Imagen que contiene Texto

Descripción generada automáticamente

**Desarrollo de un sistema de detección de intrusos en redes utilizando técnicas de inteligencia artificial**

Grado en Ingeniería Informática

2024 - 2025

|  |
| --- |
| **ALFONSO GALOCHA SANCHEZ**  **ALUMNO** |

|  |
| --- |
| **ALBERTO BALLESTEROS RODRIGUEZ**  **TUTOR** |

Índice de Contenidos

[1. Introducción 5](#_Toc896089722)

[2. Metodología 6](#_Toc1891987855)

[2.1. Limpieza del Dataset 7](#_Toc361467700)

[2.2. Análisis Exploratorio de Datos (EDA) 9](#_Toc1143340802)

[2.3. Selección y entrenamiento de modelos de IA 11](#_Toc935788764)

[2.4. Validación cruzada 13](#_Toc844893482)

[2.5. Otros tipos de validación 14](#_Toc1685723848)

[2.6. Entorno de Desarrollo y herramientas 15](#_Toc619230630)

[3. Resultados 16](#_Toc1798516679)

[3.1 Preparación del dataset 16](#_Toc1104354121)

[3.2 Análisis exploratorio del dataset (EDA) 18](#_Toc790893970)

[3.3 Evaluación de los modelos 21](#_Toc1965438065)

[3.3.1 Evaluación del modelo Random Forest 21](#_Toc728543331)

[3.3.2 Evaluación del modelo Regresión Logística 24](#_Toc699439161)

[3.3.3 Evaluación del modelo Support Vector Machine 26](#_Toc1241171515)

[3.4 Comparativa de los modelos 29](#_Toc90749660)

[3.4.1 Comparación de eficiencia computacional 30](#_Toc1853788753)

[1. Conclusiones 32](#_Toc1668036142)

[2. Limitaciones y Líneas futuras 32](#_Toc52438506)

[3. Bibliografía 33](#_Toc1978754365)

**Resumen**

En un mundo donde la ciberseguridad se ha vuelto fundamental para proteger sistemas y redes, este Trabajo de Fin de Grado se centra en el diseño, implementación y evaluación de un sistema de detección de intrusos (IDS) que utiliza técnicas de inteligencia artificial. Mediante algoritmos de aprendizaje automático, el objetivo es identificar tráfico malicioso en las redes, superando las limitaciones de los sistemas tradicionales basados en firmas.

El desarrollo se ha llevado a cabo utilizando el conjunto de datos público CICIDS2017, que es muy reconocido en la investigación de ciberseguridad. Después del proceso de limpieza y análisis exploratorio de los datos, se seleccionaron varias características relevantes para entrenar modelos supervisados. Se aplicaron y compararon diferentes algoritmos de clasificación, como Random Forest, Regresión Logística y Máquinas de Vectores de Soporte (SVM), evaluando su rendimiento a través de métricas como el F1-score, la curva ROC-AUC y los tiempos de inferencia.

Los resultados muestran que el modelo Random Forest ofrece un rendimiento destacado tanto en precisión como en capacidad de discriminación, manteniendo una eficiencia computacional aceptable. Por su parte, la regresión logística destaca por su simplicidad y velocidad de ejecución, logrando métricas comparables a las del modelo anterior. En cambio, el modelo SVM, aunque sensible al tráfico malicioso, presenta una mayor tasa de falsos positivos y un coste computacional elevado, lo que limita su aplicabilidad práctica.

Este trabajo demuestra cómo la inteligencia artificial puede aplicarse de manera efectiva en la detección de amenazas en redes, ofreciendo una alternativa adaptable y eficiente frente a los IDS tradicionales.

**Palabras clave:** ciberseguridad, detección de intrusos, inteligencia artificial, aprendizaje automático, tráfico de red, IDS, clasificación binaria

**Abstract**

In a world where cybersecurity has become essential for protecting systems and networks, this Final Degree Project focuses on the design, implementation, and evaluation of an Intrusion Detection System (IDS) that uses artificial intelligence techniques. Through machine learning algorithms, the goal is to identify malicious network traffic, overcoming the limitations of traditional signature-based systems.

The development was carried out using the public CICIDS2017 dataset, which is widely recognized in cybersecurity research. After data cleaning and exploratory analysis, several relevant features were selected to train supervised models. Different classification algorithms were applied and compared, such as Random Forest, Logistic Regression, and Support Vector Machines (SVM), evaluating their performance through metrics like F1-score, ROC-AUC curve, and inference time.

Experimental results show that the Random Forest model offers outstanding performance in both accuracy and discriminative power, while maintaining acceptable computational efficiency. Logistic Regression stands out for its simplicity and speed, achieving metrics comparable to the previous model. On the other hand, the SVM model, although sensitive to malicious traffic, presents a higher rate of false positives and a greater computational cost, which limits its practical applicability.

This work demonstrates how artificial intelligence can be effectively applied to network threat detection, offering an adaptable and efficient alternative to traditional IDS solutions.

**Keywords:** cybersecurity, intrusion detection, artificial intelligence, machine learning, network traffic, IDS, binary classification

**Agradecimientos.**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc porttitor, diam at ornare pretium, eros ipsum mattis purus, non blandit lectus dui eu felis. Quisque gravida, lacus id aliquet pulvinar, dui diam luctus eros, vitae bibendum magna velit ut lorem. Proin sed nunc libero, sit amet adipiscing dolor. Integer tempus pretium quam, tristique commodo nunc mattis et. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nisl sapien, sagittis ut ultricies eget, ultrices at ante. Mauris porta sagittis consectetur. Donec pellentesque, purus sed fringilla rutrum, leo nulla pellentesque nunc, quis imperdiet nisi tortor vel nisl. Vestibulum suscipit pellentesque condimentum. Aliquam dignissim, enim quis molestie feugiat, neque quam viverra urna, a cursus mauris metus eleifend lacus. Vivamus sit amet erat ante. Phasellus a vehicula felis. Morbi ullamcorper luctus gravida. Duis imperdiet, turpis et bibendum venenatis, velit lectus porta massa, eu fringilla risus velit sit amet lectus. Duis ac mauris enim, at dictum velit.

# Introducción

La ciberseguridad se ha convertido en uno de los campos tecnológicos más relevantes en los últimos años, impulsado tanto por sus complejas implicaciones técnicas como por la creciente necesidad de proteger las infraestructuras digitales frente a ciberataques. El análisis avanzado de datos y la identificación automatizada de patrones inusuales se presentan como elementos esenciales para anticiparse y prevenir incidentes antes de que ocurran, especialmente en contextos cada vez más digitalizados.

Este Trabajo de Fin de Grado se centra en la ciberseguridad debido no solo al interés creciente que genera este ámbito, sino también a la necesidad crítica de mejorar los sistemas actuales de detección de intrusos (IDS). Tradicionalmente, estos sistemas han dependido en gran medida de firmas predefinidas o reglas estáticas para identificar amenazas. Sin embargo, investigaciones recientes han demostrado que esta aproximación posee limitaciones significativas, tales como su incapacidad para reconocer ataques desconocidos o mínimamente modificados y la generación de numerosos falsos positivos, que reducen su eficacia y aumentan la carga operativa (Garcia-Teodoro et al., 2009; Shiravi et al., 2012).

