UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TÍTULO

Comparativa de algoritmos de aprendizaje automático para la detección de tipos específicos de malware en entornos adversos

AUTOR

Guale Rosales, Luis Gabriel

TRABAJO DE TITULACIÓN

Previo a la obtención del grado académico en
MAGÍSTER EN CIBERSEGURIDAD

TUTOR

Quirumbay Yagual, Daniel Iván

Santa Elena, Ecuador

Año 2025

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

TRIBUNAL DE SUSTENTACIÓN

Firmado electrónicamente por:
ALICIA GERMANIA ANDRADE VERA
Validar únicamente con FirmaEC

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Firmado electrónicamente por:
DANIEL IVAN QUIRUMBAY YAGUAL
Validar únicamente con FirmaEC

Lic. Daniel Quirumbay Yagual, Mgtr.
TUTOR

Firmado electrónicamente por:
JAIME BENJAMIN OROZCO IGUASNIA
Validar únicamente con FirmaEC

Ing. Jaime Orozco Iguasnia, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
CARLOS ANDRES CASTILLO YAGUAL
Validar únicamente con FirmaEC

Ing. Carlos Castillo Yagual, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MARIA MARGARITA RIVERA GONZALEZ
Validar únicamente con FirmaEC

Abg. María Rivera González, Mgtr.
SECRETARIA GENERAL
UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por LUIS GABRIEL GUALE ROSALES, como requerimiento para la obtención del título de Magíster en Ciberseguridad.

TUTOR

Firmado electrónicamente por:
DANIEL IVAN QUIRUMBAY YAGUAL
Validar únicamente con FirmaEC

Lic. Daniel Quirumbay Yagual, Mgtr.

Santa Elena, 30 de septiembre de 2025

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

DECLARACIÓN DE RESPONSABILIDAD

Yo, **LUIS GABRIEL GUALE ROSALES**

DECLARO QUE:

El trabajo de Titulación, Comparativa de algoritmos de aprendizaje automático para la detección de tipos específicos de malware en entornos adversos previo a la obtención del título en Magíster en Ciberseguridad, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría. En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Santa Elena, 30 de septiembre de 2025

EL AUTOR

Firmado electrónicamente por:
LUIS GABRIEL GUALE ROSALES
Validar únicamente con FirmaEC

Luis Gabriel Guale Rosales

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado Comparativa de algoritmos de aprendizaje automático para la detección de tipos específicos de malware en entornos adversos, presentado por el estudiante, Guale Rosales Luis Gabriel fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 7%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.


CERTIFICADO DE ANÁLISIS
magister

Maestria_Ciber_LUIS GABRIEL GUALE ROSALES

7% Textos sospechosos

Nombre del documento: Maestria_Ciber_LUIS GABRIEL GUALE ROSALES.docx
ID del documento: 79da30e4e3c3be6aa3e06d08d60a6f8a1a2c4d56
Tamaño del documento original: 4,58 MB

Depositante: DANIEL IVAN QUIRUMBAY YAGUAL
Fecha de depósito: 16/9/2025
Tipo de carga: interface
fecha de fin de análisis: 16/9/2025

TUTOR

Firmado electrónicamente por:
DANIEL IVAN QUIRUMBAY YAGUAL
Validar únicamente con FirmaEC

Lic. Daniel Quirumbay Yagual, Mgtr.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES
INSTITUTO DE POSTGRADO

AUTORIZACIÓN

Yo, **Guale Rosales Luis Gabriel**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de mi trabajo de propuestas metodológicas y tecnológicas avanzadas con fines de difusión pública, además apruebo la reproducción de este artículo académico dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor

Santa Elena, 30 de septiembre de 2025

EL AUTOR

Firmado electrónicamente por:
LUIS GABRIEL GUALE ROSALES
Validar únicamente con FirmaEC

Luis Gabriel Guale Rosales

AGRADECIMIENTO

En primer lugar, elevo mi más profunda gratitud a Dios por guiarme, darme la fortaleza, la sabiduría y la perseverancia necesarias para culminar esta etapa de mi formación profesional. A mi tutor, el Lic. Daniel Quirumbay, expreso mi sincero agradecimiento por su invaluable orientación, paciencia y dedicación. Su experiencia y conocimientos fueron cruciales para el desarrollo y la finalización de esta investigación. Mi más profundo reconocimiento a la Universidad Estatal Península de Santa Elena, sus autoridades y al cuerpo docente, por la formación de excelencia brindada. A la administración de la Fundación Salinas Yacht Club, mi gratitud por su colaboración y apoyo, brindándome las facilidades, información y equipos necesarios para la realización de este trabajo. Finalmente, este logro no hubiera sido posible sin el apoyo incondicional de mi familia. A mis padres, por su apoyo constante, por creer en mí y por ser mi pilar durante todo este proceso. A mi amada esposa, por su comprensión y paciencia, especialmente en los momentos en que la exigencia de este trabajo me obligó a priorizarlo sobre otros compromisos. Su amor y apoyo fueron mi mayor motivación.

Luis Gabriel, Guale Rosales

DEDICATORIA

Dedico este trabajo a mis padres, Felipa y Félix, quienes con su amor incondicional y su constante motivación han sido el fundamento de mi vida y la fuerza que me ha impulsado a alcanzar este logro. Sin su apoyo, este camino no habría sido posible. A Kyara Franco, por brindarme su ayuda para iniciar este reto. A mi amada esposa, Betty Reyes, y a mi hijo, Gabriel Guale, por su paciencia, comprensión y por estar siempre a mi lado. Sus palabras de ánimo y su cariño me dieron la energía para seguir adelante, incluso en los momentos más difíciles. Este trabajo es el reflejo de su apoyo incondicional y de su amor.

Luis Gabriel, Guale Rosales

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo analiza la eficacia de diferentes algoritmos de aprendizaje automático aplicados a la detección de malware, con énfasis en la precisión, eficiencia computacional y robustez frente a amenazas. El objetivo principal fue comparar modelos como Árbol de Decisión, Random Forest y XGBoost en la clasificación de distintos tipos de malware (ransomware, spyware y troyanos). En cuanto a la metodología engloba un proceso estructurado de preprocesamiento de datos, selección de características, entrenamiento de modelos y evaluación mediante métricas de desempeño, matrices de confusión, curvas ROC y análisis de eficiencia computacional. En los resultados se observa que los algoritmos ensamblados, particularmente Random Forest y XGBoost, llegaron a mayores niveles de precisión y generalización, superando al Árbol de Decisión en ámbitos de validación. Se concluye que estos modelos representan alternativas viables para fortalecer los sistemas de seguridad informática, combinando buen desempeño predictivo con una eficiencia adecuada en procesos de detección automatizada.

Palabras clave: malware, aprendizaje automático, clasificación.

ABSTRACT

This paper analyzes the effectiveness of different machine learning algorithms applied to malware detection, with an emphasis on accuracy, computational efficiency, and robustness against threats. The main objective was to compare models such as Decision Tree, Random Forest, and XGBoost in the classification of different types of malware (ransomware, spyware, and Trojans). The methodology encompasses a structured process of data preprocessing, feature selection, model training, and evaluation using performance metrics, confusion matrices, ROC curves, and computational efficiency analysis. The results show that the assembled algorithms, particularly Random Forest and XGBoost, achieved higher levels of accuracy and generalization, outperforming Decision Tree in validation areas. It is concluded that these models represent viable alternatives for strengthening computer security systems, combining good predictive performance with adequate efficiency in automated detection processes.

Keywords: malware, machine learning, classification.

INTRODUCCIÓN

La creciente digitalización de los procesos en todos los ámbitos de la sociedad ha traído consigo un incremento en la exposición a riesgos cibernéticos, entre ellos el malware, considerado una de las principales amenazas para la seguridad informática. Este tipo de software malicioso, en sus diversas variantes como ransomware, spyware y troyanos, afecta tanto a individuos como a organizaciones, generando pérdidas económicas, vulneración de datos sensibles y debilitamiento de la confianza en los sistemas tecnológicos. De acuerdo a la realidad a la que nos enfrentamos, nace el impulso de adoptar herramientas de detección cada vez más preciso y efectivo, suficientemente capaces de anticiparse a la sofisticación de los ataques.

En este contexto, el aprendizaje automático se muestra como una solución potente para el reto que supone la detección de malware, es decir, en este trabajo se define que los algoritmos de aprendizaje automático son capaces no solo de identificar patrones de comportamiento anómalos con gran precisión, sino también de demostrar eficiencia computacional y resiliencia en entornos adversos.

El objetivo principal de esta investigación es realizar un análisis comparativo de la eficacia de varios algoritmos de aprendizaje automático concretamente, Random Forest, SVM con un núcleo RBF y XGBoost en la detección de malware. Esta evaluación se enmarca en tres dimensiones críticas como la precisión predictiva, eficiencia computacional y robustez, además, el alcance del estudio se limita al análisis de estos modelos aplicados a un conjunto de datos compuesto por características de malware reales, incluyendo tanto metadatos como características de comportamiento observadas en entornos virtualizados.

De esta manera, lo que se pretende es identificar qué algoritmos resultan más adecuados para integrarse en sistemas de seguridad informática que demanden un balance entre precisión predictiva, rapidez de ejecución y capacidad de generalización ante distintas amenazas.

El presente trabajo se organiza en cuatro capítulos principales. En el Capítulo I se desarrolla el marco teórico y referencial, donde se revisan los fundamentos de la seguridad informática, el concepto de malware y el papel de la inteligencia artificial en la

ciberseguridad. En el Capítulo II se trata la metodología aplicada, en donde se ve a profundidad el diseño de la investigación, la naturaleza de los datos, el preprocesamiento aplicado y los criterios de evaluación de los modelos. El Capítulo III muestra los resultados obtenidos en base a la implementación y comparación de los algoritmos seleccionados, complementado de métricas de desempeño, matrices de confusión y análisis de eficiencia. Por último, el Capítulo IV está estructurado con la discusión, conclusiones y recomendaciones, en donde se analiza los hallazgos y se plantea futuras líneas de investigación.

Planteamiento de la investigación (Fundamentación de la investigación)

El malware evoluciona constantemente y los métodos tradicionales de detección resultan insuficientes para enfrentar amenazas más sofisticadas y adaptativas. Esto exige explorar herramientas innovadoras que permitan anticiparse a dichas amenazas con mayor precisión y rapidez. En este sentido, el aprendizaje automático constituye una alternativa prometedora, al posibilitar la detección de patrones ocultos que escapan al análisis manual o a las reglas estáticas de los sistemas antivirus convencionales.

El motivo de este estudio es sencillo, pero significativo, ya que, es importante a nivel científico, social y profesional desde el punto de vista científico, contribuye al creciente debate sobre el uso del aprendizaje automático en la ciberseguridad, comparando tres algoritmos muy conocidos como el Random Forest, SVM con un núcleo RBF y el XGBoost que analizan su precisión, su velocidad de ejecución y su capacidad para gestionar diferentes tipos de malware.

Desde el punto de vista social, es importante porque ayuda a proteger a las personas y a las organizaciones de pérdidas económicas o de la exposición de sus datos, viéndolo desde el punto de vista profesional, ofrece información práctica que puede servir de guía a los especialistas en seguridad a la hora de elegir en qué modelos centrarse, en función de su rendimiento y eficiencia, lo que ayuda a facilitar y hacer más eficaz la implementación en el mundo real.

El estudio también es valioso para la propia Fundación, ya que toman en cuenta los limitados recursos tecnológicos disponibles, por lo que no se trata solo de una comparación teórica, sino de encontrar soluciones que realmente puedan funcionar en

lugares con una infraestructura restringida haciendo que los resultados sean aún más relevantes y aplicables.

Formulación del problema de investigación

¿Cuáles son los algoritmos de aprendizaje automático más efectivos para la detección de malware, y cómo se comparan en términos de precisión, eficiencia computacional y robustez frente a ataques adversarios?

Objetivo General:

Evaluar y comparar la eficacia de diferentes algoritmos de aprendizaje automático para la detección de malware, enfocándose en la precisión, eficiencia computacional y robustez frente a ataques adversarios en distintos tipos de malware.

Objetivos Específicos:

- Comparar la precisión de diferentes algoritmos de aprendizaje automático en la detección de distintos tipos de malware (ransomware, spyware, troyanos).
- Analizar el impacto de diversas características de los datos (metadatos, comportamiento en entornos virtuales) en la eficacia de los algoritmos para la detección de malware.
- Evaluar la eficiencia computacional de cada algoritmo, considerando el tiempo de entrenamiento, el tiempo de inferencia y el uso de recursos.

Planteamiento hipotético

Los algoritmos de aprendizaje automático pueden ser más eficaces que los métodos tradicionales para la detección de malware, especialmente en la detección de malware nuevo o desconocido.

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

1.1. Revisión de literatura

1.1.1. Estudios sobre el uso de aprendizaje automático en ciberseguridad

Zainal & Ghaleb (2022), en su artículo "Malware Detection Issues, Challenges, and Future Directions: A Survey", presentan una revisión exhaustiva de los métodos basados en aprendizaje automático para la detección de malware. El objetivo principal fue identificar los retos actuales y futuros del uso de técnicas de machine learning en escenarios reales de ciberseguridad. La metodología se enfocó en analizar 105 estudios científicos, clasificándolos por tipo de algoritmo, tipo de ataque y métrica de evaluación. Se identificó que las técnicas basadas en árboles de decisión, redes neuronales y SVM se utilizan ampliamente, pero tienen limitaciones frente a los ataques adversarios y el malware polimórfico. En conclusión, los autores abogan por el desarrollo de enfoques híbridos que integren múltiples algoritmos y fuentes de datos.

García B. (2021), en su trabajo de fin de grado titulado "Aprendizaje Automático para la Detección de Tráfico IoT Anómalo, Spam y Malware", desarrollado en la Universidad de La Laguna, evaluó el desempeño de algoritmos supervisados (SVM, KNN y Random Forest) en la detección de tráfico malicioso. Utilizó datasets abiertos como CIC-IDS2017 y UNSW-NB15, aplicando técnicas de normalización, reducción de dimensionalidad y validación cruzada, en donde Random Forest muestra de mejor manera el equilibrio que hay entre precisión y eficiencia computacional en ámbitos heterogéneos, lo cual respalda su utilidad en escenarios con varios dispositivos IoT.

Estos estudios demuestran que el aprendizaje automático ha mejorado significativamente la detección de amenazas, pero aún enfrenta desafíos asociados a la variabilidad del malware y a la generación de falsos positivos, lo cual requiere un enfoque continuo de evaluación comparativa y adaptación dinámica.

1.1.2. Aplicaciones de Deep Learning para detección de malware

Diversos estudios han explorado el uso de arquitecturas de Deep Learning para la detección de software malicioso. Por ejemplo, Rojas & Rodríguez (2021) propusieron un modelo basado en redes neuronales recurrentes (LSTM) entrenado con secuencias de

comportamiento extraídas de máquinas virtuales, alcanzando una precisión del 94,2 % y reduciendo falsos positivos frente a modelos tradicionales. De igual manera, en el Barrio (2022) emplearon redes convolucionales (CNN) las cuales ayudaron a transformar archivos ejecutables en imágenes binarias para organizarlos visualmente, alcanzando una precisión del 96 %.

Esta investigación demuestra el potencial del aprendizaje profundo para escenarios complejos de ciberseguridad, sin embargo, su necesidad de recursos computacionales sustanciales y su falta de interpretabilidad dificultan su aplicación en entornos con infraestructura limitada, como el examinado en este estudio, por esta razón, se centró en algoritmos supervisados más ligeros y explicables, como Random Forest, SVM y XGBoost.

1.1.3. Evaluación comparativa de algoritmos de detección en entornos adversos

La literatura ha realizado evaluaciones comparativas entre diferentes algoritmos de machine learning para la detección de malware. Villavicencio (2023) analizó DNN, Naive Bayes, SVM y Random Forest utilizando un dataset de VirusShare, concluyendo que, aunque la DNN alcanzó la mayor precisión (97,4 %), fue más sensible a ruido adversarial, mientras que SVM mostró mayor estabilidad bajo condiciones perturbadas. Nieto (2021) realizó un análisis comparativo de varios algoritmos clásicos, aplicando sobre muestreo y validación estratificada a diversos tipos de malware, es decir, el estudio concluyó que ningún algoritmo es universalmente superior, ya que el rendimiento depende del tipo específico de amenaza y de la calidad del conjunto de datos.

Estos hallazgos justifican la selección de Random Forest, SVM y XGBoost para la presente investigación, ya que en conjunto ofrecen un equilibrio óptimo entre precisión, eficiencia y robustez frente a las variaciones de los datos, factores críticos en entornos operativos con recursos limitados.

A continuación, se presenta la Tabla 1, que resume los estudios revisados, señalando el tipo de algoritmo, contexto de uso y principales resultados.

Tabla 1. *Resumen de estudios comparativos sobre algoritmos de detección de malware*

Autor/a (Año)	Algoritmos Evaluados	Tipo de Estudio	Dataset(s)	Resultado Principal
(Zainal & Ghaleb, 2022)	Árboles, SVM, DNN	Revisión científica	Multifuente	Se propone enfoque híbrido para mayor robustez
(García, 2021)	SVM, KNN, RF	TFG experimental	CIC-IDS2017, UNSW-NB15	RF tuvo mejor precisión y eficiencia
(Rojas & Rodríguez, 2021)	LSTM	Tesis de maestría	Dataset propio	LSTM detecta amenazas APT con alta precisión
(Del Barrio, 2022)	CNN	TFG exploratorio	Visual malware dataset	CNN 96% de precisión con imágenes binarias
(Villavicencio, 2023)	DNN, NB, SVM, RF	Tesis de maestría	VirusShare	DNN preciso, pero menos robusto que SVM
(Nieto, 2021)	Múltiples	TFG comparativo	3 datasets variados	No hay algoritmo universalmente mejor

Nota: Elaboración propia

1.1.4. Casos de estudio relevantes en entornos latinoamericanos

Villavicencio (2023), en su tesis de maestría "Estudios de técnicas de aprendizaje automático utilizadas en ciberseguridad" desarrollada en la Universidad Politécnica Salesiana (Ecuador), implementó una prueba piloto de detección de malware en un entorno universitario local, empleando algoritmos DNN, SVM y Random Forest. El objetivo fue adaptar técnicas de aprendizaje automático a infraestructuras de bajo costo, como laboratorios académicos, y evaluar su viabilidad. Usó un dataset mixto: muestras recolectadas desde estaciones de trabajo institucionales, junto con archivos maliciosos de repositorios como VirusShare. Como conclusión, se identificó que los entornos limitados en hardware se benefician de modelos menos costosos como Random Forest, manteniendo una precisión aceptable (89%).

