proyecto permiten concluir que las técnicas de Machine Learning aplicadas al análisis de tráfico de red son altamente eficaces para la detección de ataques DDoS.

Los modelos supervisados alcanzan métricas extremadamente elevadas, obteniendo valores próximos al 100% en accuracy, precisión, recall y F1-Score. Esto demuestra que las

variables seleccionadas durante el preprocesamiento contienen información relevante para diferenciar de forma precisa entre tráfico Benigno y tráfico malicioso.

Entre todos los modelos implementados, Gradient Boosting y Random Forest han sido los que ofrecen mejores resultados globales, destacando tanto por su precisión como por su robustez frente a diferentes patrones de tráfico. No obstante, como modelo final del sistema desarrollado se ha seleccionado HistGradientBoosting, ya que ha Mostrado el mejor rendimiento global durante la fase experimental y un número de errores de clasificación prácticamente nulo. Esta elección resulta coherente con el objetivo principal del proyecto, centrado en maximizar la capacidad de detección de tráfico malicioso y reducir al mínimo los ataques no identificados.

Aunque Random Forest también ha obtenido resultados excelentes y ofrece una mayor facilidad de interpretación mediante el análisis de importancia de variables, HistGradientBoosting se considera la alternativa más adecuada para la aplicación final debido a su rendimiento predictivo. Su principal limitación es un mayor coste computacional relativo, derivado del entrenamiento secuencial de los árboles, aunque este inconveniente resulta asumible en el sistema desarrollado, ya que el modelo se entrena previamente y la aplicación web utiliza el modelo ya almacenado para realizar las predicciones.

La regresión logística, aunque más sencilla, y con un coste computacional considerablemente inferior, también ha obtenido resultados muy competitivos. Sin embargo, presenta un número de errores superior y una menor capacidad para modelar relaciones no lineales complejas, por lo que no se considera la opción más adecuada como modelo final.

Por otro lado, los modelos no supervisados permiten analizar la estructura interna de los datos y detectar agrupaciones automáticas dentro del tráfico de red. Tanto K-Means como el clustering jerárquico muestran capacidad para identificar comportamientos similares entre registros, aunque con menor precisión que los modelos supervisados. Por ello, estos métodos se consideran complementarios al modelo principal, especialmente útiles para la exploración de patrones anómalos o el análisis de tráfico no etiquetado.

Además, las técnicas de reducción de dimensionalidad y visualización facilitan la representación gráfica de las anomalías detectadas y permiten interpretar de forma más intuitiva el comportamiento del tráfico analizado.

En conjunto, los resultados obtenidos validan el uso de las técnicas de inteligencia artificial como herramienta eficaz para mejorar la detección temprana de amenazas en entornos de ciberseguridad. De forma concreta, HistGradientBoosting se establece como el modelo final de clasificación integrado en el sistema desarrollado, mientras que Random Forest y las técnicas no supervisadas aportan información complementaria para el análisis e interpretación del tráfico de red.

Aunque los resultados obtenidos son muy positivos, deben interpretarse teniendo en cuenta que el dataset utilizado procede de un entorno controlado. Por tanto, el comportamiento del sistema en una red empresarial podría verse afectado por tráfico más variable, nuevos tipos de ataques o cambios de patrones de uso de la red.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

El presente trabajo ha demostrado la viabilidad del uso de técnicas de Machine Learning para la detección inteligente de ciberataques en tráficos de red, especialmente en escenarios relacionados con ataques distribuidos de denegación de servicio (DDoS).

A lo largo del proyecto se han desarrollado distintas fases de análisis, procesamiento y modelado de datos utilizando datasets especializados en ciberseguridad, permitiendo entrenar y evaluar múltiples modelos supervisados y no supervisados orientados a la detección de intrusiones.

Los resultados obtenidos muestran que los modelos supervisados alcanzan niveles de precisión muy elevados. Aunque Random Forest e HistGradientBoosting presentan métricas prácticamente perfectas, se ha seleccionado HistGradientBoosting como modelo final del sistema, al haber obtenido el mejor rendimiento global durante la fase experimental y presentar un número de errores de clasificación prácticamente nulo. Random Forest también ha demostrado una elevada robustez y ha resultado especialmente útil para analizar la importancia de las variables, mientras que los modelos no supervisados han permitido estudiar la estructura interna de los datos e identificar agrupaciones de tráfico con comportamientos similares.