De hecho, estudios recientes alertan sobre una creciente adopción de técnicas de inteligencia artificial por parte de ciberdelincuentes para sofisticar sus ataques. Por ejemplo, Rigaki y Garcia (2018) han documentado cómo las Redes Generativas Antagónicas (GANs) se emplean para adaptar las comunicaciones de malware y evadir los sistemas tradicionales de detección. Sadeghzadeh (2020) también destaca cómo la IA facilita la automatización y perfeccionamiento de ataques como phishing y malware avanzado. Esta evolución en las técnicas ofensivas evidencia la necesidad urgente de desarrollar sistemas defensivos más dinámicos y robustos.

En paralelo, la inteligencia artificial está siendo explorada como herramienta clave para reforzar la detección y prevención de intrusiones. Diversas investigaciones han validado que técnicas basadas en aprendizaje automático y minería de datos pueden analizar grandes volúmenes de información en tiempo real, mejorando sustancialmente la detección de amenazas y reduciendo el impacto de ataques exitosos (Buczak & Guven, 2016; Schmitt, 2023).

Por lo tanto, el objetivo central de este proyecto es diseñar un IDS basado en técnicas de inteligencia artificial orientado específicamente a identificar patrones sospechosos asociados con amenazas comunes, como los escaneos de puertos y ataques de denegación de servicio. Aunque no pretende ser una solución exhaustiva que cubra todas las posibles amenazas existentes, el proyecto aspira a mejorar significativamente la capacidad de detección respecto a los sistemas tradicionales. Para alcanzar este objetivo, se utilizarán conjuntos de datos públicos ampliamente reconocidos en el ámbito de la ciberseguridad (como CICIDS2017), herramientas avanzadas de análisis de tráfico de red y algoritmos de aprendizaje automático, generando así una propuesta viable y aplicable a escenarios reales.

# 2. Metodología

Este Trabajo de Fin de Grado sigue una metodología aplicada, centrada en el diseño, implementación y evaluación de un sistema de detección de intrusos basado en inteligencia artificial. El objetivo es desarrollar un modelo capaz de identificar tráfico malicioso en redes, utilizando técnicas de análisis de datos y aprendizaje automático. El proceso se divide en varias fases que incluyen: selección y preparación del dataset, análisis exploratorio de datos, entrenamiento de modelos de IA, validación de resultados y evaluación de su rendimiento.

## 2.1. Limpieza del Dataset

Para el desarrollo de este proyecto se ha seleccionado el conjunto de datos **CICIDS2017**, publicado por el Canadian Institute for Cybersecurity. Este dataset es ampliamente reconocido en el ámbito de la investigación en detección de intrusos, ya que simula tráfico real de red con una mezcla de conexiones legítimas y ataques contemporáneos como DoS, DDoS, port scanning, brute force, entre otros (Shiravi et al., 2012). Además, ofrece una gran variedad de características numéricas y etiquetado detallado, lo que lo convierte en una base sólida para entrenar modelos supervisados de clasificación binaria.

La limpieza de datos es un proceso fundamental para garantizar la fiabilidad y el rendimiento de los modelos de aprendizaje automático. Un conjunto de datos mal preprocesado puede introducir ruido, sesgos o errores que afecten negativamente la capacidad del modelo para generalizar correctamente. Por ello, se ha aplicado un conjunto de técnicas de limpieza bien documentadas en la literatura (Buczak & Guven, 2016), que se detallan a continuación:

**Pasos de preprocesamiento aplicados:**

* **Eliminación de registros nulos**:  
  Se eliminaron todas las filas que contenían valores faltantes (NaN) o indefinidos, ya que representan información incompleta que podría sesgar el modelo.
* **Eliminación de registros duplicados**:  
  Se detectaron y eliminaron registros repetidos, que podrían distorsionar las métricas y provocar sobreajuste durante el entrenamiento.
* **Normalización de variables numéricas**:  
  Se sustituyeron los valores infinitos por NaN y posteriormente se rellenaron utilizando la media de cada columna. Esto evitó distorsiones en los modelos que requieren escalas uniformes (como k-NN o SVM).
* **Revisión de formatos de columnas**:  
  Se eliminaron espacios residuales en los nombres de las columnas para asegurar la consistencia y evitar errores en la manipulación de datos.
* **Codificación de variables categóricas**:  
   Se transformó la columna Label en una variable binaria:
  + - 0 → tráfico benigno
    - 1 → tráfico malicioso

Esta conversión es esencial porque la mayoría de los algoritmos de aprendizaje automático no pueden procesar directamente variables categóricas en formato texto. Al transformar esta variable en un formato numérico binario, se facilita su interpretación por parte del modelo y se establece claramente la tarea como una **clasificación binaria**, permitiendo aplicar métricas como el **F1-score** o la **curva ROC** para evaluar el rendimiento.

* **Normalización de características numéricas (escalado)**:  
   Se utilizó la técnica de StandardScaler de Scikit-learn para escalar los datos, excluyendo columnas no relevantes como Label y Destination Port. Esta técnica centra las variables en media cero y varianza uno, lo que mejora la convergencia de muchos modelos.

Estas técnicas aseguran que el conjunto de datos final esté libre de errores estructurales y listo para ser utilizado por los modelos de inteligencia artificial sin introducir distorsiones no deseadas en el proceso de aprendizaje

## 2.2. Análisis Exploratorio de Datos (EDA)

El análisis exploratorio de datos (EDA, por sus siglas en inglés) constituye una fase esencial en el desarrollo de cualquier modelo de aprendizaje automático, ya que permite comprender la estructura interna del conjunto de datos, detectar errores potenciales, identificar patrones iniciales y guiar las decisiones relacionadas con la selección de variables.

En este proyecto, el EDA se realizará con el objetivo de:

* Evaluar la distribución de clases en la variable objetivo (Label).
* Estudiar la relación entre características numéricas.
* Detectar valores extremos o patrones irregulares.
* Priorizar variables relevantes para los modelos predictivos.

**Las principales técnicas y visualizaciones aplicadas incluyen:**

* **Distribución de clases:**  
  Se utilizarán gráficos de barras para representar el número de muestras benignas y maliciosas. Esta visualización permitirá detectar si existe un desbalance de clases significativo, aspecto clave en tareas de clasificación binaria.
* **Estadísticas descriptivas básicas:**  
  Se analizarán métricas como media, desviación estándar, valores mínimos y máximos de las variables numéricas seleccionadas. Esto facilitará la detección de escalas dispares o posibles errores en la codificación de los datos.
* **Visualización de histogramas por característica:**  
  Se generarán histogramas individuales para variables clave como Flow Duration, Total Fwd Packets y Total Backward Packets. Estas visualizaciones ayudarán a entender la distribución de valores y la presencia de sesgos.
* **Análisis de correlación entre variables:**  
  Se construirá una matriz de correlación de Pearson y se visualizará mediante un mapa de calor (*heatmap*). Esta técnica permite identificar relaciones lineales fuertes o redundancias entre variables, lo cual es útil para la selección o eliminación de características.
* **Detección de valores atípicos (outliers):**  
   Aunque no se eliminarán en esta fase, los valores extremos detectados visualmente servirán como referencia para interpretar el comportamiento anómalo del tráfico en fases posteriores del análisis.