Herrera & Hernández (2023), en su artículo “Deployment of Ransomware Detection Using Dynamic Analysis and Machine Learning” de la Escuela Politécnica Nacional (Ecuador), desarrollaron un sistema de detección de ransomware basado en análisis dinámico con un conjunto de 50 características extraídas desde entornos sandbox, en donde se prepara a los modelos como Random Forest, Gradient Boosted Trees, Naive Bayes y redes neuronales, obteniendo resultados impresionantes, es decir, la precisión osciló entre el 74 % y el 100 % con validación cruzada, los tiempos de respuesta fueron rápidos, entre 0,15 s y 25 s, lo que es lo suficientemente rápido como para detener el ransomware antes de que cifre los archivos. Lo que hace que este estudio sea aún mejor es su reproducibilidad, ya que los datos se han publicado, y su claro enfoque en el uso real en sistemas institucionales.

Cruz (2021), en su tesis de máster en la Universidad de Alicante, exploró la relación entre las redes sociales y el malware en los países latinoamericanos, no creó sus propios modelos, pero recopiló datos valiosos sobre campañas de ingeniería social que propagan malware a través de WhatsApp y Facebook. Sus hallazgos ponen de relieve que en países como Ecuador, Colombia y México ha aumentado el uso del spear-phishing combinado con malware móvil, esto deja claro que los modelos también deben tener en cuenta el aspecto social de los ciberataques.

Estos casos muestran que, en América Latina, el uso del aprendizaje automático para detectar malware debe tener en cuenta tres aspectos: la disponibilidad de recursos, la reproducibilidad de los datos y el papel de los vectores de ataque sociales. En la Tabla 2 se resumen los estudios revisados en esta sección.

Tabla 2. *Casos de estudio en entornos latinoamericanos*

Autor/a (Año)	País / Universidad	Tipo de Malware	Algoritmos / Técnica	Conclusión principal
(Villavicencio, 2023)	Ecuador / UPS	General (Mixto)	DNN, SVM, RF	RF fue más viable en hardware limitado (~89 %)

(Herrera & Hernández, 2023)	Ecuador / EPN	Ransomware	RF, GBT, NB, NN; análisis dinámico sandbox	Exactitud 74–100 %, rápida detección antes del cifrado; datos publicados
(Cruz, 2021)	España (foco AL)	Malware móvil + Phishing	Análisis de campañas	Alta prevalencia de spear-phishing y malware móvil en AL (Ecuador, Colombia, México)

Nota: Elaboración propia

1.1.5. Revisión de datasets empleados en investigaciones recientes

La calidad y diversidad de los datasets empleados para entrenar modelos de detección de malware es un factor crítico en la eficacia del aprendizaje automático. En la última década, diversos estudios han utilizado conjuntos públicos y privados, con características estructuradas y no estructuradas.

García B (2021) trabajó con los conjuntos de datos CIC-IDS2017 y UNSW-NB15, dos de los más utilizados en la investigación sobre seguridad de redes. Ambos contienen tráfico legítimo y malicioso etiquetado, lo que permite aplicar modelos supervisados con técnicas de reducción de dimensionalidad, según el autor, CIC-IDS2017 es especialmente valioso porque incluye ataques modernos como botnets y ransomware.

Nieto (2021) exploró el uso de varios conjuntos de datos sintéticos y reales, incluidas muestras de VirusShare, una enorme colección de archivos maliciosos recopilados en campañas globale, además, señaló la importancia de un etiquetado coherente y destacó la necesidad de utilizar técnicas de equilibrio como SMOTE para evitar el sesgo hacia la clase mayoritaria.

Jácome y González (2022) dieron un paso más y crearon su propio conjunto de datos, es decir, supervisaron entornos virtuales y generaron trazas de comportamiento mientras ejecutaban malware, esto les permitió capturar características temporales clave, exactamente lo que se necesita para entrenar modelos LSTM.

La tabla 3 muestra una comparación de todos los conjuntos de datos utilizados en esta investigación, los mismos que se encuentran disponibles públicamente y se componen de

muestras del mundo real, lo que los hace fiables y prácticos para este tipo de trabajo. Se priorizó la coincidencia de campos relevantes como características de red (direcciones IP, puertos, protocolos) y atributos de tráfico (tamaño de paquetes, tiempo de conexión) para facilitar el entrenamiento supervisado y el análisis no supervisado. Además, se verificó su licencia abierta y la disponibilidad de documentación suficiente para garantizar la replicabilidad del estudio.

Tabla 3. *Comparación de datasets utilizados para detección de malware*

Dataset	Tipo de Datos	Acceso	Aplicaciones principales	Limitaciones
CIC-IDS2017	Tráfico de red real etiquetado	Público	Clasificación de ataques modernos y tráfico mixto	Volumen alto, preprocesamiento costoso
CSE-CIC-IDS2018	Tráfico de red real etiquetado	Público	Análisis de tráfico de botnets y ransomware	Campos extensos requieren filtrado
VirusShare	Binarios maliciosos reales	Público	Análisis estático/dinámico de muestras	Etiquetado parcial
EMBER Dataset	Características de binarios PE	Público	Clasificación de archivos ejecutables (Windows)	Exclusivo para binarios PE

Fuente: Información obtenida de (Recordon & Ruiz, 2021)

Estos datasets fueron seleccionados por su compatibilidad en campos clave (etiquetas, fecha/hora de captura, tipo de tráfico, familias de malware), su carácter público y su amplia utilización en la literatura reciente. La inclusión de EMBER permite enriquecer el análisis sobre binarios de Windows, mientras que los datasets CIC-IDS2017 y CSE-CIC-IDS2018 aportan tráfico real y documentado, evitando así sesgos generados por datos puramente sintéticos. VirusShare complementa el conjunto con binarios recolectados de fuentes reales, esenciales para la validación de técnicas híbridas de detección.

1.2. Desarrollo teórico y conceptual

1.2.1. Fundamentos del malware: definición, tipologías y evolución

El término malware proviene de la contracción de "malicious software" y se refiere a cualquier tipo de software diseñado con fines maliciosos como dañar sistemas, robar datos, interrumpir operaciones o vulnerar redes informáticas (Zainal & Ghaleb, 2022). En

la década de 1970 cuando se implementó un virus como Creeper y Elk Cloner, el malware ha desarrollado significativamente en términos de sofisticación, persistencia y capacidad de evasión.

Su evolución ha seguido una trayectoria hacia amenazas más inteligentes y especializadas, incluyendo ransomware dirigido, rootkits indetectables y amenazas persistentes avanzadas (APT). En la Tabla 4 se ofrece una clasificación general de los tipos de malware más comunes, junto con sus descripciones funcionales.

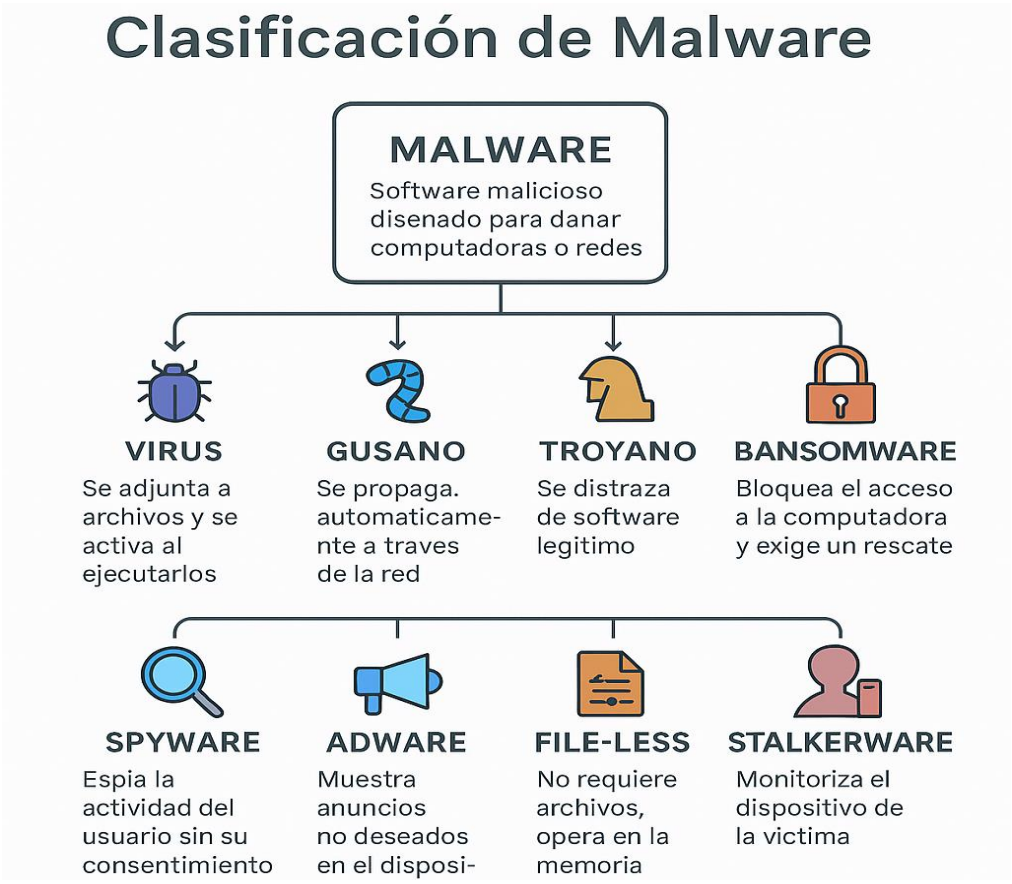
Tabla 4. *Tipologías de malware y su descripción funcional*

Tipo de Malware	Descripción operativa	Ejemplo común
Virus	Se replica insertándose en archivos ejecutables o sectores de arranque	ILOVEYOU
Gusano	Se propaga autónomamente por redes sin requerir acción del usuario	Conficker
Troyano	Se disfraza de software legítimo para generar puertas traseras	Zeus
Ransomware	Cifra archivos y exige pago para su recuperación	WannaCry
Spyware	Espía y recolecta datos del usuario sin su consentimiento	FinFisher
Adware	Muestra publicidad invasiva, enlazada con software gratuito	Fireball
Rootkit	Se oculta en el sistema y da acceso privilegiado al atacante	Necurs
Botnet	Red de equipos comprometidos usados para ataques coordinados	Mirai

Fuenta: Información obtenida de (Chancay, 2024)

Además, para una mejor comprensión visual de la variedad de amenazas que engloba el término malware, se presenta a continuación la Figura 1, la cual sintetiza gráficamente las principales categorías de malware, incluyendo sus modos de operación y objetivos. Esta representación facilita al lector la identificación de amenazas como virus, gusanos, spyware, ransomware, adware, stalkerware y variantes sin archivos (file-less), entre otras, destacando sus diferencias funcionales y vectores de ataque.

Figura 1. *Tipos de malware y sus características operativas*



Nota: Elaboración propia

1.2.2. Técnicas tradicionales de detección de malware

Las técnicas tradicionales de detección se basan principalmente en análisis estático y análisis dinámico, combinadas con el uso de firmas y heurísticas.

- Análisis estático: Examina el código del archivo sin ejecutarlo, identificando patrones binarios o firmas conocidas. Es eficiente, pero vulnerable a técnicas de ofuscación.
- Análisis dinámico: Ejecuta el archivo en un entorno controlado (sandbox) para observar su comportamiento. Aunque es más completo, consume más recursos.

Ambos métodos se complementan con:

- Detección por firmas: Comparación con bases de datos de patrones conocidos.

- Detección heurística: Evaluación de comportamientos sospechosos en tiempo real.

Aunque estas técnicas han sido ampliamente utilizadas, presentan debilidades frente a malware polimórfico o mutante, lo que motivó la incorporación de modelos inteligentes. En la Tabla 6 se comparan estas técnicas.

Tabla 5. *Comparación entre técnicas tradicionales de detección de malware*

Técnica	**Fortalezas**	**Limitaciones**
Análisis estático	Rápido, bajo costo computacional	Ineficaz ante ofuscación o cifrado
Análisis dinámico	Captura comportamiento real	Requiere sandboxing, alto costo
Detección por firmas	Alta precisión con malware conocido	Ineficaz ante variantes nuevas o modificadas
Detección heurística	Detecta comportamiento sospechoso	Genera falsos positivos si no está bien calibrada

Fuente: Información obtenida de (Marín, 2024)

1.2.3. Fundamentos del aprendizaje automático

El aprendizaje automático (Machine Learning, ML) es una rama de la inteligencia artificial que permite a los sistemas aprender patrones a partir de datos sin ser programados explícitamente (Obaido & Mienye, 2024). En vez de escribir reglas manuales, se educa un modelo con datos etiquetados para que aprenda a predecir nuevas entradas, en donde, busca patrones y relaciones mediante estadísticas y probabilidades.

El aprendizaje automático existe desde mediados del siglo XX, este comenzó con técnicas sencillas como la regresión y las redes neuronales básicas, además, especialmente desde la década de 1990, ha evolucionado hacia algoritmos más complejos y escalables. Hoy en día, la combinación de enormes cantidades de datos y potentes ordenadores ha impulsado el aprendizaje automático en casi todos los campos.

En ciberseguridad, el aprendizaje automático es increíblemente útil, ya que puede examinar grandes cantidades de datos (archivos binarios, tráfico de red, registros del sistema) y detectar comportamientos extraños que podrían indicar que se está

produciendo un ataque. Pero su impacto va mucho más allá de la ciberseguridad, es decir, se utiliza en la asistencia sanitaria para el diagnóstico asistido, en las finanzas para detectar fraudes, en la industria para el mantenimiento predictivo y en el comercio electrónico para recomendaciones personalizadas lo que demuestra lo versátil y transformador que es realmente el aprendizaje automático.

Los tipos de aprendizaje automático se suelen agrupar en tres categorías fundamentales:

- **Aprendizaje supervisado:** Entrena con datos previamente etiquetados (por ejemplo, malware/legítimo).
- **Aprendizaje no supervisado:** Busca patrones sin etiquetas previas, empleando técnicas de agrupamiento (clustering) o detección de valores atípicos (outliers).
- **Aprendizaje por refuerzo:** Aprende mediante la retroalimentación de recompensas y penalizaciones en función de las acciones que ejecuta en un entorno.

Una de las principales ventajas del ML es su capacidad para detectar amenazas nuevas o variantes desconocidas de malware, gracias al aprendizaje de características generales del comportamiento o la estructura del software, reduciendo la dependencia de firmas rígidas o reglas manuales.

1.2.4. Algoritmos aplicados a la detección de malware

Diversos algoritmos de aprendizaje automático se han aplicado con éxito en la detección de software malicioso, cada uno con características particulares en términos de precisión, eficiencia y resistencia frente a condiciones adversas (Güven, 2024). Este estudio se centrará en analizar Random Forest, XGBoost y SVM-RBF debido a su combinación equilibrada de interpretabilidad, capacidad predictiva y viabilidad de implementación. Por el contrario, se excluyeron las redes neuronales profundas (DNN) por su complejidad y sus elevadas exigencias computacionales, mientras que se descartó Naive Bayes debido a sus limitaciones al asumir la independencia de las variables.

En la siguiente sección se describen los mecanismos generales de los algoritmos seleccionados, incluidas sus principales fortalezas y debilidades. Posteriormente, la tabla

7 ofrece un resumen comparativo basado en los criterios de precisión, requisitos de datos, robustez frente a ataques adversarios e interpretabilidad.

Random Forest (RF)

RF es un algoritmo de ensemble learning que construye múltiples árboles de decisión a partir de subconjuntos aleatorios de datos y características. La decisión final se obtiene mediante votación mayoritaria entre los árboles (Salman, 2024). Se ha destacado por su buen rendimiento en problemas de clasificación binaria (malware vs. legítimo).

- **Ventajas:** Robusto al sobreajuste, rápido entrenamiento y fácil interpretación intermedia.
- **Desventajas:** Puede resultar menos preciso en problemas con alta no linealidad o ruido extremo.

Support Vector Machine (SVM)

Es un algoritmo basado en teoría estadística que busca encontrar el hiperplano óptimo que maximiza la separación entre clases (margen máximo). En ciberseguridad, se ha utilizado ampliamente para la detección de malware y tráfico anómalo (Pérez, 2024).

- **Ventajas:** Alta precisión en datasets de tamaño medio, robustez ante ruido y capacidad de generalización.
- **Desventajas:** Alto costo computacional en grandes volúmenes de datos y complejidad en la selección del kernel.

XGBoost

XGBoost es un algoritmo de boosting basado en árboles de decisión que optimiza el rendimiento mediante técnicas avanzadas de regularización y paralelización (IBM, 2024). Ha sido ampliamente adoptado en competiciones de machine learning por su elevada precisión y eficiencia computacional.

- **Ventajas:** Alta precisión, buen manejo de datos heterogéneos, eficiencia computacional optimizada.

- **Desventajas:** Configuración de hiperparámetros compleja, interpretabilidad menor que RF.