Además del desarrollo de los modelos de inteligencia artificial, el proyecto incorpora un dashboard web interactivo capaz de representar métricas, visualizar resultados, simular monitorización en tiempo real y generar alertas automáticas ante posibles situaciones de riesgo. Esta funcionalidad aporta una aproximación más cercana a entornos reales de monitorización de ciberseguridad.

En conjunto, el sistema desarrollado valida el potencial de las técnicas de inteligencia artificial como herramienta eficaz para mejorar la detección temprana de amenazas y automatizar procesos y análisis dentro del ámbito de la ciberseguridad.

7.2 OBJETIVOS ALCANZADOS

Los objetivos planteados al inicio del proyecto han sido alcanzados satisfactoriamente. Entre los principales resultados obtenidos destacan:

- Estudio de diferentes técnicas de detección de intrusiones basadas en Machine Learning.
- Análisis y comparación de datasets especializados en ciberseguridad.
- Desarrollo de un pipeline completo de procesamiento de datos.
- Implementación de modelos supervisados y no supervisados para la detección de ataques DDoS.
- Evaluación comparativa del rendimiento de distintos algoritmos de clasificación.
- Aplicación de técnicas de reducción de dimensionalidad y clustering.
- Desarrollo de visualizaciones avanzadas para representar el comportamiento del tráfico de red.
- Implementación de un dashboard web interactivo para la monitorización del sistema.
- Desarrollo de un sistema básico de alertas automáticas ante tráfico malicioso.

El proyecto ha permitido integrar conocimientos relacionados con programación, análisis de datos, inteligencia artificial y ciberseguridad en una solución funcional orientada a la detección de amenazas.

7.3 LIMITACIONES DEL PROYECTO

Aunque los resultados obtenidos son muy positivos, el sistema desarrollado presenta algunas limitaciones que deben tenerse en cuenta.

En primer lugar, el entrenamiento y evaluación de los modelos se ha realizado utilizando datasets públicos obtenidos en entornos controlados. Por tanto, el comportamiento del sistema en redes reales podría verse afectado por tráfico más variable, nuevos patrones de ataque o situaciones no presentes en los conjuntos de datos utilizados.

Además, el sistema de monitorización en tiempo real implementado corresponde a una simulación basada en archivos CSV y no a la captura directa de tráfico real desde interfaces de red. Esto limita parcialmente su aplicación en entornos productivos reales.

Otra limitación importante es el elevado coste computacional asociado a algunos modelos avanzados, especialmente en fases de entrenamiento sobre grandes volúmenes de datos.

Finalmente, el sistema de alertas desarrollado utiliza reglas simples basadas en porcentajes de tráfico malicioso, por lo que podría mejorarse mediante mecanismos más avanzados de análisis adaptativo.

7.4 LÍNEAS FUTURAS DE TRABAJO

Como posibles líneas futuras de desarrollo, el proyecto podría ampliarse mediante diferentes mejoras orientadas tanto al rendimiento del sistema como su aplicación en entornos reales.

Entre las principales líneas futuras destacan:

- Integración del sistema con captura de tráfico real utilizando herramientas como Wireshark o Scapy.
- Implementación de modelos de Deep Learning para mejorar la detección de patrones complejos de ataque.
- Mejora del sistema de alertas mediante técnicas adaptativas y análisis dinámico del comportamiento de red.
- Despliegue del dashboard web en servidores cloud o entornos empresariales reales.
- Desarrollo de funcionalidades de autenticación y control de acceso para la aplicación web.

- Incorporación de nuevos tipos de ataques y datasets actualizados para mejorar la robustez del sistema.

Estas mejoras permitirían evolucionar el sistema hacia una plataforma más compleja, escalable y preparada para entornos reales de ciberseguridad.

Capítulo 8. BIBLIOGRAFÍA

[1] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, y K.-Y. Tung, «Intrusion detection system: A comprehensive review», *J. Netw. Comput. Appl.*, vol. 36, n.º 1, pp. 16-24, ene. 2013, doi: 10.1016/j.jnca.2012.09.004.

[2] K. A. Scarfone y P. M. Mell, «Guide to Intrusion Detection and Prevention Systems (IDPS)», National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[3] «IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018) ». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv

[4] A. L. Buczak y E. Guven, «A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection», *IEEE Commun. Surv. Tutor.*, vol. 18, n.º 2, pp. 1153-1176, 2016, doi: 10.1109/COMST.2015.2494502.