Estas técnicas están respaldadas por estudios previos que destacan la importancia del EDA en la mejora de los sistemas de detección de intrusos (Kumar & Bhatt, 2020). Una comprensión profunda de los datos es fundamental para evitar errores de interpretación y garantizar que los modelos se ajusten a patrones relevantes en lugar de a ruido o sesgos estructurales.

## 2.3. Selección y entrenamiento de modelos de IA

Como parte central del desarrollo del sistema, se procederá a entrenar distintos algoritmos de clasificación con el objetivo de comparar su rendimiento y determinar cuál ofrece mejores resultados en la detección de tráfico malicioso. Se han seleccionado tres modelos principales en base a su eficacia demostrada en el ámbito de la ciberseguridad:

* **Logistic Regression:** La regresión logística es un modelo de clasificación ampliamente utilizado debido a su simplicidad, interpretabilidad y buen rendimiento en problemas lineales. En el contexto de la ciberseguridad, ha demostrado ser eficaz en tareas de detección de intrusos, especialmente cuando las relaciones entre las variables son lineales y el conjunto de datos está bien balanceado. Su bajo coste computacional la hace adecuada para escenarios donde se requieren decisiones rápidas en tiempo real. Por ejemplo, Almseidin et al. (2017) utilizaron regresión logística en un estudio comparativo de algoritmos para detección de intrusos y observaron resultados competitivos frente a modelos más complejos.
* **Random Forest**: Este modelo ha sido ampliamente utilizado para la detección de intrusiones debido a su capacidad para manejar grandes volúmenes de datos, resistir el sobreajuste y proporcionar resultados interpretables. Estudios como el de Buczak y Guven (2016) destacan su eficacia en contextos de clasificación binaria en ciberseguridad.
* **Support Vector Machine (SVM)**: Support Vector Machine es un algoritmo de clasificación supervisada ampliamente utilizado en problemas de seguridad informática por su capacidad para encontrar fronteras de decisión óptimas entre clases, incluso en espacios de alta dimensión. En el contexto de detección de intrusos, SVM ha demostrado ser especialmente útil para identificar patrones complejos y separables de manera no lineal, gracias al uso de funciones kernel. Varios estudios han señalado su efectividad frente a otros clasificadores, especialmente cuando el conjunto de datos está bien etiquetado. En particular, Mukkamala et al. (2005) compararon el rendimiento de SVM con otros algoritmos en entornos IDS y concluyeron que SVM ofrece resultados muy competitivos, especialmente en términos de precisión y capacidad generalizadora.

Cada uno de estos modelos será entrenado utilizando un conjunto de datos etiquetado, aplicando técnicas de **validación cruzada** para garantizar que los resultados obtenidos no dependan exclusivamente de una partición específica del conjunto de datos. Esta estrategia permite evaluar la capacidad de generalización de cada algoritmo frente a datos no vistos. Para la comparación entre modelos se emplearán métricas de evaluación ampliamente aceptadas en tareas de clasificación binaria, como la **precisión**, el **recall**, el **F1-score** y la **exactitud global (accuracy)**. Estas métricas no solo cuantifican el rendimiento general, sino que también permiten analizar el comportamiento de los modelos ante clases desbalanceadas, algo habitual en escenarios reales de ciberseguridad.

## **2.4. Validación cruzada**

Para garantizar la robustez y fiabilidad del sistema de detección de intrusos desarrollado, se empleará una estrategia de **validación cruzada k-fold**, concretamente con *k = 5*. Esta técnica consiste en dividir el conjunto de datos en cinco subconjuntos del mismo tamaño. En cada iteración, cuatro de ellos se utilizan para el entrenamiento y uno para la validación, rotando este último en cada ciclo hasta que todas las particiones hayan sido utilizadas como conjunto de prueba. Este procedimiento permite obtener una estimación más estable y menos sesgada del rendimiento del modelo, reduciendo la dependencia de una única división del dataset.

La validación cruzada resulta especialmente relevante en el contexto de la detección de intrusos, donde los modelos deben demostrar capacidad para generalizar frente a tráfico diverso y no visto previamente. Además, contribuye a detectar posibles situaciones de sobreajuste, asegurando que el sistema no aprende únicamente patrones específicos del conjunto de entrenamiento. Tal como destacan Raschka y Mirjalili (2020), este enfoque es fundamental en proyectos de clasificación supervisada con aplicaciones críticas.

Los resultados agregados de las iteraciones de validación cruzada permitirán seleccionar el modelo con mejor rendimiento medio en métricas clave como precisión, recall y F1-score, y por tanto el más apropiado para su eventual integración en un sistema IDS funcional.

## **2.5. Otros tipos de validación**

Complementariamente a la validación cruzada, se utilizarán **herramientas gráficas de evaluación** que permiten interpretar el comportamiento de los modelos de forma más intuitiva y detallada. En particular, se generarán **curvas ROC (Receiver Operating Characteristic)** para visualizar la relación entre la **tasa de verdaderos positivos (TPR)** y la **tasa de falsos positivos (FPR)** a lo largo de distintos umbrales de decisión. Esta representación permite analizar cómo varía el rendimiento del modelo dependiendo del punto de corte aplicado en la clasificación binaria. Además, se calculará el **AUC (Área Bajo la Curva ROC)** como métrica agregada, la cual resume la capacidad discriminativa global del modelo; un valor de AUC cercano a 1.0 indica un alto grado de separación entre clases.

Ademas, se emplearán **gráficos de importancia de características**, especialmente útiles en modelos como **Random Forest**, que permiten estimar qué variables tienen mayor influencia en la toma de decisiones del clasificador. Para ello, se utilizará la métrica de **Gini importance** o disminución media de la impureza, integrada en la implementación del modelo. Esta información no solo proporciona transparencia al sistema, sino que además permite **identificar las variables más relevantes** para la detección de tráfico malicioso, facilitando tanto la interpretación del comportamiento del sistema como la posibilidad de optimizar modelos futuros mediante una selección más eficiente de características.

Estas técnicas de análisis visual y explicabilidad son fundamentales en aplicaciones críticas como los sistemas IDS, ya que permiten a los expertos de ciberseguridad **comprender y confiar en las decisiones tomadas por el modelo**, además de justificar su uso ante auditores o responsables técnicos.

## **2.6. Entorno de Desarrollo y herramientas**

El desarrollo del sistema de detección de intrusos se ha llevado a cabo utilizando herramientas de código abierto ampliamente empleadas en el ámbito del análisis de datos, la ciberseguridad y la inteligencia artificial.