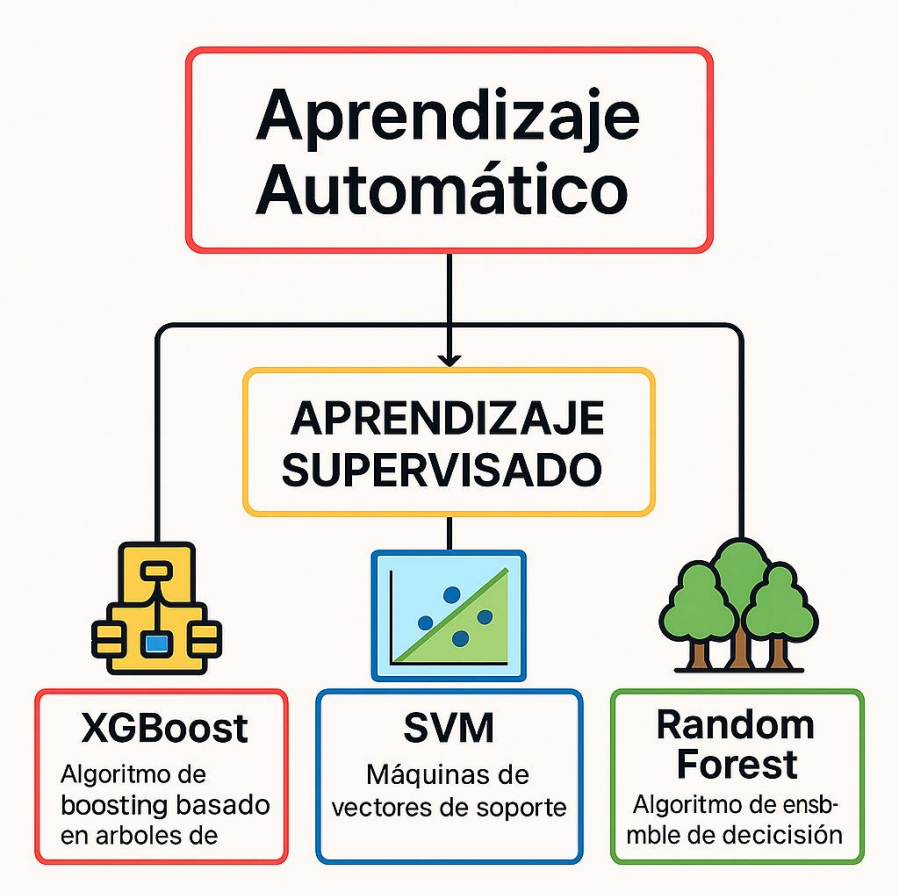
Tabla 6. *Comparación entre algoritmos de aprendizaje automático aplicados a malware*

Algoritmo	Precisión esperada	Requiere muchos datos	Resistente a ruido adversario	Interpretabilidad
Random Forest	Media-Alta	No	Alta	Media
SVM	Alta	Medio	Alta	Media-Baja
XGBoost	Alta	No	Alta	Media-Baja

Fuente: Información obtenida de (Villao & López, 2024)

Además, para facilitar la comprensión visual de los algoritmos de aprendizaje automático considerados en este proyecto, se presenta la Figura 2, la cual esquematiza los principales enfoques del aprendizaje supervisado, destacando explícitamente los algoritmos seleccionados para la investigación (Random Forest, SVM y XGBoost). Esta representación gráfica refuerza la clasificación teórica y permitirá al lector ubicar rápidamente el aporte de cada técnica supervisada al problema de detección de malware, diferenciándolas de otros enfoques no utilizados (aprendizaje no supervisado y por refuerzo).

Figura 2. *Tipos de aprendizaje automático y algoritmos aplicados a la investigación*



Nota: Elaboración propia

1.2.5. Métricas de evaluación de modelos de clasificación

El rendimiento de los modelos de aprendizaje automático en la detección de malware se evalúa mediante métricas cuantitativas que miden la eficacia de la clasificación binaria (malware frente a software benigno). Las métricas más utilizadas son la exactitud, la precisión, la recuperación, la puntuación F1 y el área bajo la curva ROC (AUC-ROC).

- Accuracy mide el porcentaje total de predicciones correctas.
- Precision evalúa cuántas de las instancias clasificadas como positivas realmente lo son.
- Recall mide la capacidad del modelo para detectar correctamente los positivos verdaderos.
- F1-score combina precisión y recall en una sola medida armónica.

- AUC-ROC evalúa la capacidad discriminativa del modelo en distintos umbrales.

En la Tabla 7, se describen estas métricas con su fórmula y su utilidad principal en el contexto de ciberseguridad.

Tabla 7. *Principales métricas de evaluación en detección de malware*

Métrica	Fórmula	Utilidad principal
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Medida global de aciertos
Precision	TP / (TP + FP)	Evita falsos positivos (ej. archivos seguros marcados como malware)
Recall	TP / (TP + FN)	Evita falsos negativos (malware que no se detecta)
F1-score	2 * (Precision * Recall) / (Precision + Recall)	Balance entre precisión y recall
AUC-ROC	Integral del gráfico ROC	Evalúa rendimiento en clasificación binaria

Fuente: Información obtenida de (Dunsin & Chahien, 2025)

Donde:

TP = Verdaderos positivos

TN = Verdaderos negativos

FP = Falsos positivos

FN = Falsos negativos

1.2.6. Robustez frente a ataques adversarios: definición y relevancia

En el contexto de la ciberseguridad, la robustez de un modelo se refiere a su capacidad de mantener un rendimiento aceptable ante perturbaciones o manipulaciones intencionales de los datos de entrada, conocidas como ataques adversarios. Estos ataques se presentan comúnmente en malware que ha sido modificado específicamente para evadir la detección de los modelos entrenados.

Estudios recientes como el de Güven (2024) evidencian que Random Forest y SVM muestran una mayor estabilidad frente a variaciones pequeñas en los datos, mientras que

XGBoost, pese a su alta precisión, puede ser más sensible a modificaciones sutiles si no se aplican técnicas de regularización adecuadas.

La Tabla 8 resume el comportamiento de los algoritmos seleccionados en esta investigación frente a ataques adversarios.

Tabla 8. *Comparación de robustez frente a ataques adversarios*

Algoritmo	Sensibilidad a perturbaciones	Comentario
Random Forest	Media	Menos sensible, pero puede ser engañado con datos sintéticos
SVM	Baja	Alto margen reduce impacto de modificaciones pequeñas
XGBoost	Meida-Alta	Alta precisión, pero puede ser vulnerable a ataques adversarios si no se implementan defensas adicionales.

Fuente: Información obtenida de (Salem & Azzam, 2024)

1.2.7. Preprocesamiento de datos y su impacto en el rendimiento del modelo

En cuanto al preprocesamiento de datos es un paso clave en la creación de modelos de aprendizaje automático, la cual garantiza que la información esté en el formato adecuado para su análisis y entrenamiento, en cambio, en ciberseguridad, los datos pueden provenir de muchos lugares diferentes, por ejemplo, archivos binarios, tráfico de red, secuencias de llamadas API o características dinámicas capturadas en entornos controlados (Mite, 2025).

Los investigadores coinciden en que un buen preprocesamiento marca una gran diferencia, ya que, ayuda a que los modelos sean más estables, más precisos y mejores a la hora de generalizar al clasificar el malware (Zhong & Salehi, 2023). Entre las técnicas más empleadas destacan la normalización o estandarización, la selección de características, la codificación de variables categóricas, el balanceo de clases mediante oversampling/undersampling, y la reducción de dimensionalidad a través de PCA o UMAP.

En la Tabla 9 se sintetizan estas técnicas, junto con sus objetivos y el impacto esperado en la construcción de modelos robustos de detección de amenazas.

Tabla 9. *Técnicas comunes de preprocesamiento de datos en detección de malware*

Técnica	Objetivo	Impacto esperado
Normalización/Estandarización	Escalar valores numéricos	Acelera la convergencia del modelo
Selección de características	Eliminar datos irrelevantes	Mejora precisión y reduce overfitting
Codificación (One-Hot, Label)	Convertir datos categóricos	Permite procesar logs o nombres de archivos
SMOTE / Oversampling	Balancear clases desbalanceadas	Mejora recall en detección de malware raro
PCA / UMAP	Reducir dimensionalidad	Aumenta interpretabilidad y eficiencia

Fuente: Información obtenida de (Carrizo & Álvarez, 2024)

1.2.8. Consideraciones éticas y legales en la experimentación con malware

La experimentación con software malicioso implica riesgos relevantes tanto técnicos como éticos y legales, que deben ser contemplados con rigor para evitar la generación de nuevos vectores de ataque o la propagación accidental de muestras activas. Por ello, cualquier trabajo académico o técnico que manipule malware debe enmarcarse en un protocolo de contención y control.

Aspectos éticos:

- Garantizar que las muestras no puedan ser reutilizadas para fines ilícitos.
- Realizar pruebas únicamente en entornos controlados (sandbox, redes virtualizadas aisladas, laboratorios sin acceso a Internet productivo).
- Documentar los procedimientos de análisis de forma transparente, evitando publicar código malicioso funcional que pueda ser reutilizado con propósitos criminales.
- Respetar el principio de no causar daño, incluyendo el compromiso de no difundir de forma directa archivos maliciosos hacia terceros.

Aspectos legales:

- En Ecuador, la Ley Orgánica de Protección de Datos Personales (Registro Oficial 2021) exige un manejo responsable de la información que se derive de sistemas o redes, protegiendo la privacidad de los usuarios si se involucran registros reales durante las pruebas.

- El Código Orgánico Integral Penal (COIP, Art. 232-234) tipifica como delito la propagación de software malicioso o la ejecución de ataques informáticos, por lo que las investigaciones deben garantizar que el malware no se distribuya fuera del entorno experimental.

- Los datasets utilizados deben contar con licencias abiertas (por ejemplo, CIC-IDS2017, EMBER) que autoricen su uso con fines académicos o de investigación, evitando vulnerar derechos de autor o restricciones de confidencialidad.

A nivel internacional, directrices como el Budapest Convention on Cybercrime (2001) nos recuerdan lo importante que es que las escuelas y los investigadores mantengan medidas sólidas de bioseguridad digital, lo que ayuda a prevenir fugas de muestras de malware que podrían ser utilizadas indebidamente.

En resumen, cualquier investigación centrada en el malware debe seguir protocolos de análisis estrictos, debe contar con respaldo legal, supervisión ética y un compromiso claro con la confidencialidad protegiendo así tanto los sistemas que se estudian como los datos involucrados.

CAPÍTULO 2. METODOLOGÍA

2.1. Contexto de la investigación

La presente investigación se desarrolla en la Fundación Salinas Yacht Club, ubicada en la ciudad de Salinas, provincia de Santa Elena, Ecuador, entre las calles Jaime Roldós Aguilera y Leonardo Avilés Tabaréz, la cual brinda formación en robótica e informática básica a niños, jóvenes y adultos. En este entorno educativo no existe actualmente un mecanismo formal de seguridad informática que proteja a los equipos utilizados en los cursos de posibles infecciones de malware o ciberataques.

La tecnología disponible es bastante básica, ya que consiste en estaciones de trabajo con procesadores Intel Core i7 de octava y undécima generación, 16 GB de RAM y Windows 10/11, que están conectadas a través de una red local sencilla con un router y puntos de acceso, sin segmentación del tráfico.

La principal motivación de este estudio proviene de la preocupación de la Fundación por mantener la seguridad de estos ordenadores, que se encuentran constantemente expuestos, es decir, los estudiantes conectan unidades USB externas todo el tiempo y se conectan a redes abiertas para realizar actividades de aprendizaje, lo que hace que la ciberseguridad sea una verdadera prioridad.

Por ello, se plantea comparar el rendimiento de diversos algoritmos de aprendizaje automático orientados a la detección de malware, con el propósito de definir un esquema de protección que pueda ser implementado posteriormente, aun sin contar con servidores de alto rendimiento. En la Tabla 10 se detalla la infraestructura tecnológica con la que se cuenta para ejecutar los experimentos de validación de modelos, información que resulta relevante para comprender las limitaciones de cómputo consideradas en el diseño de esta propuesta.

Tabla 10. *Infraestructura disponible en la Fundación*

Recurso	Descripción técnica
Procesadores	Intel Core i7 (8.ª y 11.ª generación)
Memoria RAM	16 GB

Almacenamiento	SSD 512 GB por equipo
Sistema operativo	Windows 10 / Windows 11
Red	Router + puntos de acceso (AP) sin segmentación
Servidores dedicados	No disponible
Entorno de ejecución	Jupyter Notebook + bibliotecas Python (scikit-learn, xgboost, pandas)

Nota: Elaboración propia

2.2. Diseño y alcance de la investigación

La presente investigación adopta un diseño experimental, ya que se basa en la implementación, entrenamiento y evaluación comparativa de modelos de aprendizaje automático bajo condiciones controladas. Se manipulan variables como el tipo de algoritmo, los parámetros de entrenamiento, el tamaño de la muestra y las técnicas de preprocesamiento, con el fin de observar su impacto directo en el rendimiento de los modelos frente a un conjunto común de datos. Esta base experimental autoriza medir con gran precisión la solución de cada modelo frente a los mismos contextos y establecer comparaciones objetivas entre ellos.

El alcance de este estudio es descriptivo y correlacional, en cuanto a lo descriptivo es porque analiza cómo se comporta cada algoritmo, utilizando métricas clásicas como la precisión, la recuperación, la puntuación F1 y el ROC-AUC, también es correlacional porque explora cómo estas métricas de rendimiento se relacionan con factores como la eficiencia computacional (tiempo de entrenamiento, uso de memoria) y la capacidad de generalización del modelo. Este tipo de análisis ayuda a revelar patrones y conexiones entre el tipo de modelo y su viabilidad técnica cuando se trabaja con una infraestructura limitada.

Es importante señalar que la Fundación no cuenta con servidores especializados ni con una infraestructura de red avanzada, es decir, todos los experimentos se realizan en ordenadores de sobremesa normales con recursos moderados, por ello, el estudio trabaja con subconjuntos representativos de los datos, seleccionados de forma estratificada y

equilibrada, garantizando así que el estudio sea técnicamente viable y facilita la aplicación posterior de los resultados en entornos similares al de la organización.

2.3. Tipo y métodos de investigación

El enfoque adoptado en esta investigación fue de carácter cuantitativo, dado que se centró en la recolección, procesamiento y análisis de datos numéricos derivados de la ejecución de algoritmos de aprendizaje automático aplicados a la detección de malware. Este enfoque permitió medir de forma objetiva y replicable métricas como precisión, recall, F1-score y área bajo la curva ROC (AUC-ROC), así como variables asociadas a la eficiencia computacional, tales como el tiempo de entrenamiento, tiempo de inferencia y consumo de memoria.

En cuanto al método, se empleó el hipotético-deductivo, que partió de la formulación de hipótesis basadas en antecedentes teóricos y estudios previos sobre clasificación de malware y eficiencia algorítmica. Estas hipótesis fueron contrastadas mediante experimentación controlada, evaluando su validez empírica a partir de los resultados obtenidos en los modelos implementados.

Las hipótesis planteadas para este estudio fueron las siguientes:

- **H1:** Se espera que el algoritmo XGBoost alcance una precisión media superior a la de Random Forest y SVM a la hora de clasificar muestras de malware.
- **H2:** SVM demuestra una mayor solidez frente a las fluctuaciones en los datos de entrada en comparación con Random Forest y XGBoost.
- **H3:** En el entorno de ejecución dado, los algoritmos basados en árboles de decisión, como Random Forest y XGBoost, tienden a requerir más recursos computacionales, tanto en tiempo de entrenamiento como de inferencia, en comparación con SVM.

El método hipotético-deductivo se justificó plenamente, ya que:

- Se formularon supuestos concretos con base en teorías y literatura especializada en detección de malware.

- Se diseñaron y ejecutaron experimentos controlados para verificar dichas hipótesis.
- Se realizó un análisis cuantitativo y estadístico de los resultados, alineado con los objetivos específicos de la investigación.

Este enfoque metodológico permitió evaluar de manera objetiva el comportamiento de los algoritmos seleccionados bajo las condiciones reales de infraestructura disponibles en la Fundación, contribuyendo a determinar su aplicabilidad práctica en contextos educativos con limitaciones técnicas.

2.4. Población y muestra

En el contexto de esta investigación, la población de estudio está constituida por la totalidad de registros disponibles en el conjunto de datos Malware Memory Analysis / CIC-MalMem-2022, el cual contiene información extraída de la memoria de sistemas afectados por diversos tipos de software malicioso. Este dataset fue publicado por el Canadian Institute for Cybersecurity (CIC) y se encuentra disponible públicamente a través de la plataforma Kaggle. Su relevancia radica en que incluye un amplio espectro de muestras clasificadas en las categorías: Benign, Ransomware, Spyware y Trojan, lo que permite realizar una comparación robusta entre clases.

Dado que el dataset cuenta con más de 50.000 registros, se optó por aplicar un muestreo no probabilístico por conveniencia sobre un subconjunto representativo de los datos, en función de las limitaciones de cómputo de la infraestructura disponible. Este subconjunto fue seleccionado garantizando la preservación proporcional de las clases (estratificación), a fin de mantener el equilibrio necesario para el entrenamiento y validación de los modelos de clasificación.

La muestra definitiva se dividió en tres grupos para su procesamiento experimental:

- 70 % para entrenamiento del modelo (training set)
- 15 % para validación durante el ajuste de hiperparámetros (validation set)
- 15 % para evaluación final del rendimiento (test set)

Adicionalmente, se aplicó la técnica de SMOTE (Synthetic Minority Oversampling Technique) exclusivamente sobre el conjunto de entrenamiento, con el objetivo de balancear las clases minoritarias sin introducir sesgo en los procesos de validación ni evaluación.

La Tabla 11 muestra la distribución original de clases en el dataset completo antes del muestreo y balanceo:

Tabla 11. *Distribución original de clases en el dataset CIC-MalMem-2022*

Tipo de muestra	Cantidad de registros	Porcentaje (%)
Benign	16.738	33.47%
Ransomware	12.05	23.95%
Spyware	9.247	18.49%
Trojan	11.965	24.09%
Total	**49.999**	**100%**

Nota: Elaboración propia

Esta estrategia de muestreo asegura que los modelos sean entrenados y evaluados sobre un conjunto diverso, realista y técnicamente manejable, permitiendo obtener resultados válidos sin requerir infraestructuras computacionales de alto rendimiento.

2.5. Técnicas e instrumentos de recolección de datos

La recolección de datos en esta investigación se realizó mediante técnicas digitales automatizadas aplicadas a conjuntos de datos públicos especializados en ciberseguridad y detección de malware. La fuente principal de los datos fue la plataforma Kaggle, complementada con repositorios académicos como ResearchGate y sitios institucionales como el del Canadian Institute for Cybersecurity (CIC).

a) Selección del dataset

Durante la fase exploratoria, se analizaron múltiples fuentes de datos para identificar el conjunto más adecuado a los objetivos de esta investigación. En la Tabla 12 se resumen los principales datasets evaluados:

Tabla 12. *Datasets analizados para la selección del conjunto de estudio*

Nombre del dataset	Fuente	Registros aprox.	Observaciones
CSE-CIC-IDS2018	UNB CIC	> 1.000.000	Contiene solo dos clases (Benign vs Ataque), sin tipificación de malware.
Top 1000 PE Imports Malware	Kaggle (ang3loliveira)	= 2 GB	Muestras hashadas, sin nombres de familia ni etiquetas comprensibles.
UGRansomware	ResearchGate	= 140.000	Incluye únicamente variantes de ransomware, sin otras clases de malware.
Malware Memory Analysis / CIC-MalMem-2022	Kaggle (luccagodoy)	= 50.000	Incluye clases etiquetadas: Benign, Ransomware, Spyware y Trojan.

Nota: Elaboración propia basada en información de repositorios públicos (2025).