[5] «Streamlit • A faster way to build and share data apps». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://streamlit.io/

[6] «Welcome to Python.org», Python.org. Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://www.python.org/

[7] F. Pedregosa *et al.*, «Scikit-learn: Machine Learning in Python», *J. Mach. Learn. Res.*, vol. 12, n.º 85, pp. 2825-2830, 2011.

[8] «Visual Studio Code - The open source AI code editor | Your home for multi-agent development». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://code.visualstudio.com/

[9] K. A. Scarfone y P. M. Mell, «Guide to Intrusion Detection and Prevention Systems (IDPS)», National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-94, 2007. doi: 10.6028/NIST.SP.800-94.

[10] A. L. Buczak y E. Guven, «A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection», *IEEE Commun. Surv. Tutor.*, vol. 18, n.º 2, pp. 1153-1176, 2016, doi: 10.1109/COMST.2015.2494502.

[11] Y. Alotaibi y M. Ilyas, «Ensemble-Learning Framework for Intrusion Detection to Enhance Internet of Things' Devices Security», *Sensors*, vol. 23, n.º 12, p. 5568, jun. 2023, doi: 10.3390/s23125568.

[12] E. Alalwany y I. Mahgoub, «Classification of Normal and Malicious Traffic Based on an Ensemble of Machine Learning for a Vehicle CAN-Network», *Sensors*, vol. 22, n.º 23, p. 9195, nov. 2022, doi: 10.3390/s22239195.

[13] Y. Alotaibi y M. Ilyas, «Ensemble-Learning Framework for Intrusion Detection to Enhance Internet of Things' Devices Security», *Sensors*, vol. 23, n.º 12, p. 5568, jun. 2023, doi: 10.3390/s23125568.

[14] W. Li, W. Meng, y L. F. Kwok, «Surveying Trust-Based Collaborative Intrusion Detection: State-of-the-Art, Challenges and Future Directions», *IEEE Commun. Surv. Tutor.*, vol. 24, n.º 1, pp. 280-305, 2022, doi: 10.1109/COMST.2021.3139052.

[15] «Network Intrusion dataset (CIC-IDS- 2017) ». Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset

[16] L. Breiman, «Random Forests», *Mach. Learn.*, vol. 45, n.º 1, pp. 5-32, oct. 2001, doi: 10.1023/A:1010933404324.

[17] J. H. Friedman, «Greedy function approximation: A gradient boosting machine. », *Ann. Stat.*, vol. 29, n.º 5, oct. 2001, doi: 10.1214/aos/1013203451.

[18] *Principal Component Analysis*. en Springer Series in Statistics. New York: Springer-Verlag, 2002. doi: 10.1007/b98835.

[19] «Graphical Representation of Data Using Principal Components», en *Principal Component Analysis*, en Springer Series in Statistics., New York: Springer-Verlag, 2002, pp. 78-110. doi: 10.1007/0-387-22440-8_5.

[20] «Choosing a Subset of Principal Components or Variables», en *Principal Component Analysis*, en Springer Series in Statistics., New York: Springer-Verlag, 2002, pp. 111-149. doi: 10.1007/0-387-22440-8_6.

[21] J. MacQueen, «Some methods for classification and analysis of multivariate observations», *Proc. Fifth Berkeley Symp. Math. Stat. Probab. June 21-July 18, 1965, Dec. 27, 1965-January 7, 1966*, jun. 1965, Accedido: 27 de mayo de 2026. [En línea]. Disponible en: https://digicoll.lib.berkeley.edu/record/113015

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

Los Objetivos de Desarrollo Sostenible constituyen un conjunto de 17 metas interconectadas adoptadas por las Naciones Unidas en 2015 con el propósito de garantizar un futuro sostenible para todas las personas.

Figura 36. Objetivos de Desarrollo Sostenible

En la actualidad, la transformación digital y el crecimiento de las infraestructuras tecnológicas han incrementado notablemente la importancia de la ciberseguridad dentro de la sociedad. La protección de sistemas de información, redes de comunicación y servicios digitales se ha convertido en un elemento fundamental para garantizar la estabilidad de empresas, instituciones y servicios esenciales. En este contexto, los avances en inteligencia artificial, análisis de datos y aprendizaje automático permiten desarrollar soluciones cada

vez más eficientes para la detección temprana de amenazas y la protección de infraestructuras críticas.