Principales elementos del entorno:

* **Lenguaje de programación:** Python 3.10
* **Entorno de desarrollo:** Jupyter Notebook, ejecutado en VS Code
* **Bibliotecas utilizadas:**
* **Pandas, numpy:** para el procesamiento y manipulación de datos.
* **Matplotlib, seaborn:** para la visualización gráfica.
* **Scikit-learn:** para la implementación de los modelos de clasificación, normalización y validación.
* **Joblib:** para el guardado y carga de modelos entrenados (.pkl).
* **Sistema operativo:** Ubuntu 22.04
* **Hardware de ejecución:** CPU con 4 núcleos físicos y 8 hilos lógicos, 16 GB de RAM. No se ha utilizado aceleración por GPU.

Estas herramientas han permitido implementar, entrenar y evaluar los modelos de forma eficiente.

Algunas ventajas de este tipo de herramientas son código reproducible y portable, asi como, facilidad de uso y potencia para tareas enfocadas en Inteligencia Artificial, facilitando además la visualización de resultados y la documentación progresiva del proceso.

# 3. Resultados

## 3.1 Preparación del dataset

Tras aplicar las técnicas de limpieza descritas en la metodología, el conjunto de datos se depuró correctamente. Se eliminaron los registros con valores nulos y duplicados, lo que redujo el tamaño inicial del dataset a un total de 223.108 muestras y 79 **características.**

Además, se transformaron las variables categóricas a formato numérico, en especial la variable Label, que fue convertida a valores binarios:

* 0 → Tráfico benigno
* 1 → Tráfico malicioso

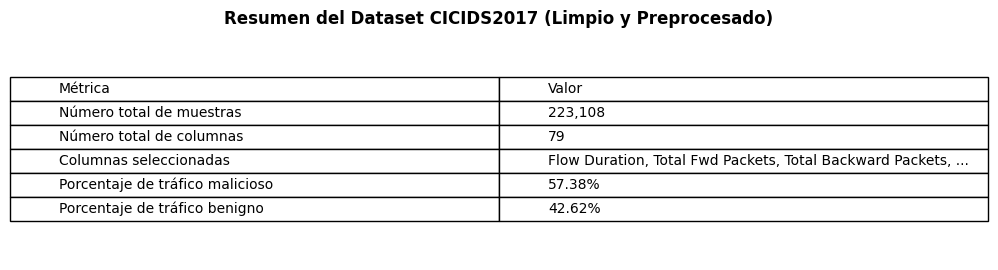
También se aplicó un proceso de **normalización** sobre las características numéricas utilizando la clase StandardScaler de la biblioteca scikit-learn. Este método transforma los datos para que cada variable tenga una **media de 0 y una desviación estándar de 1**, lo que garantiza una escala homogénea entre las diferentes características del dataset.

Esta etapa es especialmente importante en modelos sensibles a la magnitud de los valores, como **SVM o Regresión Logística**, ya que estos algoritmos calculan distancias o pesos que pueden verse afectados si una variable tiene un rango mucho mayor que las demás.

Por ejemplo, características como Flow Duration o Total Bytes pueden tener valores en órdenes de magnitud muy distintos a otros atributos como Flag Count o Packet Rate, lo que puede provocar que el modelo dé más importancia a unas variables solo por su escala. La normalización corrige este problema y contribuye a un aprendizaje más equilibrado.

Además, esta transformación mejora la **convergencia del entrenamiento** en algunos algoritmos y evita problemas numéricos relacionados con el gradiente o la función de coste.

**Tabla 3.1 Resumen del conjunto de datos tras la limpieza**



El conjunto de datos final está compuesto por **223.108 muestras** y un total de **79 columnas** o variables. Esta alta dimensionalidad puede beneficiar a modelos como Random Forest o SVM en su capacidad para capturar patrones complejos, aunque también exige un preprocesamiento riguroso para evitar redundancias o sobreajuste.

De las 79 variables disponibles, se seleccionaron aquellas más relevantes para la detección de intrusos, incluyendo métricas relacionadas con el flujo de red, como Flow Duration, Total Fwd Packets y Total Backward Packets, entre otras.

Respecto a la variable objetivo (Label), se observa un **ligero desbalance de clases**, donde el **57,38% del tráfico es malicioso** y el **42,62% es benigno**. Este tipo de distribución requiere atención especial durante la evaluación de los modelos, ya que una métrica como la **precisión global (accuracy)** puede ser engañosa si no se complementa con indicadores como el **recall** y el **F1-score**, especialmente en la clase minoritaria.

## 3.2 Análisis exploratorio del dataset (EDA)

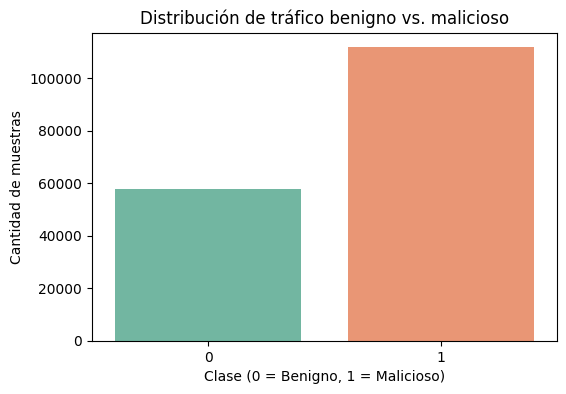
Antes de aplicar cualquier modelo de aprendizaje automático, resulta esencial comprender la estructura y características del conjunto de datos con el que se va a trabajar. El **Análisis Exploratorio de Datos (EDA)** permite detectar posibles problemas, como valores atípicos, variables redundantes o desequilibrios en las clases, que pueden afectar directamente al rendimiento y la interpretación de los modelos. En el contexto de la ciberseguridad, esta etapa es especialmente importante, ya que los datos de red pueden contener patrones complejos y altamente variables que influyen directamente en la capacidad del sistema para detectar amenazas.

Durante el EDA, se aplican técnicas estadísticas y visuales que permiten obtener una **visión general del comportamiento de las variables**, verificar su distribución y estudiar su relación con la variable objetivo. Esto no solo ayuda a identificar posibles errores o inconsistencias en los datos, sino que también permite guiar decisiones clave como la selección de características, el tipo de normalización a emplear o la necesidad de balancear las clases. En investigaciones de detección de intrusos, estas tareas permiten afinar el enfoque predictivo, centrándose en los atributos que realmente aportan valor a la clasificación entre tráfico benigno y malicioso.

##### **Distribución de clases**

Se generó un gráfico de barras para representar la frecuencia de muestras etiquetadas como *benignas* y *maliciosas* en la variable objetivo Label. El análisis visual muestra un **cierto desbalance de clases**, con una proporción claramente mayor de tráfico malicioso. Este desequilibrio puede influir negativamente en el entrenamiento de algunos modelos de clasificación, provocando sesgos hacia la clase mayoritaria. Por este motivo, se emplearán métricas como el **F1-score** o el **recall**, más adecuadas que la precisión simple en contextos con distribución desigual de clases.