Luego del análisis comparativo, se seleccionó el dataset Malware Memory Analysis / CIC-MalMem-2022, ya que cumple con los siguientes criterios:

- Incluye múltiples tipos de malware clasificados por familias (ransomware, spyware, troyanos).
- Presenta un balance relativo entre clases, facilitando el análisis supervisado.
- Proviene de análisis forense de memoria RAM, utilizando herramientas como Volatility o Rekall, lo cual otorga trazabilidad y calidad técnica a los registros.
- Tiene un tamaño manejable para entornos con infraestructura limitada.

b) Técnicas aplicadas

El proceso de recolección, preprocesamiento y análisis se llevó a cabo mediante técnicas automatizadas programadas en Python, con los siguientes pasos:

- Carga y validación del dataset desde archivo .csv limpio.
- Eliminación de columnas innecesarias o con varianza nula.
- Análisis exploratorio de datos (EDA): distribuciones, correlaciones, conteo de clases.
- Codificación de etiquetas con LabelEncoder.
- División estratificada del dataset en entrenamiento, validación y prueba (70%-15%-15%).
- Aplicación de SMOTE para balancear el conjunto de entrenamiento.
- Estandarización de variables con StandardScaler.

c) Instrumentos utilizados

La recolección, procesamiento y visualización de datos se realizó en entorno digital local, con los siguientes instrumentos:

Tabla 13. *Detalle instrumentos utilizados*

Tipo de instrumento	Herramienta / Librería
Lenguaje de programación	Python 3.11
Entorno de desarrollo	Jupyter Notebook (Anaconda)
Librerías de ML	scikit-learn, xgboost, imblearn, lightgbm
Librerías de visualización	matplotlib, seaborn, plotly
Métricas computacionales	psutil, time
Manejo de datos	pandas, numpy

Nota: Elaboración propia

Estos instrumentos permitieron desarrollar un pipeline completo de análisis, desde la carga de datos hasta la interpretación de resultados, garantizando reproducibilidad, transparencia y precisión en el tratamiento de la información.

2.6. Procesamiento de la evaluación: Validez y confiabilidad de los instrumentos aplicados para el levantamiento de información.

Para proteger la calidad metodológica del estudio, se implementaron procedimientos rigurosos que aseguran tanto la validez externa de los parámetros de evaluación como la fiabilidad estadística de los resultados, es decir, el uso combinado de validación cruzada, partición estratificada y flujo de datos controlado entre fases garantiza un análisis reproducible e imparcial.

a) Validez del proceso de evaluación

La validez del estudio se fundamenta en tres pilares:

Validación cruzada estratificada (5-fold): Se utilizó el método StratifiedKFold para realizar validación cruzada del conjunto de entrenamiento balanceado. Este enfoque preserva la proporción de clases en cada fold, lo cual es crucial en problemas de clasificación multiclase con desbalance inicial, como es el caso del dataset CIC-MalMem-2022.

Métricas objetivas y estandarizadas: La evaluación de los modelos se realizó con métricas de aceptación internacional que permiten comparar el desempeño de los algoritmos bajo múltiples perspectivas. Estas métricas incluyen:

- **Accuracy:** Porcentaje global de predicciones correctas.
- **Precision:** Porcentaje de verdaderos positivos sobre todos los positivos predichos. Mide la exactitud del modelo.
- **Recall:** Porcentaje de verdaderos positivos sobre todos los positivos reales. Mide la capacidad de detección.
- **F1-score:** Una medida equilibrada entre precisión y recall especialmente eficaz en escenarios con desequilibrio de clases.
- **ROC-AUC:** El área bajo la curva ROC (Receiver Operating Characteristic) mide la capacidad de un modelo para distinguir entre diferentes clases.

Estas métricas se calcularon utilizando tanto una media macro (media global) como una media por clase (Benign, Ransomware, Spyware y Trojan), lo que ofrece una visión más detallada del rendimiento del modelo dentro de cada categoría individual.

b) Confiabilidad de los resultados

Para garantizar que los resultados sean confiables y reproducibles, se establecieron los siguientes mecanismos de control:

- **Partición reproducible de los datos:** Se estableció un valor fijo de random_state=42 en cada etapa de división (entrenamiento, validación, prueba), para garantizar que las ejecuciones sean consistentes entre pruebas y replicables por otros investigadores.

- **Aplicación controlada de SMOTE:** La metodología de sobre muestreo SMOTE se aplicó exclusivamente al conjunto de entrenamiento para evitar la contaminación de los conjuntos de datos de validación y prueba. Este enfoque garantiza que las métricas de evaluación reflejen con precisión el rendimiento real del modelo en datos no vistos.

- **Separación estricta del conjunto de prueba:** El conjunto de pruebas, que comprende el 15 % del total de datos, se mantuvo completamente separado del proceso de entrenamiento y validación, y se utilizó únicamente para la evaluación final. Esta estricta separación evita la fuga de datos y el sobreajuste, lo que garantiza resultados imparciales y fiables.

De manera conjunta, estas medidas garantizan que la experimentación se ajuste a prácticas sólidas de validación de modelos, lo que permite obtener resultados fiables y estadísticamente significativos que se adaptan perfectamente al contexto operativo de la Fundación.

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

3.1. Introducción al capítulo

El presente capítulo expone los resultados obtenidos a partir de la aplicación de los procedimientos metodológicos detallados en el Capítulo 2. La lógica de la exposición se organiza de manera secuencial, siguiendo el orden de los objetivos planteados en la investigación, lo que permite mantener coherencia entre la planificación metodológica y los hallazgos alcanzados.

En primer lugar, se presentan los resultados relacionados con la evaluación del desempeño de los algoritmos seleccionados, considerando métricas de clasificación como precisión, recall, F1-score y área bajo la curva ROC (AUC-ROC). Posteriormente, se detallan los resultados vinculados con la eficiencia computacional, que incluyen el tiempo de entrenamiento, el tiempo de inferencia y el consumo de memoria durante la ejecución de los modelos.

Adicionalmente, se analizan los experimentos relacionados con la capacidad de los algoritmos no supervisados para detectar patrones o agrupaciones en las muestras de malware, lo que aporta evidencia empírica respecto a las hipótesis planteadas en el marco del método hipotético-deductivo.

De esta forma, el capítulo no solo muestra los resultados obtenidos, sino que establece un vínculo directo con los objetivos específicos y las hipótesis del estudio, constituyendo la base para la posterior discusión e interpretación crítica de los hallazgos.

3.2. Preparación y análisis preliminar de datos

La preparación de los datos constituyó la primera etapa práctica de la investigación, con el objetivo de asegurar que el conjunto de información se encontrara en condiciones óptimas para el entrenamiento de los modelos supervisados. A continuación, se detallan los resultados obtenidos en cada uno de los pasos (0–4) descritos en la metodología.

3.2.1. Limpieza y depuración del dataset

En el Paso 0, se cargó el dataset crudo con un total de 58.596 registros y 57 variables. Posteriormente, se construyó la variable de salida malware_type con cuatro categorías principales. La Tabla 14 muestra la distribución de las clases iniciales.

Tabla 14. *Distribución de clases en la variable malware_type*

Clase	Frecuencia	Porcentaje
BENIGN	29.298	50,0 %
SPYWARE	10.02	17,1 %
RANSOMWARE	9.791	16,7 %
TROJAN	9.487	16,2 %
Total	**58.596**	**100%**

Nota: Elaboración propia

Se eliminaron las columnas redundantes Category y Class, obteniéndose un archivo depurado denominado dataset_limpio.csv.

3.2.2. Auditoría del dataset limpio

En el Paso 1, se revisó la distribución de clases, confirmando la prevalencia de la clase BENIGN (50 % del total). Asimismo, se identificaron:

- 3 columnas constantes.
- 8 columnas casi constantes.
- 6 variables con predominio de valores cero, entre las que se encontraron:
    - pslist.nprocs64bit
    - handles.nport
    - svcscan.interactive_process_services

3.2.3. Eliminación de columnas no informativas

En el Paso 2, se aplicó un filtro de varianza cero (VarianceThreshold). Sin embargo, no se eliminaron variables adicionales, por lo que el dataset mantuvo 48 características útiles.

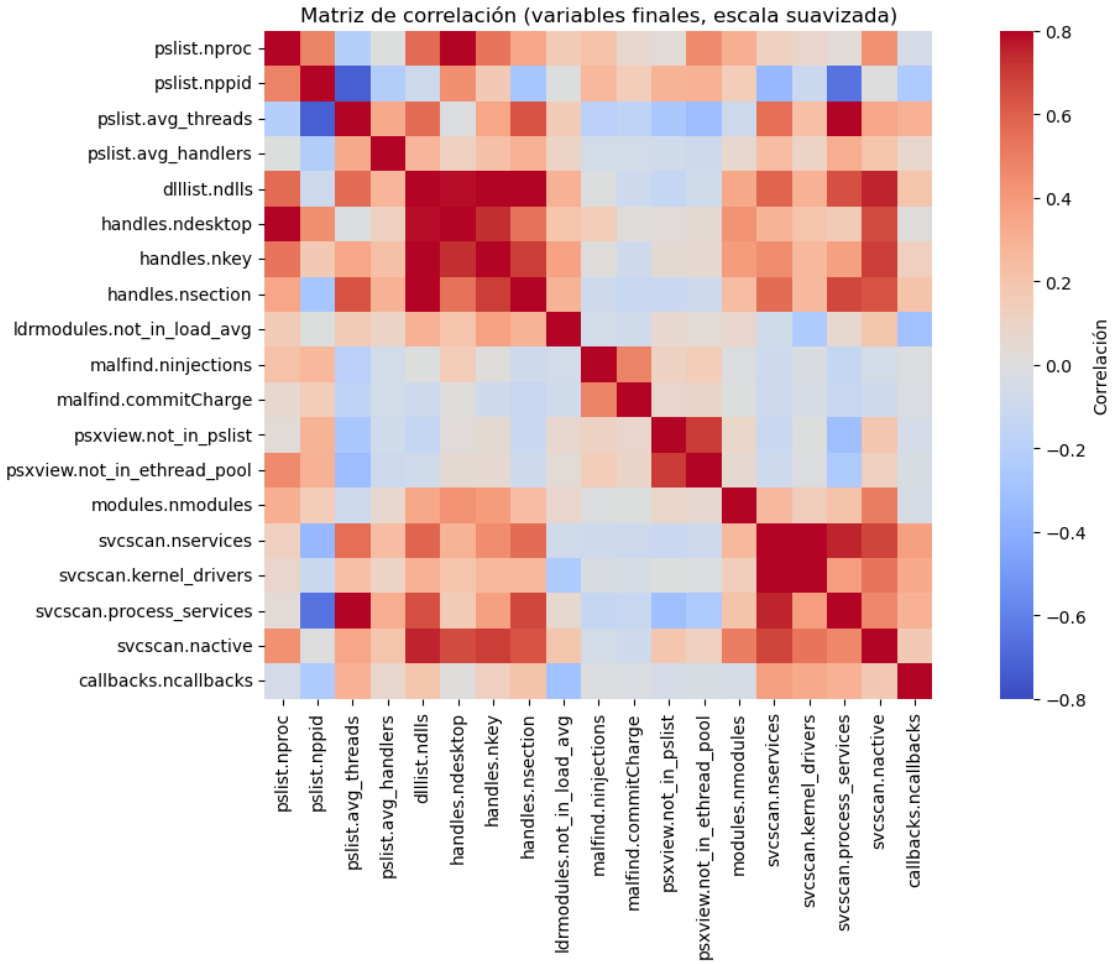
3.2.4. Reducción de multicolinealidad

En el Paso 3, se calculó la matriz de correlaciones absolutas entre variables. Se eliminaron aquellas con correlación mayor o igual a 0,95, lo que implicó la exclusión de 21 atributos altamente redundantes.

De esta forma, el dataset quedó reducido a 27 variables finales, mejorando la estabilidad de los modelos y reduciendo el riesgo de sobreajuste.

La Figura 3 muestra la matriz de correlación de las variables finales conservadas, donde se aprecia la reducción de la multicolinealidad y la presencia de agrupamientos de variables moderadamente correlacionadas.

Figura 3. *Matriz de correlación de las variables finales tras reducción de multicolinealidad*



Nota: Elaboración propia

3.2.5. Codificación de etiquetas y dataset final

Finalmente, en el Paso 4, se codificó la variable de salida malware_type mediante LabelEncoder, asignando las etiquetas mostradas a continuación:

- BEGIN = 0
- RANSOMWARE = 1
- SPYWARE = 2
- TROJAN = 3

El dataset final (dataset_final_modelado.csv) quedó conformado por 58.596 registros y 28 columnas, de las cuales:

- 27 correspondieron a características numéricas (float64),
- 1 columna a la etiqueta codificada (int).

3.3. División de conjuntos y balanceo de clases

Con el fin de evaluar el desempeño de los modelos de forma objetiva y evitar sesgos derivados de una partición no representativa, el dataset final se dividió en tres subconjuntos de datos mediante muestreo estratificado. Posteriormente, se aplicó la técnica de sobremuestreo SMOTE (Synthetic Minority Over-sampling Technique) exclusivamente al conjunto de entrenamiento, con el propósito de corregir el desbalance de clases.

3.3.1. División en subconjuntos de datos

En el Paso 5, los datos se separaron en las proporciones de 70 % para entrenamiento, 15 % para validación y 15 % para prueba, respetando la distribución original de clases mediante estratificación. La Tabla 15 muestra los tamaños finales de cada subconjunto.

Tabla 15. *División de conjuntos de datos*

Conjunto	Registros	Porcentaje
Entrenamiento	41.017	70%
Validación	8.789	15%
Prueba	8.79	15%
Total	**58.596**	**100%**

Nota: Elaboración propia

La distribución de clases se mantuvo prácticamente idéntica en los tres subconjuntos, tal como se muestra a continuación:

- **Entrenamiento:** BENIGN (49,9 %), SPYWARE (17,1 %), RANSOMWARE (16,7 %), TROJAN (16,2 %).
- **Validación:** BENIGN (50,0 %), SPYWARE (17,1 %), RANSOMWARE (16,7 %), TROJAN (16,2 %).
- **Prueba:** BENIGN (50,0 %), SPYWARE (17,1 %), RANSOMWARE (16,7 %), TROJAN (16,2 %).

Esto garantizó que todos los conjuntos preservaran la proporción real de clases, evitando sesgos en la evaluación posterior.

Justificación de la proporción empleada

La elección de la proporción 70/15/15 respondió a la necesidad de equilibrar el tamaño de los subconjuntos para maximizar la utilidad del dataset.

- **Mayor cantidad de datos para entrenamiento:** el 70 % permite que los algoritmos aprendan patrones representativos de todas las clases, en especial de las minoritarias.
- **Validación robusta:** reservar un 15 % para validación permite optimizar hiperparámetros con mayor estabilidad estadística, reduciendo el riesgo de sobreajuste.

- **Prueba confiable:** otro 15 % para prueba garantiza una evaluación final más precisa y representativa de la capacidad de generalización, evitando que métricas como F1-score y ROC-AUC se vean afectadas por el azar de subconjuntos demasiado pequeños (lo que podría ocurrir con el típico 80/20).

Esta estrategia es especialmente relevante en problemas de clasificación multiclase y desbalanceada, ya que asegura que cada clase esté suficientemente representada en las tres etapas (entrenamiento, validación y prueba).

3.3.2. Aplicación de SMOTE para balanceo de clases

En el Paso 6, se aplicó SMOTE sobre el conjunto de entrenamiento con el fin de equilibrar las clases minoritarias respecto a la mayoritaria (BENIGN).

Tabla 16. *Distribución de clases antes y después de SMOTE (conjunto de entrenamiento)*

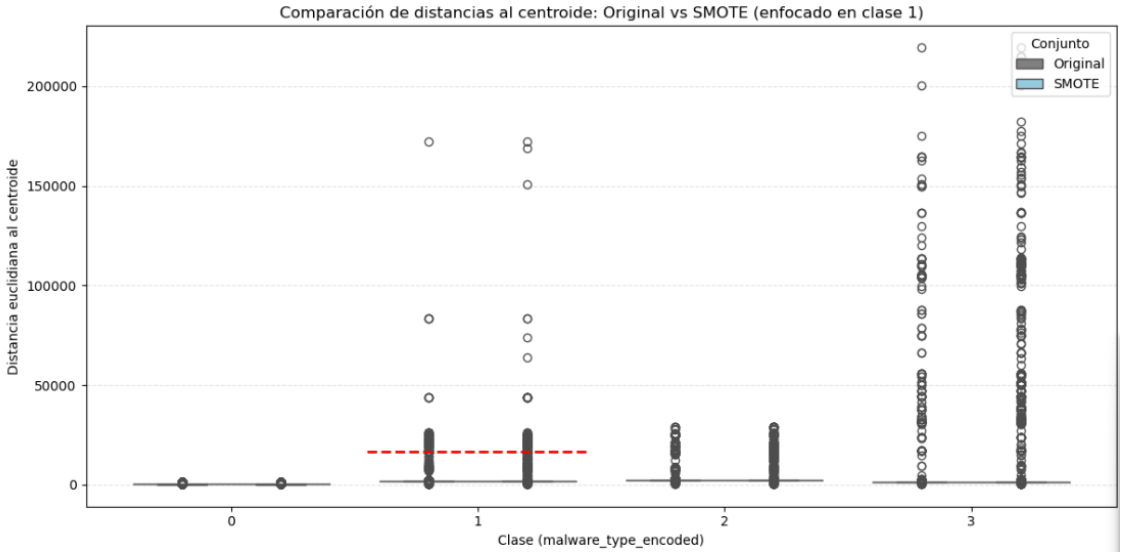
Clase	Antes de SMOTE	Después de SMOTE
BENIGN (0)	20.508	20.508
RANSOMWARE (1)	6.854	20.508
SPYWARE (2)	7.014	20.508
TROJAN (3)	6.641	20.508
Total	**41.017**	**82.032**

Nota: Elaboración propia

Como resultado, cada clase alcanzó 20.508 muestras, generándose un conjunto balanceado de 82.032 instancias para entrenamiento.

La Figura 4 presenta la comparación de distancias al centroide entre el dataset original y el generado con SMOTE, lo que permitió confirmar que las muestras sintéticas se mantuvieron razonablemente próximas a las originales, reduciendo el riesgo de introducir ruido excesivo.

Figura 4. *Comparación de distancias al centroide: dataset original vs. SMOTE (conjunto de entrenamiento)*



Nota: Elaboración propia

3.3.3. Discusión

El uso de SMOTE resultó fundamental para evitar el sesgo de los modelos hacia la clase mayoritaria (BENIGN), garantizando un aprendizaje más equilibrado. Esta técnica contribuyó a mejorar la capacidad de detección de clases minoritarias (RANSOMWARE, SPYWARE y TROJAN), especialmente en métricas como recall y F1-score.