El presente Trabajo Fin de Grado se alinea con varios Objetivos de Desarrollo Sostenible al abordar el desarrollo de un sistema inteligente de detección de ciberataques mediante técnicas de Machine Learning aplicadas al análisis de tráfico de red.

En primer lugar, el proyecto contribuye directamente al ODS 9: Industria, innovación e infraestructura, ya que promueve el desarrollo de soluciones tecnológicas innovadoras orientadas a reforzar la seguridad de las infraestructuras digitales empresariales. La detección temprana de ataques DDoS y otras amenazas informáticas permite mejorar la resiliencia de los sistemas de comunicación, reducir riesgos de interrupción de servicios y aumentar la fiabilidad de las infraestructuras tecnológicas. Además, el uso de inteligencia artificial aplicado a la ciberseguridad representa una línea de innovación con gran impacto dentro del ámbito industrial y tecnológico.

Asimismo, este trabajo se relaciona con el ODS 16: Paz, justicia e instituciones sólidas, debido a que la ciberseguridad constituye actualmente un elemento clave para garantizar entornos digitales seguros y confiables. La protección frente a ciberataques contribuye a preservar la integridad de los sistemas de información, proteger datos sensibles y evitar incidentes que puedan comprometer la seguridad de organizaciones públicas y privadas. El desarrollo de sistemas inteligentes de detección de intrusiones favorece además la confianza en las tecnologías digitales y promueve infraestructuras más seguras y transparentes.

Por otro lado, el proyecto también mantiene relación con el ODS 4: Educación de calidad, ya que fomenta la aplicación práctica de conocimientos adquiridos en diferentes disciplinas como programación, estadística, análisis de datos, inteligencia artificial y ciberseguridad. El desarrollo del sistema ha permitido integrar competencias técnicas y científicas dentro de un entorno práctico orientado a la resolución de problemas reales. Además, el proyecto contribuye a impulsar el aprendizaje de tecnologías emergentes altamente demandadas en el ámbito profesional actual.

De forma complementaria, el trabajo también puede vincularse parcialmente con el ODS 8: Trabajo decente y crecimiento económico, debido a que la mejora de la ciberseguridad empresarial contribuye a proteger servicios digitales, reducir pérdidas económicas derivadas de ataques informáticos y favorecer entornos tecnológicos más seguros para el desarrollo de actividades económicas y empresariales.

En conjunto, este proyecto demuestra cómo la aplicación de técnicas de inteligencia artificial y análisis de datos puede contribuir no solo al avance tecnológico, sino también al desarrollo de soluciones alineadas con objetivos globales de sostenibilidad, innovación y seguridad digital.

ANEXO II. MANUAL DE INSTALACIÓN

ANEXO II. 1. INTRODUCCIÓN

Este anexo describe el procedimiento necesario para ejecutar la aplicación web desarrollada durante el proyecto, así como los requisitos software necesarios para su funcionamiento.

La aplicación desarrollada permite realizar tareas de monitorización y detección inteligente de ciberataques mediante modelos de Machine Learning previamente entrenados. El sistema ha sido implementado utilizando Python y Streamlit, permitiendo la visualización interactiva de resultados y métricas de seguridad en tiempo real.

ANEXO II. 2. REQUISITOS DEL SISTEMA

Para la ejecución del sistema es necesario disponer de los siguientes requisitos:

- Python 3.10 o superior.
- Conexión a internet.
- Navegador web actualizado.
- Librerías Python especificadas en el archivo requirements.txt

ANEXO II. 3. ESTRUCTURA DEL PROYECTO

El sistema desarrollado está compuesto por los siguientes archivos principales:

- app.py: aplicación principal desarrollada con Streamlit
- modelo_final_hgp.pkl: modelo de machine learning entrenado
- scaler.pkl: escalador utilizado durante el preprocesamiento
- requirements.txt: librerías necesarias para la ejecución
- trafico_live.csv: dataset utilizado para simulación de tráfico

ANEXO II. 4. INSTALACIÓN DE DEPENDENCIAS

Las librerías necesarias para la ejecución del sistema pueden instalarse mediante el siguiente comando:

```
pip install -r requirements.txt
```

ANEXO II. 5. EJECUCIÓN LOCAL DE LA APLICACIÓN

Una vez instaladas las dependencias, la aplicación puede ejecutarse localmente mediante el siguiente comando:

```
streamlit run app.py
```

Tras ejecutar el comando anterior, el sistema abrirá automáticamente la aplicación web en el navegador predeterminado del usuario.