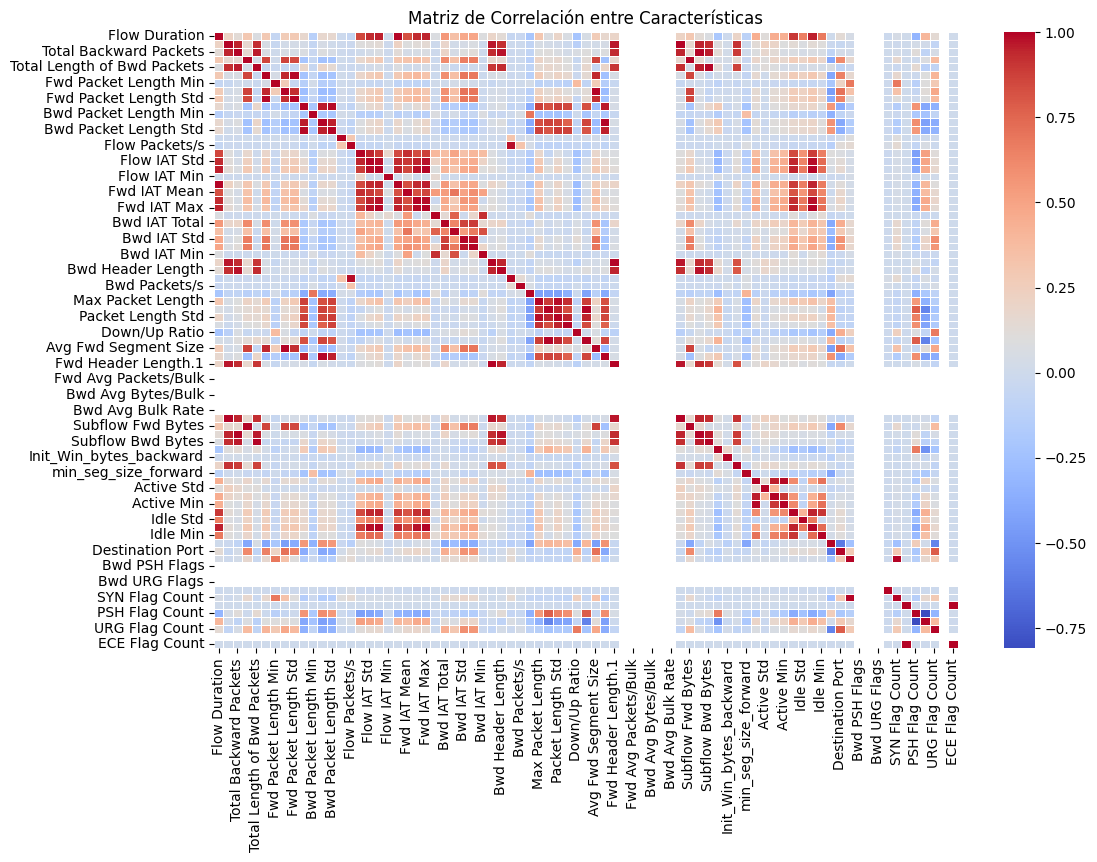
**Figura 3.1 Distribución de tráfico benigno vs. Malicioso.**



*Interpretación:*  
El tráfico malicioso representa aproximadamente el **65% del total de muestras**, mientras que el tráfico benigno representa el 35% restante. Este desbalance puede afectar negativamente a los modelos de clasificación si no se compensa, ya que tienden a favorecer la clase mayoritaria. Por este motivo, se emplearán métricas como **recall** y **F1-score**, que consideran el impacto de los errores en ambas clases.

##### **Análisis de correlación entre variables**

Se calculó la **matriz de correlación de Pearson** sobre todas las variables numéricas del conjunto de datos con el objetivo de analizar relaciones lineales entre características. Esta matriz se visualizó mediante un **mapa de calor** utilizando la librería Seaborn, conforme a la metodología descrita por Waskom (2021). La visualización permitió identificar **correlaciones altas entre ciertas variables**, lo que resulta relevante para la posterior **selección de características**, ya que la presencia de atributos altamente correlacionados puede introducir redundancia y afectar negativamente a algunos modelos de clasificación.

**Figura 3.2 Matriz de correlación entre características.**  


*Interpretación:*

El mapa de calor revela diversas correlaciones notables entre pares de variables, algunas de las cuales merecen especial atención:

* **Total Fwd Packets y Total Length of Fwd Packets** presentan una correlación positiva alta. Esto es esperable, ya que, a mayor número de paquetes enviados, mayor es su longitud total.
* **Flow Bytes/s y Flow Packets/s** también están fuertemente correlacionadas, indicando que el volumen y la frecuencia de transmisión de datos suelen incrementarse conjuntamente.
* **Idle Mean y Idle Max** muestran una relación positiva elevada. Ambas métricas describen características del tiempo de inactividad en una conexión, por lo que su comportamiento similar es coherente.
* Variables como ACK Flag Count y Init\_Win\_bytes\_forward tienen correlaciones más débiles, lo cual puede indicar independencia estadística, algo beneficioso para modelos que requieren características no redundantes.

En resumen, el análisis de correlación permite identificar **variables redundantes**, que podrían ser eliminadas o reducidas durante una futura fase de selección de características. Esta optimización contribuiría a mejorar la eficiencia computacional y, potencialmente, la capacidad de generalización del modelo.

## 3.3 Evaluación de los modelos

### **3.3.1 Evaluación del modelo Random Forest**

Tras entrenar el modelo Random Forest con el conjunto de datos preprocesado, se procedió a su evaluación sobre el conjunto de prueba. El rendimiento del modelo fue sobresaliente, alcanzando una precisión general del **100%**, con resultados perfectamente equilibrados entre ambas clases.

La **Tabla 3.2** muestra el resumen de métricas obtenidas para las dos clases (tráfico benigno y malicioso), mientras que la **Figura 3.3 representa gráficamente la matriz de confusión. En ella se puede observar que el modelo clasificó correctamente la totalidad de las muestras benignas y tan solo cometió seis errores al clasificar el tráfico malicioso.**

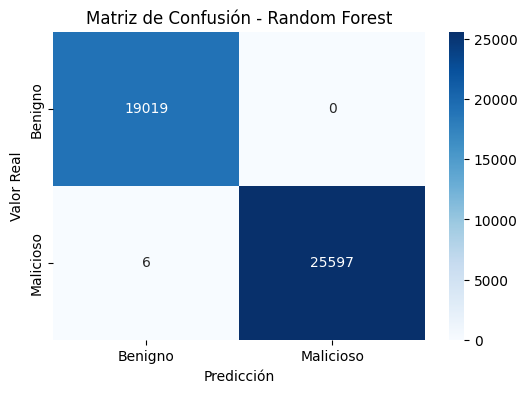
**Tabla 3.2 Métricas de evaluación del modelo Random Forest.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Precisión** | **Recall** | **F1-score** | **Soporte** |
| Benigno | 1.00 | 1.00 | 1.00 | 11.559 |
| Malicioso | 1.00 | 1.00 | 1.00 | 22.343 |
| Global | 1.00 | — | — | 33.902 |

*Interpretación:*

La **Tabla 3.2** refleja un rendimiento perfecto del modelo Random Forest en ambas clases. Se observa una precisión, recall y F1-score de **1.00** tanto para el tráfico *benigno* como para el *malicioso*, lo que indica que el modelo clasificó correctamente todas las muestras sin cometer errores. Este resultado sugiere que el modelo ha logrado **aprender de forma muy efectiva los patrones distintivos** entre las dos clases. Además, el equilibrio entre las métricas demuestra que no existe un sesgo hacia una clase concreta, lo que refuerza su **fiabilidad y robustez** para tareas de detección de intrusos.