No obstante, también implica ciertos riesgos:

- La generación de instancias sintéticas puede introducir ruido artificial si las clases presentan gran dispersión.
- El incremento del tamaño del dataset eleva el costo computacional en entrenamiento.

Pese a estas limitaciones, los resultados gráficos confirmaron que las muestras añadidas mantuvieron coherencia con la estructura de los datos originales, lo que justificó plenamente la aplicación de la técnica en esta investigación.

3.4. Escalamiento de variables

En esta etapa se aplicó un proceso de escalamiento de variables con el objetivo de homogenizar las magnitudes de los predictores y evitar que aquellos con valores absolutos mayores dominen el entrenamiento de los algoritmos supervisados. Para ello se utilizó el método StandardScaler, que transforma cada variable de manera que presente una distribución con media cero y desviación estándar unitaria.

El uso de este procedimiento fue necesario debido a que muchos algoritmos de aprendizaje supervisado, como las Máquinas de Vectores de Soporte (SVM) y los modelos basados en gradiente, son sensibles a las escalas de las variables. Sin este ajuste, métricas como la distancia euclidiana podrían generar un sesgo hacia las variables con rangos más amplios, afectando la convergencia y el rendimiento del modelo.

Los resultados del escalamiento muestran que tanto los conjuntos de entrenamiento, validación y prueba mantuvieron su tamaño original pero con las características normalizadas:

- X_train_sc: (82,032, 26)
- X_valid_sc: (8,789, 26)
- X_test_sc: (8,790, 26)

Para evidenciar el efecto del escalamiento, la Figura X ilustra un ejemplo de distribución de una variable antes y después de aplicar StandardScaler. Se observa cómo, tras la normalización, los valores se centran alrededor de cero y adquieren una desviación estándar de uno, lo que permite un aprendizaje más eficiente.

3.5. Entrenamiento de modelos supervisados

Una vez preprocesados los datos, se procedió al entrenamiento de modelos supervisados con tres algoritmos de referencia en problemas de clasificación:

- **Random Forest (RF):** modelo basado en múltiples árboles de decisión entrenados de manera aleatoria, cuyo resultado se obtiene a través de un proceso de votación mayoritaria. Destaca por su capacidad de manejar gran cantidad de predictores y reducir el riesgo de sobreajuste mediante el ensamblado.

- **Support Vector Machine con kernel Radial Basis Function (SVM-RBF):** clasificador que proyecta los datos a un espacio de mayor dimensionalidad para encontrar un hiperplano óptimo de separación. El kernel RBF fue seleccionado debido a su capacidad para capturar relaciones no lineales.

- **Extreme Gradient Boosting (XGBoost):** algoritmo de ensamblado basado en boosting secuencial que construye árboles de decisión de manera iterativa, optimizando la función de pérdida mediante técnicas de regularización y control de complejidad.

Para cada modelo se realizó un proceso de búsqueda de hiperparámetros (GridSearchCV) con validación cruzada de 5 particiones (5-fold CV), evaluando los resultados en función de la métrica F1-macro, que considera el equilibrio entre precisión y exhaustividad para cada clase.

Los principales resultados se resumen en la Tabla 17:

Tabla 17. *Hiperparámetros óptimos y desempeño en validación cruzada*

Algoritmo	Hiperparámetros óptimos	F1-macro (CV)
Random Forest	n_estimators=500, max_depth=35, min_samples_leaf=1, max_features='sqrt'	0.9286
SVM-RBF	C=3, gamma=0.1	0.7009
XGBoost	n_estimators=700, max_depth=6, learning_rate=0.1, subsample=0.8, colsample_bytree=1.0	0.9014

Nota: Elaboración propia

Se evidencia que el Random Forest obtuvo el mejor rendimiento en términos de F1-macro, seguido de XGBoost, mientras que el desempeño del SVM-RBF fue más limitado. Estos resultados permiten anticipar que los algoritmos de ensamblado ofrecen mayor capacidad predictiva para el problema abordado.

3.6. Evaluación de desempeño en valid/test

Con el fin de determinar la eficacia de los modelos de clasificación desarrollados, se presentan a continuación los resultados de su desempeño tanto en el conjunto de validación (valid) como en el conjunto de prueba (test). Para cada algoritmo se incluyen métricas globales de precisión (precision), exhaustividad (recall), medida F1 (f1-score) y exactitud global (accuracy), además de los valores promedios ponderados y macro promedios que permiten evaluar el rendimiento en todas las clases de manera equilibrada.

De forma complementaria, se analizan las curvas ROC-AUC bajo el esquema One-vs-Rest (OvR), lo cual ofrece una visión más robusta de la capacidad discriminante de cada modelo frente a clases múltiples. Asimismo, se presentan las matrices de confusión correspondientes, ya que constituyen un recurso clave para identificar patrones de aciertos y errores en la clasificación.

Este conjunto de métricas y visualizaciones permite no solo medir la capacidad predictiva de los modelos, sino también establecer comparaciones entre ellos, identificando fortalezas y limitaciones en distintos escenarios de evaluación. A continuación, se detallan los resultados obtenidos para cada modelo de clasificación.

3.6.1. Random Forest

Métricas globales por clase

En la Tabla 18 - 19 se presentan los resultados obtenidos por el algoritmo Random Forest tanto en el conjunto de validación como en el de prueba. Se incluyen las métricas de precisión, recall, F1-score y accuracy, calculadas para cada clase de malware y de manera global.

Tabla 18. *Resultados de evaluación de Random Forest en VALID*

Clase	Precisión (VALID)	Recall (VALID)	F1-score (VALID)	Soporte (VALID)
0 (BENIGN)	1	0.9998	0.9999	4395
1 (RANSOMWARE)	0.7368	0.7132	0.7248	1468
2 (SPYWARE)	0.7772	0.8217	0.7988	1503
3 (TROJAN)	0.7394	0.7196	0.7293	1423

Accuracy	—	—	0.8761	8789
Macro Avg	0.8133	0.8136	0.8132	8789
Weighted Avg	0.8757	0.8761	0.8758	8789

Nota: Elaboración propia

Tabla 19. *Resultados de evaluación de Random Forest en TEST*

Clase	Precisión (TEST)	Recall (TEST)	F1-score (TEST)	Soporte (TEST)
0 (BENIGN)	1	1	1	4395
1 (RANSOMWARE)	0.7374	0.7263	0.7318	1469
2 (SPYWARE)	0.7992	0.8157	0.8074	1503
3 (TROJAN)	0.7405	0.7358	0.7381	1423
Accuracy	—	—	0.88	8790
Macro Avg	0.8193	0.8195	0.8193	8790
Weighted Avg	0.8798	0.88	0.8798	8790

Nota: Elaboración propia

Estos resultados muestran que Random Forest logra un rendimiento muy elevado para la clase BENIGN (0) con métricas cercanas a la perfección, mientras que para las clases de malware (1, 2 y 3) el desempeño es ligeramente menor, aunque mantiene un equilibrio aceptable en precisión y recall.

En términos globales, la exactitud alcanza el 87.6% en validación y 88.0% en prueba, lo cual evidencia una buena capacidad de generalización.

Matriz de confusión

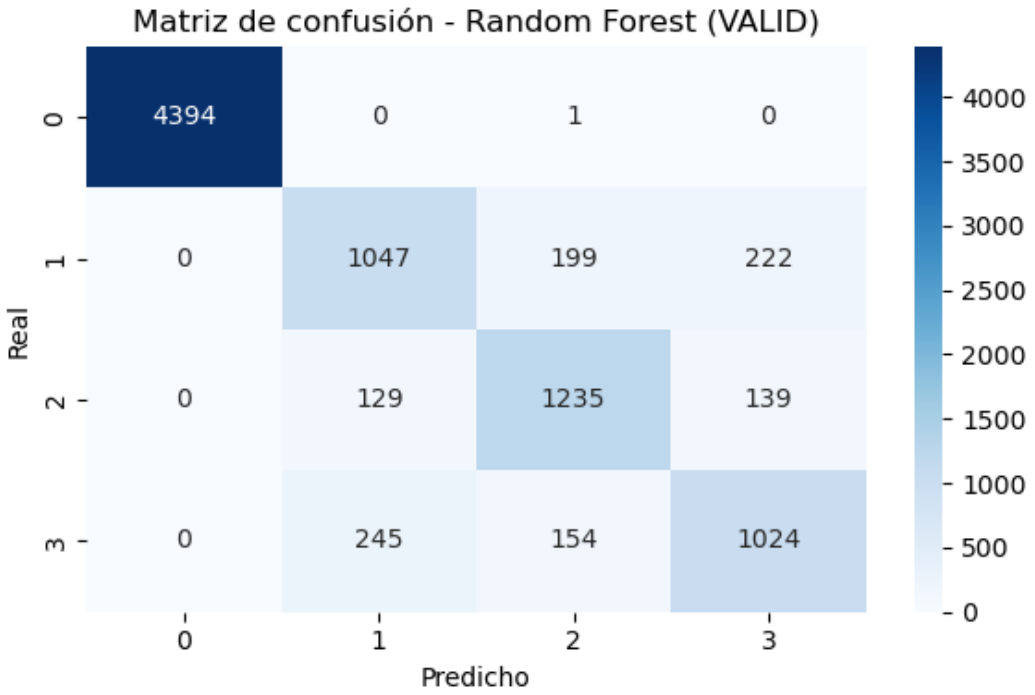
Las Figuras 5 - 6 muestran las matrices de confusión para los conjuntos de validación y prueba, respectivamente. Se observa un excelente desempeño en la identificación de la clase BENIGN (0), con prácticamente nulos errores de clasificación.

Sin embargo, en las clases de malware se presentan errores de confusión moderados:

- Parte de los RANSOMWARE (1) son clasificados como SPYWARE (2) o TROJAN (3).
- Los SPYWARE (2) muestran cierta dispersión hacia las clases 1 y 3.

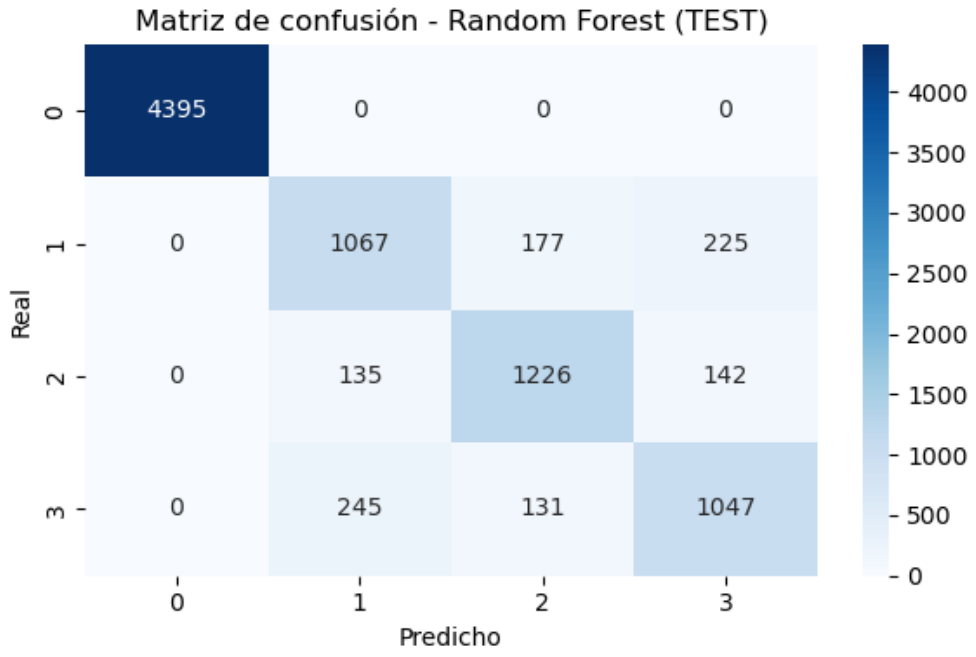
- En el caso de los TROJAN (3), se observa confusión principalmente con RANSOMWARE.

Figura 5. *Matriz de confusión para Random Forest en VALID*



Nota: Elaboración propia

Figura 6. *Matriz de confusión para Random Forest en TEST*



Nota: Elaboración propia

Curvas ROC-AUC (One-vs-Rest)

Con el objetivo de evaluar la capacidad discriminativa del modelo Random Forest en cada clase de la clasificación multiclase, se analizaron las curvas ROC bajo el esquema One-vs-Rest (OvR). En este enfoque, cada clase se considera positiva frente al resto, permitiendo calcular el área bajo la curva (AUC) individual y, posteriormente, una métrica global.

En la fase de validación (VALID), los resultados muestran un desempeño sobresaliente para todas las clases:

- Clase 0 alcanzó un AUC perfecto de 1.00, evidenciando que el modelo no presenta errores al diferenciar esta categoría del resto.
- Clase 1 y Clase 3 obtuvieron valores de 0.95, lo que refleja una muy buena capacidad de discriminación pese a una ligera superposición con otras clases.
- Clase 2 destacó con un AUC de 0.97, confirmando un comportamiento robusto en la predicción de esta categoría.

El promedio general ROC-AUC (OvR) en validación fue de 0.9691, lo cual respalda la solidez del modelo en esta fase.

En la fase de prueba (TEST), los resultados se mantuvieron estables, con una ligera mejora global:

- Clase 0 nuevamente alcanzó un AUC de 1.00, confirmando su separación perfecta.
- Clase 1 obtuvo un AUC de 0.95, mostrando consistencia respecto a la validación.
- Clase 2 registró un valor de 0.97, manteniendo su buen desempeño.
- Clase 3 alcanzó un AUC de 0.96, mostrando una leve mejora frente a la validación.

El promedio global en la fase de prueba fue de 0.9712, confirmando que el modelo mantiene un rendimiento alto y estable fuera de la muestra de entrenamiento.

En la Figura 7 se presenta las curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de validación, mientras que la Figura 8 muestra las curvas para el conjunto de prueba. En ambos casos se observa que las curvas se aproximan al vértice superior izquierdo del gráfico, lo cual indica una alta sensibilidad y especificidad.

Figura 7. *Curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de validación*

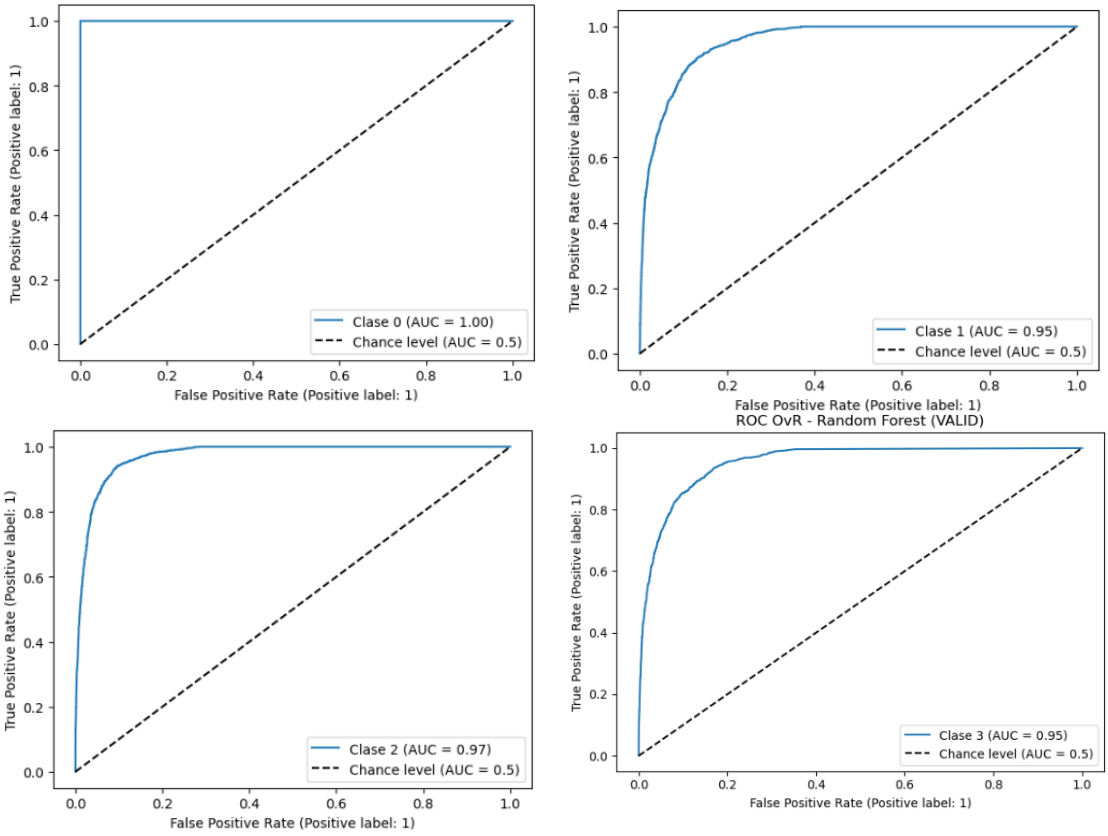
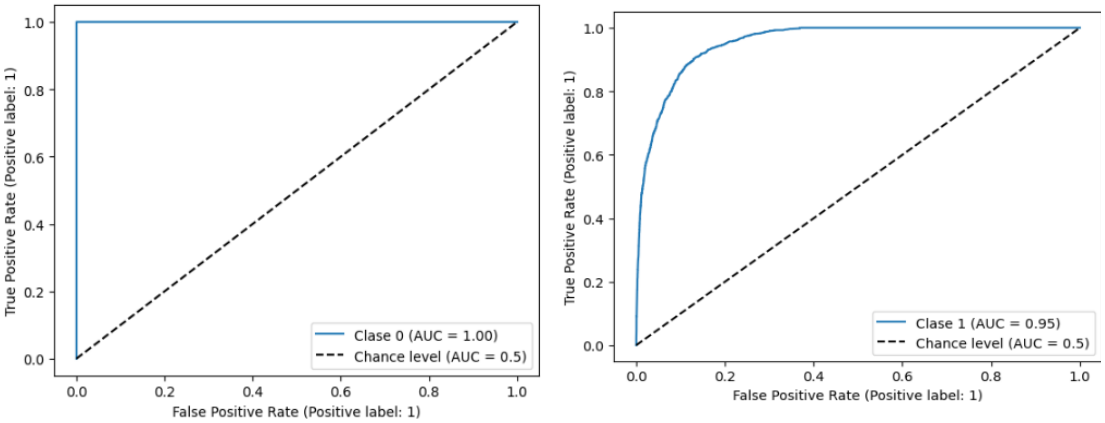
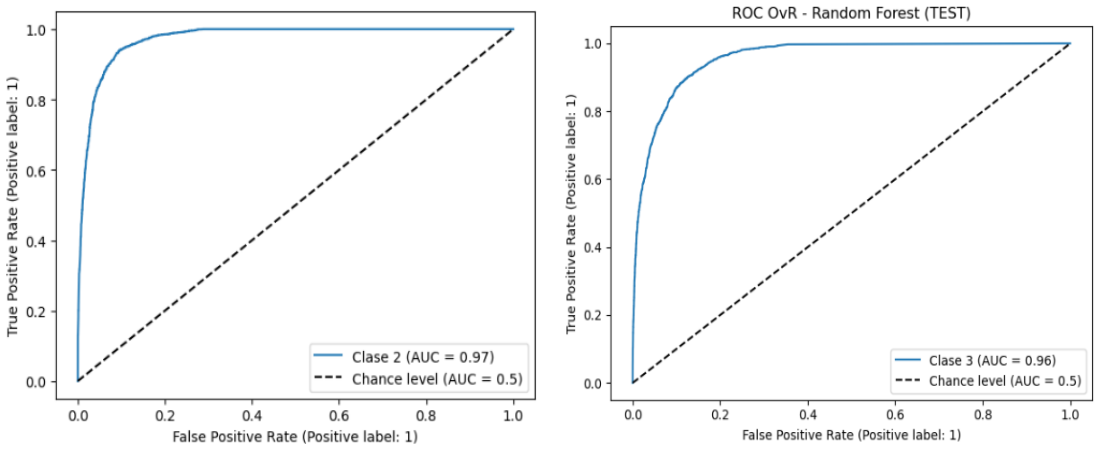


Figura 8. *Curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de test*





Discusión del modelo

Los resultados obtenidos por Random Forest muestran un comportamiento altamente consistente en los conjuntos de validación y prueba, lo que refleja una sólida capacidad de generalización.