ANEXO II. 6. DESPLIEGUE REMOTO DE LA APLICACIÓN

La aplicación web ha sido desplegada utilizando la plataforma Streamlit Community Cloud, permitiendo el acceso remoto al sistema desde cualquier navegador web sin necesidad de instalación local.

Para el despliegue remoto se ha utilizado GitHub como repositorio de almacenamiento del código fuente del proyecto.

ANEXO II. 7. ACCESO REMOTO AL SISTEMA

La aplicación desplegada puede accederse mediante la siguiente URL pública:

Sistema de Detección de Ciberataques · Streamlit

ANEXO III. MANUAL DE USUARIO

ANEXO III. 1. INTRODUCCIÓN

El anexo describe el funcionamiento y utilización de la aplicación web desarrollada para la detección inteligente de ciberataques mediante técnicas de Machine Learning.

La aplicación permite analizar el tráfico de web, visualizar métricas de seguridad y detectar posibles intrusiones mediante una interfaz interactiva desarrollada con Streamlit.

ANEXO III. 2. ACCESO A LA APLICACIÓN

El usuario puede acceder a la aplicación desde cualquier navegador web utilizando la URL pública del sistema:

Sistema de Detección de Ciberataques · Streamlit

Figura 37. Página principal del dashboard web desarrollado para la detección de intrusiones

ANEXO III. 3.CARGA DE ARCHIVOS CSV

La aplicación permite cargar archivos CSV que contengan registros de tráfico de red para realizar predicciones automáticas utilizando el modelo de Machine Learning previamente entrenado.

Para realizar el análisis deben seguirse los siguientes pasos:

1. Pulsar el botón "Upload" para subir el archivo CSV

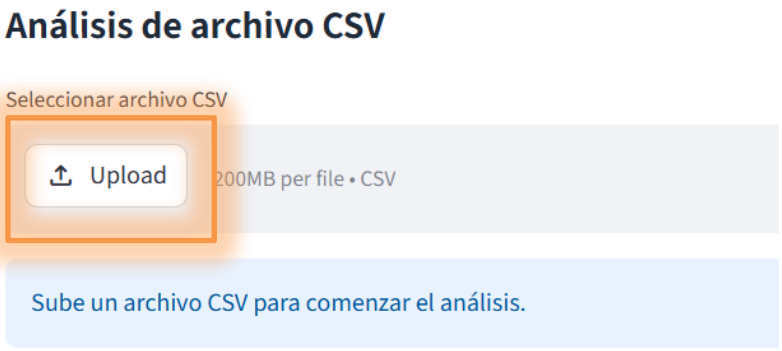


Figura 38. Botón para subir CSV

2. Seleccionar el archivo CSV deseado
3. Esperar el procesamiento automático de los datos
4. Visualizar los resultados obtenidos en el dashboard

ANEXO III. 4.VISUALIZACIÓN DE RESULTADOS

Tras el procesamiento de los datos, el sistema muestra diferentes métricas de estadísticas y gráficos interactivos que permiten interpretar el comportamiento del tráfico analizado.

Entre las principales métricas mostradas destacan:

- Registros analizados.
- Ataques detectados.
- Tráfico benigno.

- Porcentaje de tráfico malicioso.

Asimismo, el sistema genera diferentes representaciones gráficas para facilitar la interpretación de los resultados obtenidos.