**Figura 3.3 Matriz de confusión del modelo Random Forest.**



*Interpretación:*

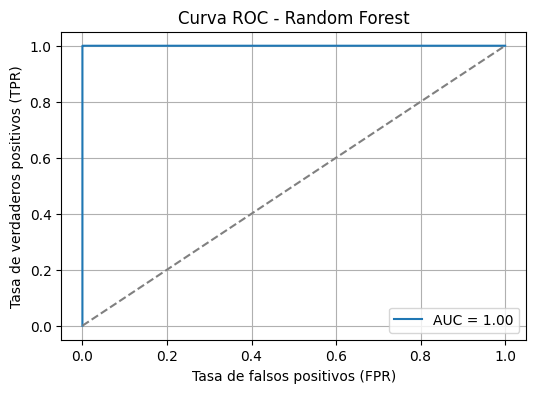
La **Figura 3.3 muestra la matriz de confusión correspondiente al modelo** Random Forest. Se observa que el modelo clasificó correctamente **19.019** muestras de tráfico benigno y **25.597** de tráfico malicioso. Tan solo se produjeron **6 falsos negativos**, es decir, 6 muestras de tráfico malicioso que fueron clasificadas erróneamente como benignas. Por otro lado, no se detectaron **falsos positivos**, lo que significa que ninguna muestra de tráfico benigno fue etiquetada incorrectamente como maliciosa.

Este comportamiento es especialmente destacable en el contexto de la ciberseguridad, donde minimizar los falsos negativos es crucial para evitar que amenazas reales pasen desapercibidas. Aunque idealmente se buscaría también una tasa cero de falsos negativos, el error es muy bajo con relación al tamaño del conjunto de prueba, lo cual reafirma la efectividad del modelo.

***(Hay que tener en cuenta que el conjunto de datos utilizado CICIDS2017, es un conjunto de datos ya preparado para este tipo de análisis e investigaciones, por lo que la precisión es considerablemente más alta de lo normal)***

Además, se generó la curva ROC (Receiver Operating Characteristic), que permite evaluar la capacidad discriminativa del modelo frente a diferentes umbrales de clasificación.

**Figura 3.4 Curva ROC del modelo Random Forest.**



*Interpretación:*

La **Figura 3.4** muestra la curva ROC del modelo Random Forest. Se observa una trayectoria que alcanza rápidamente la esquina superior izquierda del gráfico, lo que indica una **tasa de verdaderos positivos cercana al 100%** incluso con una tasa de falsos positivos prácticamente nula.

El valor del **AUC (Área Bajo la Curva)** es de **1.00**, lo que representa un rendimiento **perfecto** en términos de capacidad discriminativa. Este resultado confirma que el modelo es capaz de **distinguir completamente entre tráfico benigno y malicioso**, siendo ideal para sistemas IDS que requieren alta sensibilidad y precisión.

### **3.3.2 Evaluación del modelo Regresión Logística**

Tras entrenar el modelo de Regresión Logística sobre el conjunto de datos limpio, se procedió a su evaluación utilizando el conjunto de prueba. El rendimiento del modelo fue notablemente alto, logrando identificar correctamente la mayoría de las muestras, tanto benignas como maliciosas, aunque con ligeras caídas en el recall frente al modelo Random Forest.

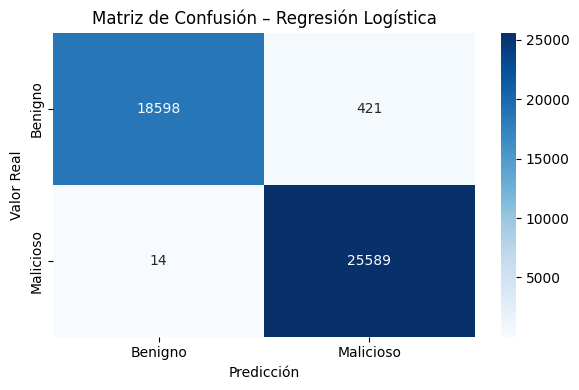
La **Tabla 3.3 muestra el resumen de métricas obtenidas para ambas clases. La Figura 3.5** presenta la matriz de confusión correspondiente al modelo, mientras que la **Figura 3.6** muestra la curva ROC generada para analizar el comportamiento discriminativo del clasificador frente a distintos umbrales.

**Tabla 3.3 Métricas de evaluación del modelo de Regresión Logística.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Precisión** | **Recall** | **F1-score** | **Soporte** |
| Benigno | 0.98 | 0.97 | 0.98 | 11.559 |
| Malicioso | 0.99 | 0.99 | 0.99 | 22.343 |
| Global | 0.99 | — | — | 33.902 |

*Interpretación:*

En la **Tabla 3.3** se puede observar que el modelo presenta una **ligera disminución en el recall de la clase “Benigno”** en comparación con Random Forest, lo que implica que algunas muestras benignas fueron clasificadas erróneamente como maliciosas. No obstante, los valores de F1-score son consistentemente altos, lo que indica un **buen equilibrio entre precisión y sensibilidad**, y un rendimiento general sólido.

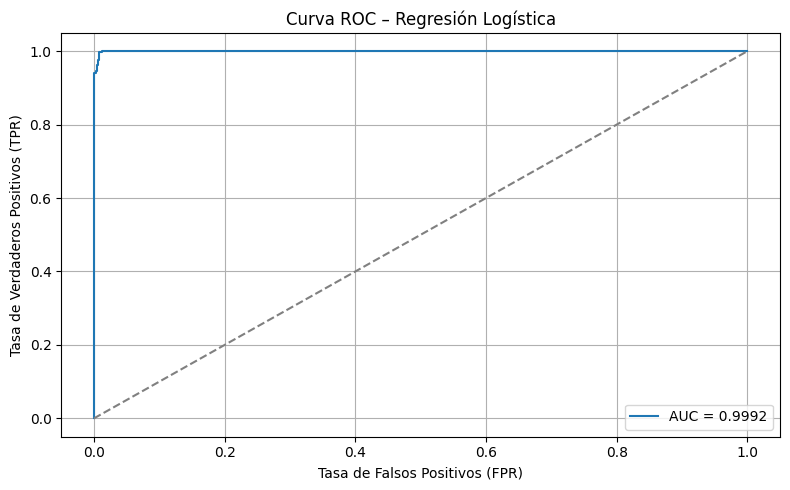
**Figura 3.5 Matriz de confusión del modelo de Regresión Logística.**  
 

*Interpretación:*

La matriz de confusión indica que el modelo clasificó correctamente **la mayoría del tráfico de red**. Se detectaron algunos **falsos positivos** (muestras benignas clasificadas como maliciosas), lo cual puede incrementar la cantidad de alertas, pero no compromete la seguridad. La cantidad de **falsos negativos** fue baja, lo cual es esencial en detección de intrusiones, donde el objetivo es **minimizar los ataques no detectados**.