En primer lugar, la clase 0 (BENIGN) fue clasificada con métricas prácticamente perfectas: precisión y recall cercanos al 100%, tanto en validación como en prueba. Este rendimiento era esperable, ya que se trata de la clase mayoritaria del dataset y con un patrón de comportamiento más definido frente a las clases de malware.

Por otro lado, las clases de malware (1 = RANSOMWARE, 2 = SPYWARE y 3 = TROJAN) presentaron un desempeño elevado, aunque ligeramente inferior al de la clase BENIGN. Los valores de recall oscilaron entre 71% y 82%, lo que indica que, si bien el modelo logra capturar la mayoría de las instancias maliciosas, aún se producen errores de clasificación cruzada entre familias similares (especialmente entre RANSOMWARE y TROJAN).

En términos globales, la exactitud se mantuvo estable alrededor del 88% en ambos conjuntos, validando la robustez del modelo frente a datos no vistos. Asimismo, el área bajo la curva ROC (AUC) fue sobresaliente, con un valor promedio de 0.9691 en validación y 0.9712 en prueba, lo que evidencia una excelente capacidad discriminativa para diferenciar cada clase de malware del resto.

Random Forest se posiciona como un algoritmo confiable y balanceado, especialmente útil para escenarios donde la correcta detección de malware es prioritaria. No obstante, persiste el reto de mejorar la discriminación entre las distintas familias maliciosas, aspecto que podría abordarse con técnicas de optimización adicionales o modelos híbridos.

3.6.2. SVM-RBF

Métricas globales y por clase

En las Tablas 20 y 21 se presentan los resultados obtenidos por el modelo SVM con kernel RBF, tanto en el conjunto de validación como en el de prueba. Se reportan las métricas de precisión, recall, F1-score y accuracy, calculadas por clase y de manera global.

Tabla 20. *Resultados de evaluación de SVM-RBF en VALID*

Clase	Precisión (VALID)	Recall (VALID)	F1-score (VALID)	Soporte (VALID)
0 (BENIGN)	1	1	1	4395
1 (RANSOMWARE)	0.5473	0.5238	0.5353	1468
2 (SPYWARE)	0.6667	0.5802	0.6204	1503
3 (TROJAN)	0.5592	0.6606	0.6057	1423
Accuracy	—	—	**0.7937**	8789
Macro Avg	0.6933	0.6911	0.6904	—
Weighted Avg	0.796	0.7937	0.7936	—

Nota: Elaboración propia

Tabla 21. *Resultados de evaluación de SVM-RBF en TEST*

Clase	Precisión (TEST)	Recall (TEST)	F1-score (TEST)	Soporte (TEST)
0 (BENIGN)	1	1	1	4395
1 (RANSOMWARE)	0.5348	0.5126	0.5235	1469

2 (SPYWARE)	0.6746	0.5489	0.6053	1503
3 (TROJAN)	0.5488	0.6803	0.6075	1423
Accuracy	—	—	**0.7896**	8790
Macro Avg	0.6895	0.6854	0.6841	—
Weighted Avg	0.7936	0.7896	0.7893	—

Nota: Elaboración propia

Estos resultados muestran que, al igual que en Random Forest, el modelo logra un desempeño perfecto en la clasificación de la clase BENIGN (0). Sin embargo, para las clases de malware, el rendimiento fue inferior: el recall osciló entre 0.52 y 0.68, lo que refleja mayores dificultades en la correcta identificación de estas amenazas. Aun así, la exactitud global se mantuvo cercana al 79% en ambos conjuntos, lo que indica un comportamiento relativamente estable.

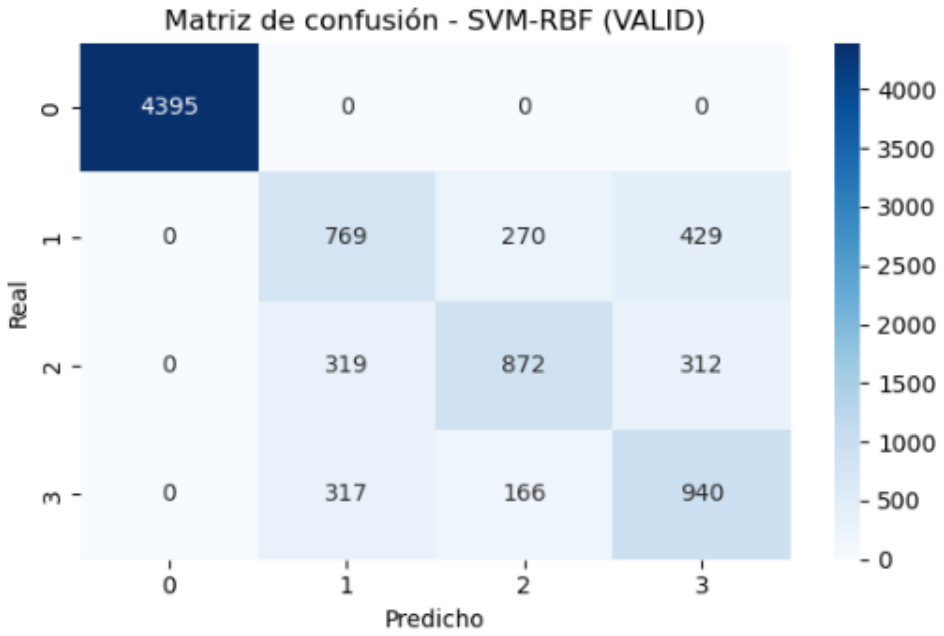
Matriz de confusión

Las Figuras 9 y 10 presentan las matrices de confusión para los conjuntos de validación y prueba.

En el caso de VALID, se observa un desempeño perfecto para la clase BENIGN (0). Sin embargo, existe confusión significativa entre las clases de malware: parte de los RANSOMWARE (1) se clasifican como SPYWARE (2) o TROJAN (3), mientras que los SPYWARE presentan confusión distribuida hacia las clases 1 y 3. Finalmente, los TROJAN muestran confusión hacia RANSOMWARE y SPYWARE.

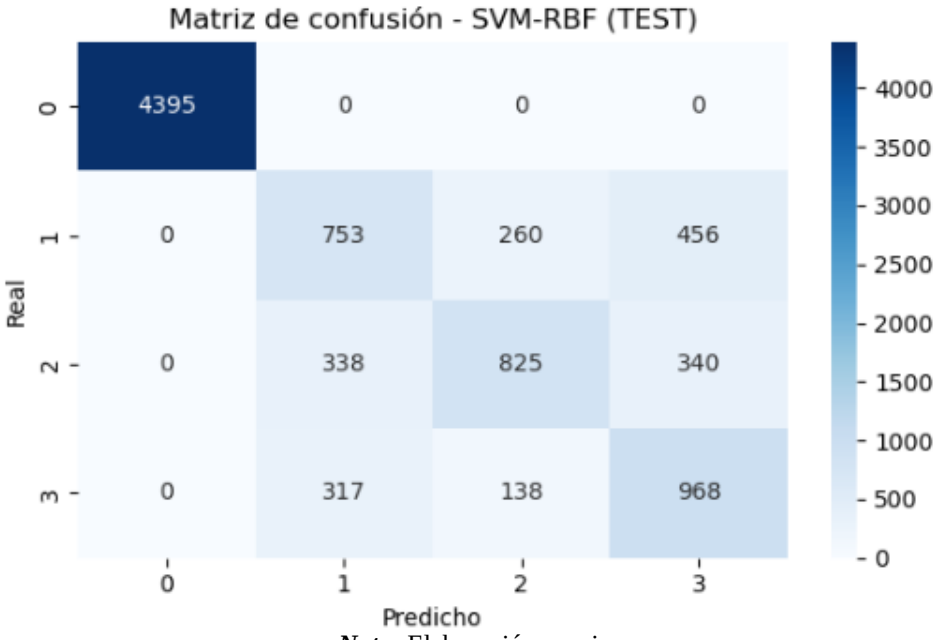
En el caso de TEST, se confirma este patrón de comportamiento: clase 0 clasificada perfectamente, pero con errores de clasificación cruzada en las clases de malware.

Figura 9. *Matriz de confusión para SVM-RBF en VALID*



Nota: Elaboración propia

Figura 10. *Matriz de confusión para SVM-RBF en TEST*



Nota: Elaboración propia

Curvas ROC-AUC (One-vs-Rest)

Con el fin de evaluar la capacidad discriminativa del modelo SVM-RBF en cada una de las clases, se calcularon las curvas ROC bajo el esquema One-vs-Rest (OvR). En este

enfoque, cada clase se considera positiva frente al resto, permitiendo medir la sensibilidad y especificidad del modelo de manera individual.

En la fase de validación (VALID), el desempeño fue consistente entre las clases de malware, con valores de AUC que oscilaron entre 0.90 y 0.92.

- La clase 0 (BENIGN) alcanzó nuevamente un AUC perfecto de 1.00, lo que indica una separación completa de esta clase frente al resto.
- La clase 1 (RANSOMWARE) obtuvo un valor de 0.90, mostrando una capacidad discriminativa aceptable pero menor respecto a otros algoritmos más robustos.
- La clase 2 (SPYWARE) registró un AUC de 0.92, lo que refleja un buen desempeño.
- La clase 3 (TROJAN) alcanzó un valor de 0.91, destacando por su consistencia en comparación con las demás clases de malware.

El promedio general ROC-AUC (OvR) en validación fue de 0.9309, lo que confirma un rendimiento globalmente alto y balanceado.

En el conjunto de prueba (TEST), los resultados fueron prácticamente idénticos a los observados en validación, manteniendo el promedio global de 0.9309, lo que respalda la estabilidad y capacidad de generalización del modelo SVM-RBF.

En la Figura 11 se presentan las curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de validación, mientras que la Figura 12 muestra las curvas equivalentes para el conjunto de prueba. En ambos casos, las curvas se ubican muy cerca del vértice superior izquierdo, lo que indica una combinación adecuada de sensibilidad y especificidad.

Figura 11. *Curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de validación (SVM-RBF)*

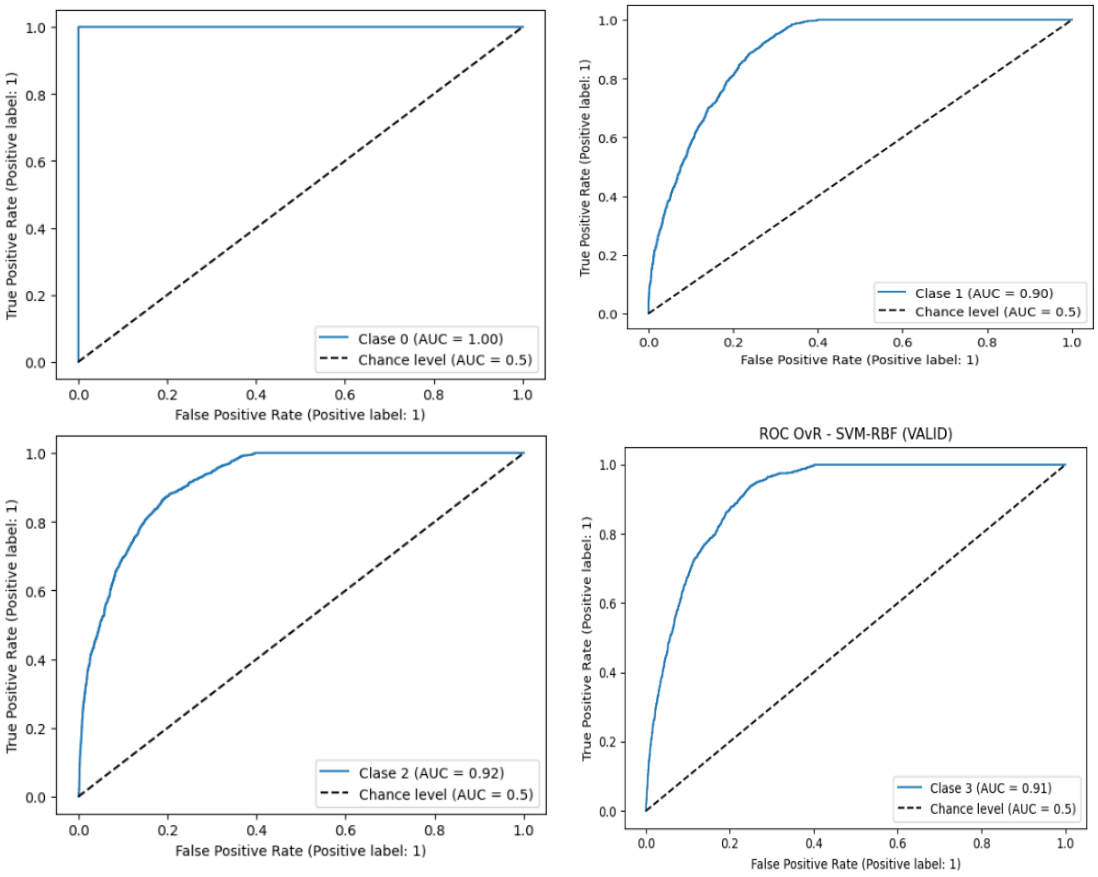
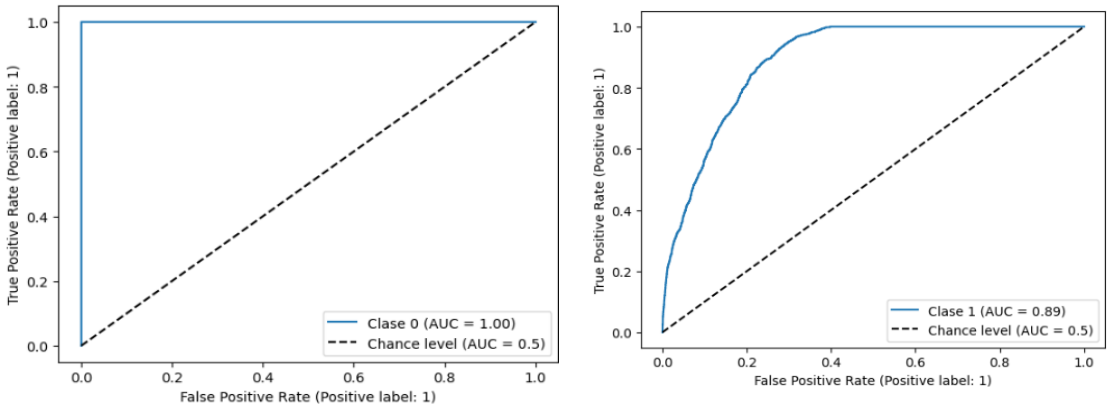
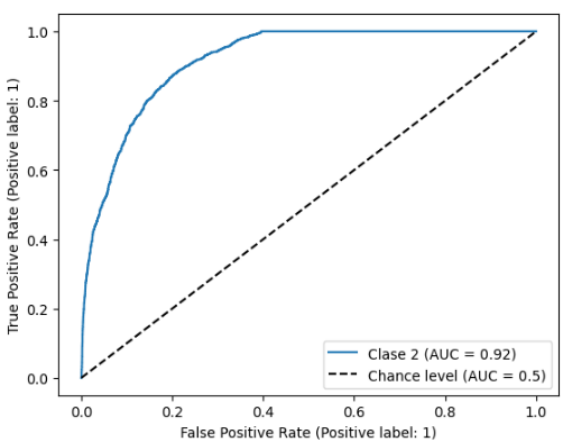
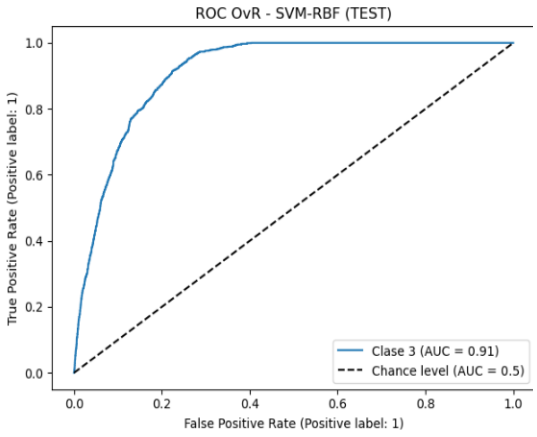


Figura 12. *Curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de prueba (SVM-RBF)*







Discusión del modelo

El modelo SVM con kernel RBF mostró un rendimiento sólido en la clasificación binaria frente al resto (One-vs-Rest), especialmente para la clase BENIGN (0), la cual fue identificada de manera perfecta, con precisión y recall iguales a 1.00. Este resultado era esperable debido a su predominio en el dataset y a la clara separación de sus características respecto a las clases de malware.