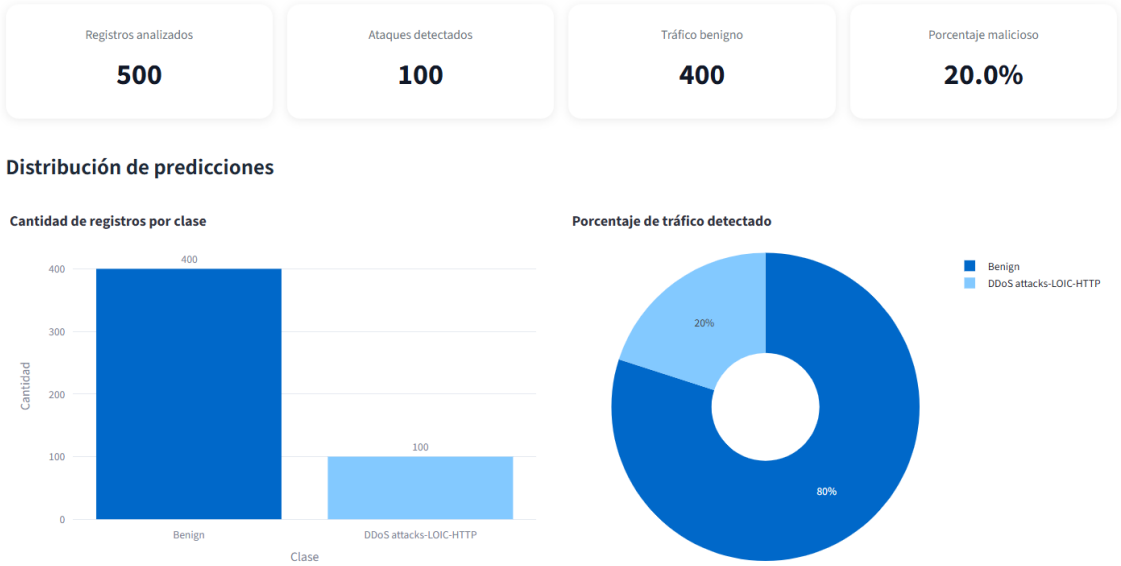


Figura 39. Medidas estadísticas y representaciones gráficas de los resultados

ANEXO III. 5.MONITORIZACIÓN EN TIEMPO REAL

La aplicación dispone de un modo de monitorización dinámica que permite simular tráfico de red en tiempo real mediante la actualización automática de registros analizados.

Esta funcionalidad permite visualizar el comportamiento del sistema ante diferentes escenarios de tráfico y posibles ataques.

El sistema utiliza un archivo CSV denominado trafico_live.csv, el cual contiene registros de tráfico empleados para simular el funcionamiento de una monitorización continua.

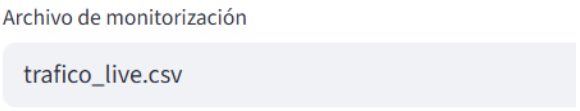


Figura 40. Archivo de monitorización

Asimismo, la aplicación permite configurar distintos parámetros relacionados con el procesamiento dinámico de datos.

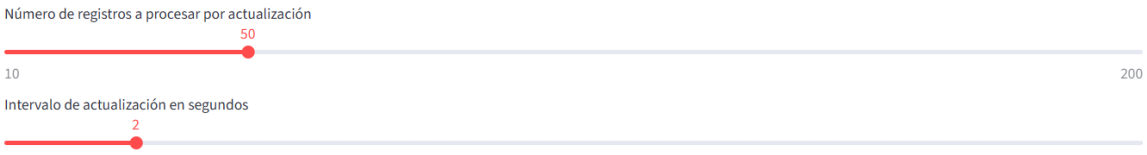


Figura 41. Configuración de parámetros para el procesamiento dinámico de datos

En concreto, el usuario puede modificar:

- El número de registros procesados en cada actualización.
- El intervalo temporal entre actualizaciones automáticas.

Estas opciones permiten adaptar la velocidad de análisis y la carga de procesamiento según las necesidades del entorno de monitorización. Finalmente, el sistema incorpora un botón de inicio que activa automáticamente el proceso de monitorización dinámica.

Figura 42. Botón de inicio de monitorización

Una vez iniciada la monitorización, la aplicación comienza a procesar registros de tráfico de manera periódica, mostrando métricas actualizadas, gráficas dinámicas y posibles alertas de seguridad cuando se detecta tráfico malicioso.

ANEXO III. 6.DESCARGA DE RESULTADOS

El sistema permite descargar los resultados generados tras el análisis en formato CSV, facilitando posteriores tareas de almacenamiento o análisis de evidencias.

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	To
400	3389	6	1871818	8	7	1128	
401	53	17	757	1	1	30	
402	80	6	33725076	2	0	0	
403	32340	6	88471690	2	0	0	
404	80	6	11258255	2	0	0	
405	80	6	1625510	3	4	20	
406	57176	6	169	2	0	0	
407	50855	6	37	1	1	0	
408	80	6	87	2	0	0	
409	3389	6	2001834	8	7	1132	

Figura 43. Botón de descarga de resultados