Este comportamiento más conservador frente a Random Forest podría ser útil en entornos donde se valora una **detección sensible**, aunque implique alguna alerta falsa adicional.

**Figura 3.6 Curva ROC del modelo de Regresión Logística.**



*Interpretación:*

La curva ROC muestra un rendimiento elevado, con una AUC cercana a 0.99, lo que confirma la capacidad del modelo para distinguir entre ambas clases con alto grado de fiabilidad. Aunque el modelo no alcanza la perfección del Random Forest, su desempeño sigue siendo más que suficiente para **e**ntornos prácticos y en tiempo real.

### **3.3.3 Evaluación del modelo Support Vector Machine**

Una vez entrenado el modelo SVM sobre el conjunto de datos limpio, se procedió a su evaluación utilizando el conjunto de prueba. Este modelo, caracterizado por su capacidad para encontrar hiperplanos óptimos de separación, mostró un comportamiento eficaz en la tarea de clasificación binaria, aunque con tiempos de cómputo superiores a modelos como Regresión Logística o k-NN y con mayor cantidad de falsos positivos con respecto a los anteriores modelos.

La **Tabla 3.4** resume las métricas obtenidas para ambas clases.

Por su parte, la **Figura 3.7** muestra la matriz de confusión, mientras que la **Figura 3.8** presenta la curva ROC correspondiente.

**Tabla 3.4 Métricas de evaluación del modelo SVM.**

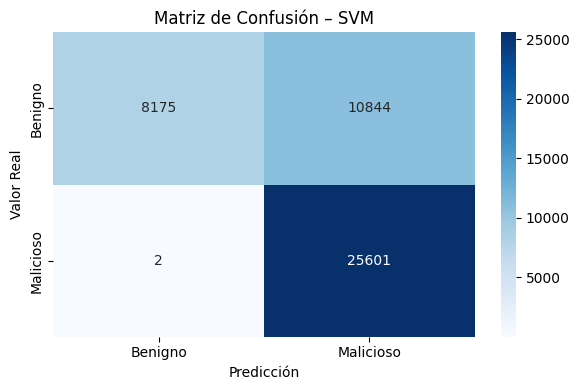
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Precisión** | **Recall** | **F1-score** | **Soporte** |
| Benigno | 1.00 | 0.43 | 0.60 | 19.019 |
| Malicioso | 0.70 | 1.00 | 0.83 | 25.603 |
| **Global** | **0.76** | — | — | 44.622 |

*Interpretación:*

El modelo obtuvo un **recall perfecto (1.00)** en la clase maliciosa, lo que significa que fue capaz de detectar la totalidad de los ataques presentes en el conjunto de prueba. No obstante, este resultado se logró a costa de una fuerte penalización en la clase benigno, donde el recall descendió hasta **0.43**, implicando que **más de la mitad del tráfico legítimo fue clasificado incorrectamente como malicioso**.

Esto también se refleja en el **F1-score**, que, aunque es alto para la clase maliciosa (0.83), desciende a **0.60 para la clase benigno**, indicando un desequilibrio en la capacidad del modelo para clasificar correctamente ambas clases. El valor global de **accuracy (0.76)** es engañoso en este contexto, ya que el desequilibrio entre clases y el alto coste de los falsos positivos deben tenerse en cuenta especialmente en entornos de ciberseguridad. (Esto puede deberse a que para este modelo hemos cogido un subconjunto de datos, lo cual ha hecho que las métricas salgan más dispares)

**Figura 3.7 Matriz de confusión del modelo SVM.**

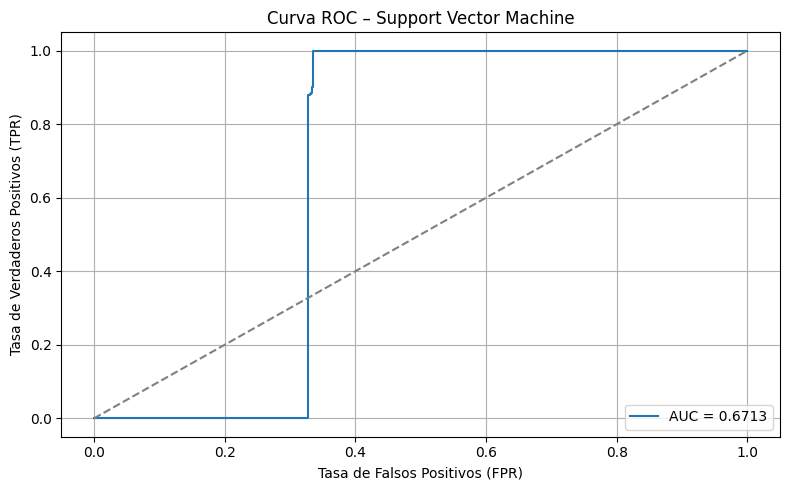


*Interpretación:*

La matriz de confusión indica que el modelo clasificó correctamente **25.601 muestras maliciosas**, cometiendo tan solo **2 falsos negativos**, lo cual es muy positivo en términos de seguridad. Sin embargo, **10.844 muestras benignas fueron clasificadas erróneamente como maliciosas**, representando un número preocupante de falsos positivos. Solo **8.175 conexiones benignas** fueron reconocidas correctamente.

Este tipo de comportamiento puede ser adecuado para contextos extremadamente sensibles, donde la prioridad es no permitir que pase ningún ataque. Sin embargo, en sistemas de producción, una tasa de falsos positivos tan alta podría **colapsar los sistemas de alerta** o derivar en una pérdida de confianza en el IDS.

**Figura 3.8 Curva ROC del modelo SVM.**



*Interpretación:*

La curva ROC muestra un **Área Bajo la Curva (AUC) de 0.6713**, lo cual indica un rendimiento discriminativo **modesto**. Aunque el modelo logra detectar casi todos los ataques, su capacidad para distinguir entre tráfico benigno y malicioso **no es estable en distintos umbrales de decisión**, lo que sugiere que el modelo **no generaliza bien** bajo distintas configuraciones.