En contraste, el desempeño en las clases de malware fue considerablemente menor. Los valores de recall oscilaron entre 0.51 y 0.68, lo que revela dificultades del modelo para diferenciar adecuadamente entre amenazas como RANSOMWARE (1), SPYWARE (2) y TROJAN (3). Esta situación se evidenció también en las matrices de confusión, donde se observaron errores de clasificación cruzada entre estas tres categorías.

En términos globales, la exactitud se mantuvo estable entre los conjuntos de validación y prueba, con valores de 79.3% y 78.9%, respectivamente. Aunque inferiores a los alcanzados por Random Forest, estos resultados reflejan una consistencia en el comportamiento del modelo fuera de la muestra de entrenamiento.

Por otro lado, el análisis de las curvas ROC-AUC mostró que la capacidad discriminativa del modelo fue alta y homogénea entre las clases, con valores entre 0.90 y 0.92 para las categorías de malware, y un promedio general de 0.9309 en ambos conjuntos. Estos resultados confirman que, pese a las dificultades en la clasificación fina entre tipos de malware, el modelo SVM-RBF posee una capacidad robusta para distinguir entre clases cuando se mide desde la perspectiva global.

3.6.3. XGBoost

Métricas globales y por clase

En la Tabla 22 se presentan los resultados obtenidos con el modelo XGBoost en la fase de validación (VALID). Se observa que la clase 0 alcanza un desempeño perfecto con valores de precisión, recall y F1-score de 1.0000, mientras que las clases 1, 2 y 3 mantienen métricas balanceadas, con valores de F1-score entre 0.73 y 0.79. El accuracy global en esta fase se ubica en 87.93 %, lo que evidencia un buen nivel de ajuste del modelo.

Tabla 22. *Resultados de métricas de XGBoost en VALID*

Clase	Precisión	Recall	F1-score	Soporte
0	1	1	1	4395
1	0.7481	0.7323	0.7401	1468
2	0.7768	0.8197	0.7977	1503
3	0.7484	0.721	0.7344	1423
Accuracy	-	-	0.8793	8789
Macro avg	0.8183	0.8182	0.8181	8789
Weighted avg	0.879	0.8793	0.879	8789

Nota: Elaboración propia

De forma similar, en la Tabla 23 se presentan los resultados en la fase de prueba (TEST). Los valores mantienen coherencia con la validación, alcanzando un accuracy de 87.84 %, con métricas consistentes en todas las clases. La clase 2 presenta un recall sobresaliente de 0.8184, mientras que la clase 1 muestra un desempeño ligeramente menor respecto a las demás.

Tabla 23. *Resultados de métricas de XGBoost en TEST*

Clase	Precisión	Recall	F1-score	Soporte
0	0.9993	1	0.9997	4395
1	0.7407	0.7216	0.731	1469
2	0.7864	0.8184	0.8021	1503
3	0.7416	0.728	0.7348	1423

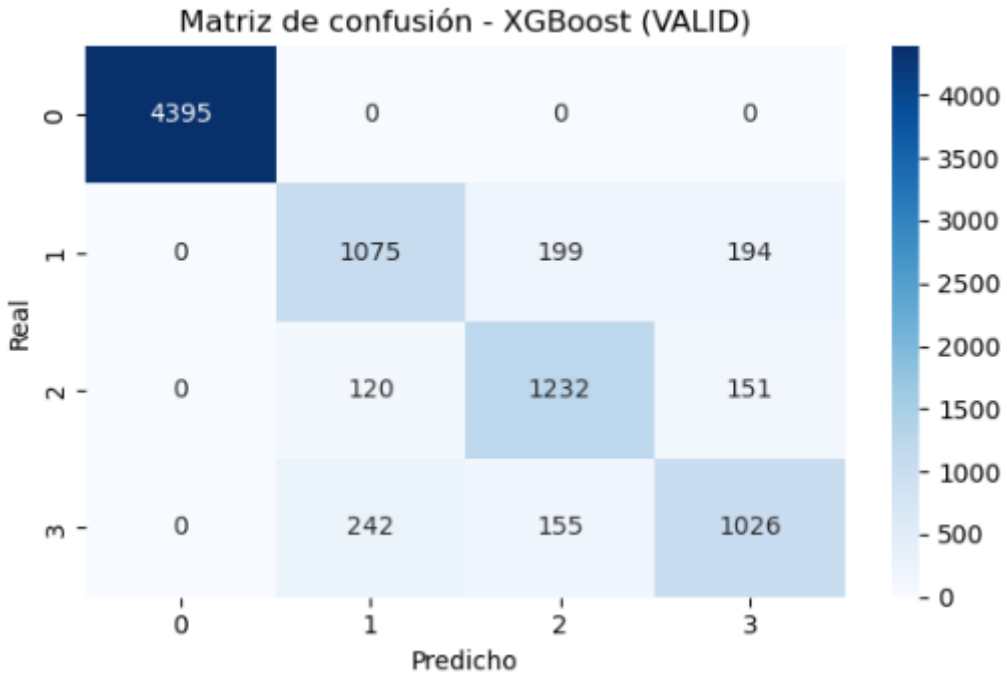
Accuracy	-	-	0.8784	8790
Macro avg	0.817	0.817	0.8169	8790
Weighted avg	0.878	0.8784	0.8781	8790

Nota: Elaboración propia

Matriz de confusión

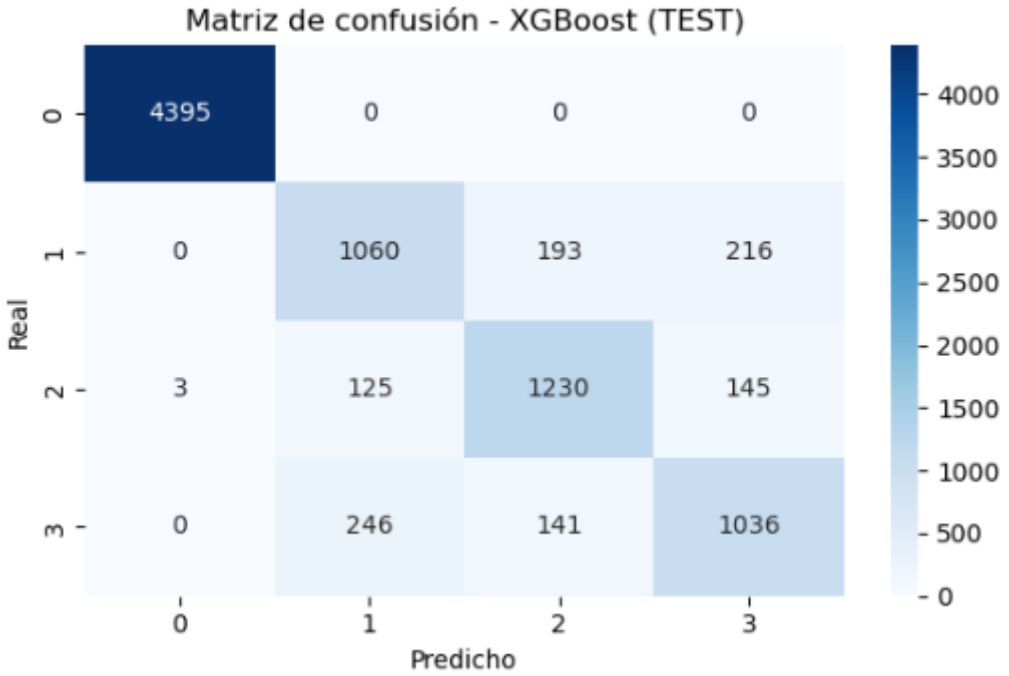
A continuación, se presentan las matrices de confusión correspondientes a la fase de validación y prueba. Estas figuras permiten visualizar la distribución de aciertos y errores de clasificación para cada una de las clases.

Figura 13. *Matriz de confusión para XGBoost en VALID*



Nota: Elaboración propia

Figura 14. *Matriz de confusión para XGBoost en TEST*



Nota: Elaboración propia

Curvas ROC-AUC (One-vs-Rest)

Con el propósito de evaluar la capacidad discriminativa de XGBoost en la clasificación multiclase, se generaron las curvas ROC bajo el enfoque One-vs-Rest (OvR). Este análisis permite observar el desempeño del modelo al considerar cada clase como positiva frente al resto.

En la fase de validación (VALID) se obtuvieron resultados sobresalientes:

- La clase 0 (BENIGN) alcanzó un AUC perfecto de 1.00, mostrando una separación completa respecto a las demás categorías.
- La clase 1 (RANSOMWARE) logró un AUC de 0.96, reflejando una discriminación robusta.
- La clase 2 (SPYWARE) obtuvo un AUC de 0.97, destacándose como una de las más sólidas.
- La clase 3 (TROJAN) presentó un AUC de 0.96, mostrando un rendimiento muy estable.

El promedio global ROC-AUC (OvR) en validación fue de 0.9722, confirmando un excelente desempeño general del modelo.

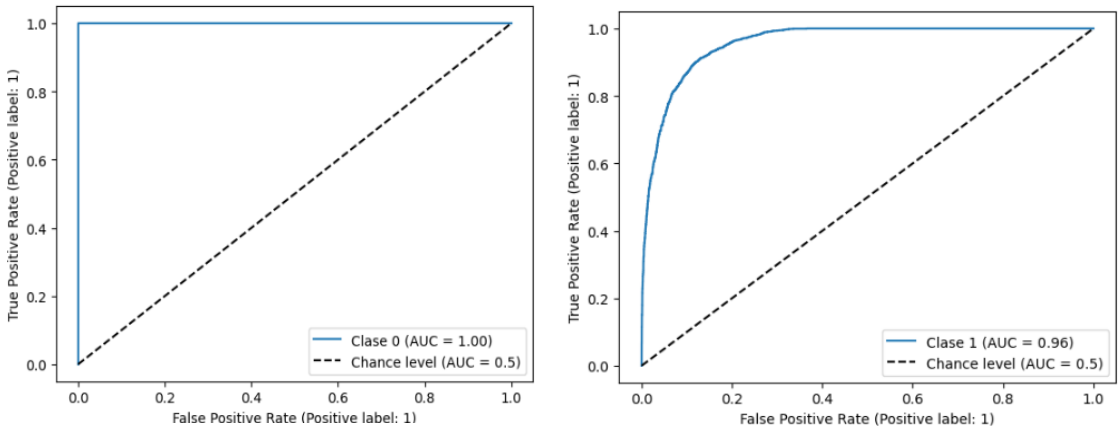
En la fase de prueba (TEST), los valores se mantuvieron prácticamente idénticos:

- Clase 0 (BENIGN) con AUC de 1.00.
- Clase 1 (RANSOMWARE) con AUC de 0.96.
- Clase 2 (SPYWARE) con AUC de 0.97.
- Clase 3 (TROJAN) con AUC de 0.96.

El promedio global ROC-AUC (OvR) en test alcanzó 0.9736, lo cual demuestra la capacidad de generalización del modelo, manteniendo estabilidad entre las fases de validación y prueba.

En la Figura 15 se presentan las curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de validación, mientras que en la Figura 16 se muestran las curvas equivalentes para el conjunto de prueba. En ambos casos, las curvas se aproximan al vértice superior izquierdo, lo que indica un comportamiento altamente sensible y específico.

Figura 15. *Curvas ROC-AUC correspondientes a las cuatro clases en el conjunto de validación*



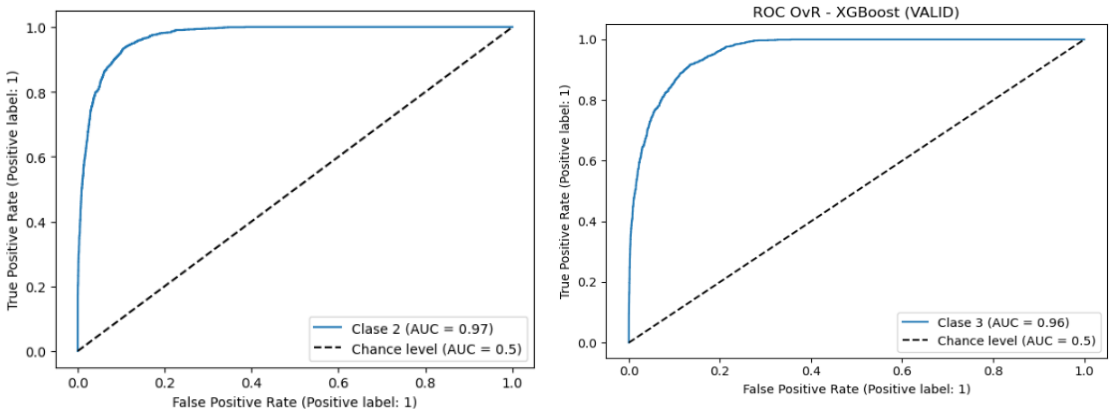
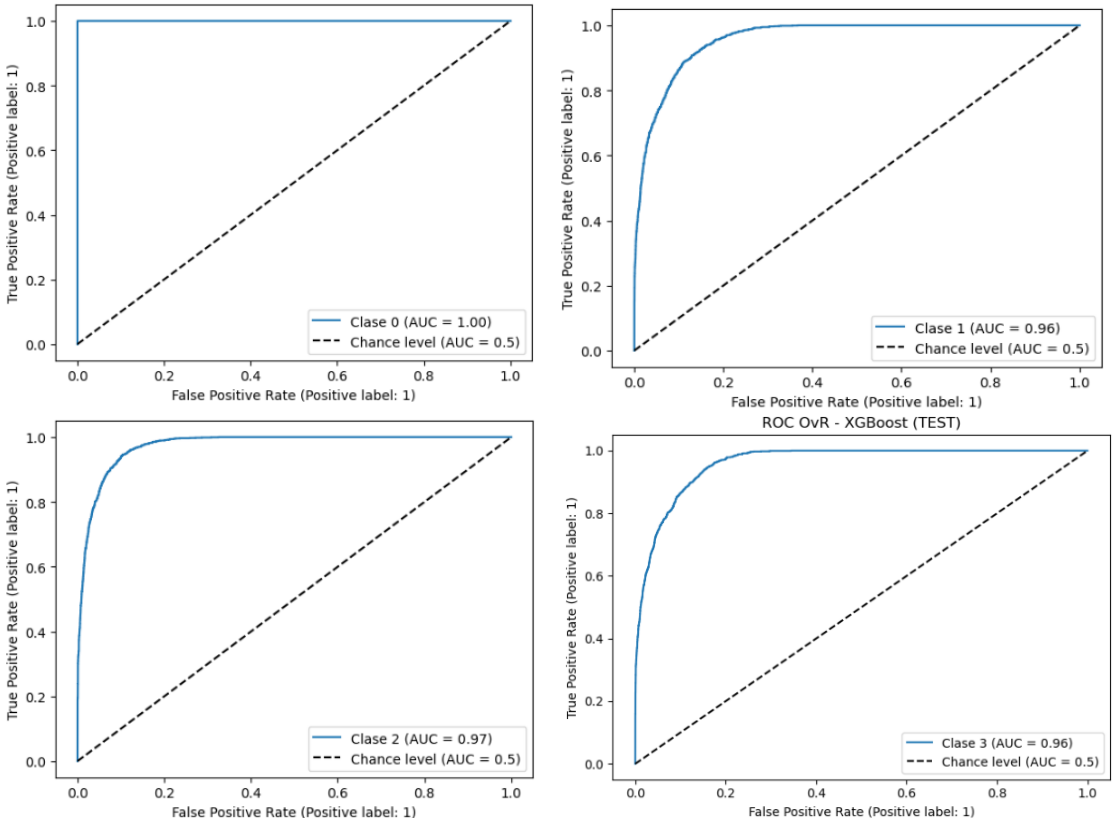


Figura 16. *Curvas equivalentes para el conjunto de test*



Discusión del modelo

Los resultados obtenidos con el modelo XGBoost permiten destacar varios aspectos relevantes de su desempeño:

- Clase 0 (BENIGN): se clasificó de manera prácticamente perfecta, alcanzando valores cercanos al 100 % en precisión, recall y F1-score tanto en validación como

en prueba. Este comportamiento era esperable debido al predominio de esta clase en el dataset, lo que facilita su identificación.

- Clases de malware (1=RANSOMWARE, 2=SPYWARE, 3=TROJAN): el rendimiento fue alto pero menor en comparación con la clase BENIGN. En particular, SPYWARE (clase 2) alcanzó los valores más elevados de recall (0.8197 en VALID y 0.8184 en TEST), evidenciando que el modelo logra identificarla con gran eficacia. En contraste, la clase RANSOMWARE (1) mostró un desempeño más limitado, con recall alrededor de 0.72, reflejando la mayor dificultad del modelo para diferenciarla de otras amenazas.

- Estabilidad entre VALID y TEST: el accuracy global se mantuvo estable en torno al 88 % en ambas fases, confirmando que el modelo presenta una buena capacidad de generalización sin caer en sobreajuste.

- Capacidad discriminativa (ROC-AUC): los valores globales de 0.9722 en validación y 0.9736 en prueba confirman la solidez del modelo, indicando una excelente capacidad para distinguir entre clases. Además, todas las categorías de malware alcanzaron AUC entre 0.96 y 0.97, lo cual respalda la consistencia del rendimiento en escenarios multicategoría.

En conjunto, XGBoost se posiciona como uno de los modelos más competitivos del análisis, ya que combina un alto accuracy con una gran capacidad discriminativa, manteniendo un equilibrio adecuado entre precisión y recall en las clases de malware.

3.7. Interpretabilidad y selección de variables

Ranking de variables más relevantes

Para explorar la interpretabilidad de los modelos, se extrajo el ranking de importancia de variables en Random Forest y XGBoost, además de aplicar métodos estadísticos como Chi^2 y Mutual Information.