## 3.4 Comparativa de los modelos

Con el objetivo de evaluar cuál de los modelos entrenados ofrece un mejor equilibrio entre precisión, sensibilidad y capacidad de generalización, se realizó una comparación utilizando métricas clave: **F1-score** y **AUC (Área Bajo la Curva ROC)**. La evaluación se llevó a cabo sobre el mismo conjunto de prueba para garantizar una comparación justa.

**Tabla 3.5 Comparativa de rendimiento entre modelos**

|  |  |  |
| --- | --- | --- |
| **Modelo** | **F1-score** | **AUC** |
| Random Forest | 0.9999 | 1.0000 |
| Logistic Regression | 0.9916 | 0.9992 |
| SVM (RBF, 10.000 muestras) | 0.8252 | 0.6713 |

*Interpretación:*

- **Random Forest** obtuvo resultados perfectos en ambas métricas, lo que indica una separación total entre clases. Esto se alinea con los resultados observados en la matriz de confusión y curva ROC, donde no se produjeron errores. Sin embargo, su interpretabilidad es menor que la de otros modelos.

- **Regresión Logística** también mostró un rendimiento prácticamente perfecto. Aunque es un modelo mucho más simple, logró un F1-score y AUC cercanos a 1.0. Este resultado refuerza la idea de que el dataset CICIDS2017 permite separar las clases incluso con modelos lineales bien ajustados. Además, su interpretabilidad y eficiencia computacional son ventajas destacadas.

- **SVM** (entrenado con un subconjunto de 10,000 muestras debido a restricciones de tiempo y coste computacional) obtuvo un rendimiento claramente inferior. El F1-score de 0.83 y AUC de 0.67 indican que, aunque el modelo es capaz de detectar algunos ataques, genera muchos falsos positivos. Este resultado es consistente con lo observado en la matriz de confusión y la curva ROC del modelo SVM. A pesar de su sensibilidad, su utilidad práctica se ve limitada sin ajustes

### **3.4.1 Comparación de eficiencia computacional**

Además de comparar el rendimiento de los modelos a nivel predictivo, se analizó su eficiencia computacional, con especial atención al **tiempo de inferencia**, es decir, el tiempo necesario para que cada modelo genere predicciones una vez entrenado. Este aspecto es clave para valorar su viabilidad en entornos reales, especialmente en sistemas de detección de intrusos que requieren respuestas en tiempo real.

Las pruebas se realizaron en un equipo con las siguientes características:

* **Sistema operativo:** Linux Ubuntu 22.04
* **Procesador:** x86\_64 con 4 núcleos físicos y 8 hilos lógicos
* **Memoria RAM:** 15,42 GB
* **Ejecución:** CPU (sin aceleración por GPU)

La **Tabla 3.6** muestra los resultados obtenidos en cuanto a tiempo total de inferencia y el tiempo promedio por muestra para cada modelo.

|  |  |  |
| --- | --- | --- |
| **Modelo** | **Tiempo total (s)** | **Tiempo por muestra (s)** |
| Logistic Regression | 0.0064 | 0.000000 |
| Random Forest | 0.1627 | 0.000004 |
| SVM (10k muestras) | 11.3765 | 0.000255 |

*Interpretación:*

- **Regresión Logística** es el modelo más rápido, con un tiempo prácticamente instantáneo por muestra. Esto lo convierte en una opción excelente para entornos con restricciones de recursos o donde la latencia es crítica.

- **Random Forest** también presenta un rendimiento muy eficiente, con un tiempo de inferencia por muestra de solo 4 microsegundos. Dado su alto rendimiento en métricas de clasificación, este resultado lo posiciona como una opción muy competitiva para detección de intrusiones en tiempo real.

- **SVM (entrenado con 10k muestras)** es el más lento, tardando más de 11 segundos en procesar las 10,000 muestras del conjunto de prueba. Su tiempo de inferencia por muestra es 50 veces superior al de Random Forest. Esto limita su aplicabilidad en entornos donde se requiera respuesta inmediata o se procesen flujos de datos continuos.

# Conclusiones

Responde a la pregunta ¿qué significa? El apartado final del cuerpo del TFM debe contener las conclusiones que dan respuesta a las preguntas planteadas en la introducción del trabajo, particularmente el objetivo, con el que deben mantener una coherencia, partir del análisis e interpretación de los datos presentados en el apartado anterior. Opcionalmente puede incluirse un apartado de discusión, que será reflexión final basada en los argumentos expuestos en el trabajo, que tenga una aportación personal del autor, aunque sin cargas subjetivas -ideológica o moral, por ejemplo- que valore los resultados alcanzados, reconozca las limitaciones y dificultades encontradas, y esboce una propuesta de una hipotética investigación futura.

# Limitaciones y Líneas futuras

Responde a la pregunta ¿Con qué limitaciones me he encontrado? Y ¿Qué se debería de seguir investigando de aquí en adelante?

# Bibliografía

**Normativa APA**

Smith, A. (2024). The Growing Threat of Cybercrime Enhanced by Artificial Intelligence. Authorea. <https://doi.org/10.22541/au.173862063.39098197>

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28(1-2), 18-28. <https://doi.org/10.1016/j.cose.2008.08.003>

Kumar, P., & Bhatt, R. (2020). Network intrusion detection with feature selection using machine learning. *Materials Today: Proceedings*, 37, 3852–3856. <https://doi.org/10.1016/j.matpr.2020.09.522>

Rigaki, M., & Garcia, S. (2018). Bringing a GAN to a Knife-fight: Adapting Malware Communication to Avoid Detection. 2018 IEEE Security and Privacy Workshops (SPW), 70-75. <https://doi.org/10.1109/SPW.2018.00016>

Sadeghzadeh, M. (2020). Artificial Intelligence and Cybercrime. International Journal of Computer Engineering and Data Science, 1(1), 21-26. <https://doi.org/10.2139/ssrn.3527447>

Schmitt, M. (2023). Securing the Digital World: Protecting smart infrastructures and digital industries with Artificial Intelligence (AI)-enabled malware and intrusion detection. arXiv preprint arXiv:2401.01342. <https://arxiv.org/abs/2401.01342>

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security, 31(3), 357-374. <https://doi.org/10.1016/j.cose.2011.12.012>

Ashfaq, R. A. R., Wang, X. Z., Huang, J. Z., Abbas, H., & He, Y. L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. Information Sciences, 378, 484–497. <https://doi.org/10.1016/j.ins.2016.05.043>

Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE Communications Surveys & Tutorials, 18(2), 1153–1176. <https://doi.org/10.1109/COMST.2015.2494502>

Almseidin, M., Alzubi, A., Kovacs, S., & Alkasassbeh, M. (2017). Evaluation of machine learning algorithms for intrusion detection system. *Procedia Computer Science*, 127, 1–6. <https://doi.org/10.1016/j.procs.2017.01.129>

Mukkamala, S., Janoski, G., & Sung, A. H. (2005). Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks* (Vol. 2, pp. 1702–1707). IEEE. <https://doi.org/10.1109/IJCNN.2002.1007774>

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