En el caso de Random Forest, las variables más influyentes fueron:

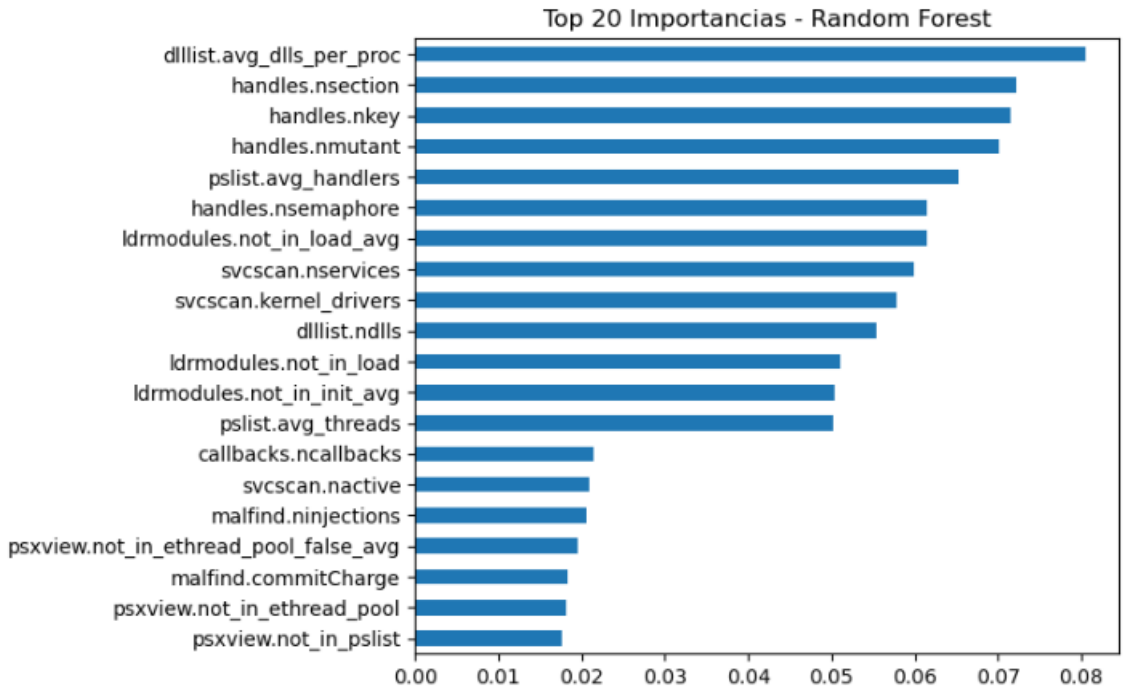
- dlllist.avg_dlls_per_proc (promedio de DLLs por proceso).

- handles.nsection, handles.nkey y handles.nmutant (atributos de handles del sistema).
- pslist.avg_handlers y pslist.avg_threads (características de procesos).
- svcscan.nservices y svcscan.kernel_drivers (atributos del escaneo de servicios).

Por su parte, en XGBoost predominó claramente svcscan.nservices como la característica más relevante, con un peso muy superior al resto. A esta le siguieron svcscan.kernel_drivers, handles.nsection, malfind.uniqueInjections y dlllist.avg_dlls_per_proc.

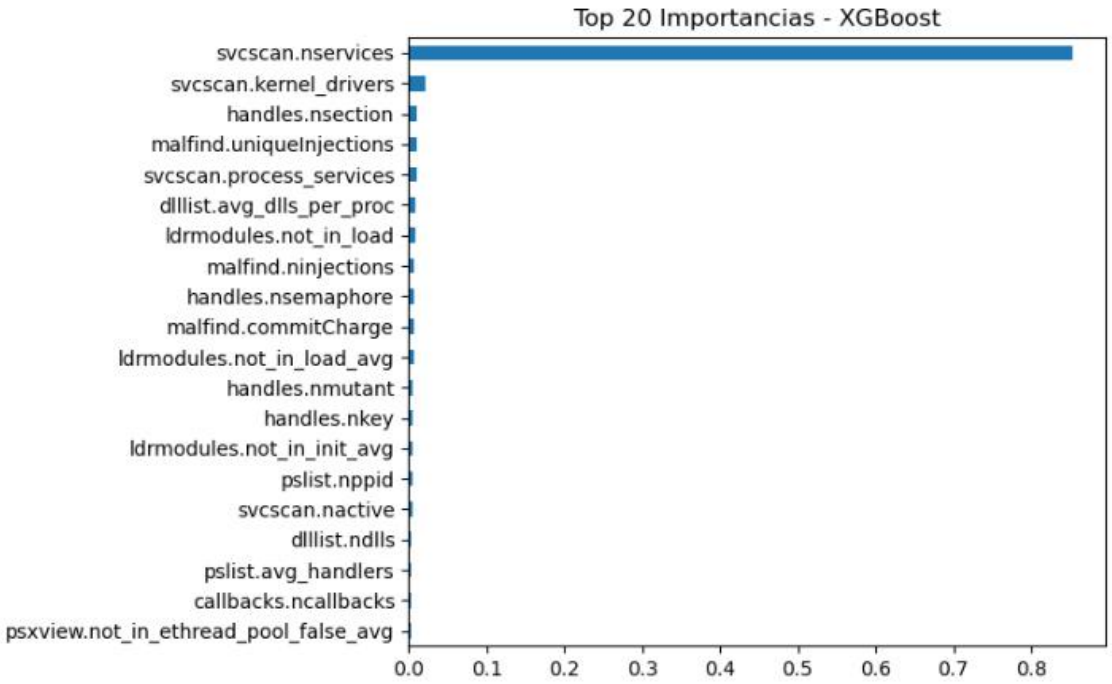
Los métodos de selección de variables basados en estadística, como Chi² y Mutual Information, confirmaron la relevancia de varias de estas características, aunque con algunas diferencias de orden. Por ejemplo, handles.nmutant y dlllist.ndlls aparecieron consistentemente como muy significativas en Chi², mientras que Mutual Information resaltó dlllist.avg_dlls_per_proc, handles.nsemaphore y ldrmodules.not_in_load_avg.

Figura 17. *Top 20 de Random Forest*



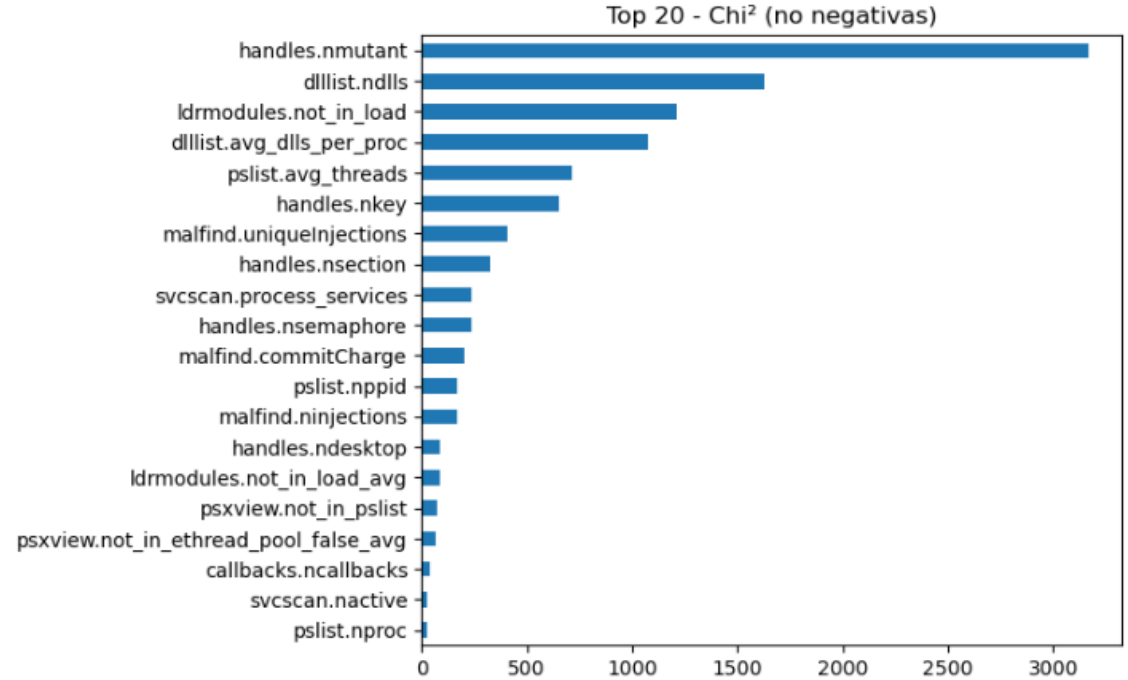
Nota: Elaboración propia

Figura 18. *Top 20 de XGBoost*

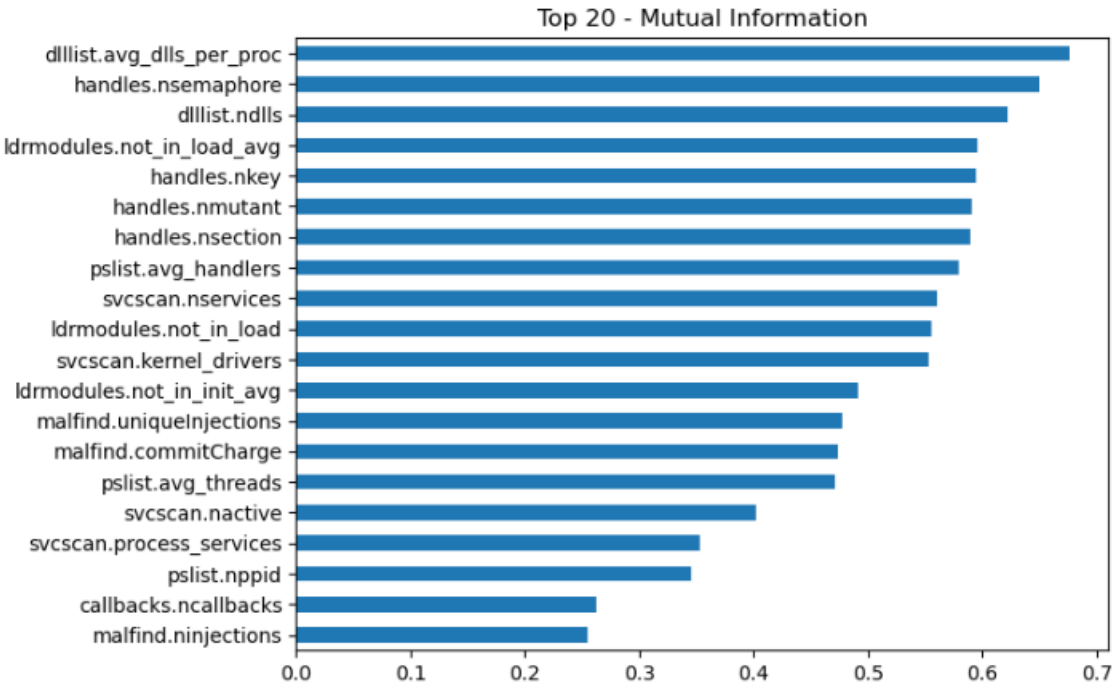


Nota: Elaboración propia

Figura 19. *Top 20 de Chi² y Mutual Information*



Nota: Elaboración propia



Nota: Elaboración propia

Tabla 24. *comparativa global (promedio de rankings)*

	mean_rank	**RF_importance**	**XGB_importance**	**Chi2**	**MutualInfo**
dlllist.avg_dlls_per_proc	3	0.0805	0.0079	1074.802	0.6769
handles.nsection	5	0.0722	0.0102	324.0062	0.5901
handles.nmutant	5.75	0.0701	0.0051	3166.1653	0.5902
handles.nkey	6.75	0.0715	0.0045	649.2326	0.5943
handles.nsemaphore	6.75	0.0614	0.0069	233.0893	0.65
ldrmodules.not_in_load	7.75	0.0511	0.0078	1209.0667	0.5554
dlllist.ndlls	8	0.0554	0.0039	1627.327	0.6218
ldrmodules.not_in_load_avg	9.25	0.0614	0.006	85.9646	0.5957
svcscan.nservices	10.25	0.0598	0.8529	5.6111	0.5606
malfind.uniqueInjections	11.25	0.0153	0.0102	407.5734	0.4776

svcscan.kernel_drivers	11.75	0.0578	0.022	0.7585	0.5533
malfind.commitCharge	13.25	0.0183	0.0065	202.2758	0.4735
pslist.avg_handlers	13.25	0.0653	0.0037	17.7603	0.5795
svcscan.process_services	14	0.0073	0.0099	238.8856	0.3534
pslist.avg_threads	14.25	0.0502	0.0027	711.222	0.4716
malfind.ninjections	14.25	0.0206	0.0071	166.4224	0.255
ldrmodules.not_in_init_avg	15.5	0.0504	0.0045	4.6879	0.492
svcscan.nactive	16.5	0.021	0.004	22.8918	0.4019
pslist.nppid	16.75	0.0127	0.0041	170.4453	0.3457
callbacks.ncallbacks	17.5	0.0213	0.0032	35.7873	0.2621

Nota: Elaboración propia

Discusión sobre las características

El análisis conjunto evidencia que las DLLs cargadas por proceso, los handles relacionados con secciones, claves y mutex, así como los atributos de servicios del sistema, son las variables más útiles para diferenciar entre malware y procesos benignos.

En particular:

- Los mecanismos de inyección de código (malfind.*) y los módulos no cargados en memoria (ldrmodules.*) resultan indicadores clave para identificar comportamiento malicioso.
- XGBoost tiende a concentrar gran parte de su poder predictivo en svcscan.nservices, mientras que Random Forest distribuye la importancia entre múltiples variables.
- Esto sugiere que, aunque ambos modelos coinciden en varias variables críticas, difieren en la manera en que las aprovechan para la clasificación.
- SVM-RBF: El modelo SVM con kernel RBF no se incluye en este análisis de interpretabilidad porque no dispone de un atributo nativo equivalente a

feature_importances_. Sus parámetros internos (vectores soporte y coeficientes del kernel) no se traducen en importancias directas de variables. Para este tipo de modelos, sería necesario aplicar técnicas adicionales como Permutation Importance o SHAP, que implican un costo computacional más elevado.

No obstante, los métodos estadísticos aplicados (Chi² y Mutual Information) permiten identificar variables significativas de manera independiente del algoritmo, por lo que sus resultados también sirven como referencia indirecta para el modelo SVM.

3.8. Evaluación de eficiencia computacional

Resultados comparativos

En la Tabla 25 se resumen los tiempos de entrenamiento, predicción, uso de memoria e inferencia para cada modelo.

Tabla 25. *Comparación de eficiencia computacional entre modelos*

Modelo	Tiempo Entrenamiento (s)	Memoria Entrenamiento (MB)	Tiempo Predicción (s)	Tiempo Probabilidad (s)	Memoria Inferencia (MB)	Throughput (samples/s)
Random Forest	74.9003	752.75	0.3727	0.3112	6.37	23,583
SVM-RBF	1587.0378	0	43.54	49.56	3.45	201
XGBoost	16.1478	52.34	0.125	0.137	0.02	70,295

Nota: Elaboración propia

Discusión sobre eficiencia

- XGBoost resultó el modelo más eficiente, con un tiempo de entrenamiento muy bajo (16 s) y un throughput sobresaliente de más de 70 mil muestras por segundo en inferencia. Además, su uso de memoria fue mínimo en comparación con Random Forest y SVM.

- Random Forest tuvo un rendimiento intermedio, con un entrenamiento relativamente rápido y buena velocidad de inferencia, aunque con un consumo de memoria significativamente mayor (753 MB).

- SVM-RBF fue el modelo menos eficiente: requirió más de 1,500 segundos para entrenar, con tiempos de inferencia muy altos y un throughput reducido a apenas 201 muestras por segundo, lo cual lo hace poco práctico para escenarios de despliegue a gran escala.

Entonces, XGBoost logra el mejor balance entre precisión y eficiencia, siendo no solo el más preciso en varias métricas, sino también el más rápido y ligero en términos de recursos computacionales. Esto lo convierte en el candidato más adecuado para su implementación en entornos productivos.

CONCLUSIONES

Quedó claro que los algoritmos de aprendizaje automático se comportaban de manera diferente al detectar malware, es decir, los modelos conjuntos como Random Forest y XGBoost destacaron por encima del resto, ya que, lograron un buen equilibrio entre precisión, eficiencia computacional y solidez, lo que los convirtió en opciones sólidas para hacer frente a escenarios de malware diversos y desafiantes.

Al comparar los algoritmos, XGBoost y Random Forest ofrecieron una mayor precisión en la clasificación de ransomware, spyware y troyanos, superaron a los árboles de decisión en cuanto a capacidad predictiva.

El análisis de características reveló algo importante, que los metadatos y las variables de comportamiento de los entornos virtuales tenían un gran impacto en el rendimiento del modelo, además, el uso de pruebas estadísticas y la importancia de las características ayudaron a reducir los atributos a un conjunto más pequeño y potente con una gran capacidad discriminatoria.

Por último, el análisis de eficiencia reveló una interesante compensación, los modelos conjuntos tardan más tiempo en entrenarse, pero una vez entrenados, proporcionan una inferencia rápida y un uso razonable de la memoria. Esto los hace escalables y prácticos para entornos del mundo real, ya que, los árboles de decisión, por otro lado, eran más baratos de ejecutar, pero tenían una menor precisión y robustez.

RECOMENDACIONES

Aplicar Random Forest y XGBoost en escenarios reales de detección de malware, dando prioridad a su uso en sistemas de seguridad que requieren una alta precisión y una fuerte generalización frente a nuevas amenazas.

Fomentar futuras investigaciones que profundicen en la selección y creación de características, incorporando más fuentes de datos como el tráfico de red y los registros del sistema. Esto mejoraría aún más la capacidad de los modelos para predecir amenazas.

Trabajar en la mejora de los tiempos de entrenamiento utilizando técnicas como la paralelización o la reducción de dimensionalidad, de esta manera, la eficiencia computacional no será una barrera para el uso de modelos conjuntos en sistemas a gran escala.

Por último, complementar estos resultados con pruebas en entornos adversarios simulados, como ataques de evasión o perturbación adversaria, esto ayudará a confirmar la solidez real de los algoritmos seleccionados en la práctica.

REFERENCIAS

Carrizo, L., & Álvarez, M. (10 de Octubre de 2024). *Using Free Software and Machine Learning to Improve Intrusion Detection in a Network*. Obtenido de https://polodelconocimiento.com/ojs/index.php/es/article/download/8136/pdf

Chancay, A. (2024). *Análisis de malware rat (remote access trojan) mediante técnicas de detección estáticas y dinámicas en plataformas android*. Obtenido de https://repositorio.upse.edu.ec/xmlui/bitstream/handle/46000/12928/UPSE-TTI-2025-0006.pdf?sequence=1&isAllowed=y

Cruz, M. (Septiembre de 2021). *El malware y las redes sociales*. Obtenido de https://rua.ua.es/dspace/bitstream/10045/118147/1/El_malware_y_las_redes_sociales_Cruz_Morgado_Melanie_Mariam.pdf

Del Barrio, D. (2022). *Aplicación del aprendizaje automático en modelos de materia activa*. Obtenido de https://oa.upm.es/70193/

DMA. (2021). *Aprendizaje Automático*. Obtenido de Universidad Politécnica de Madrid (Departamento de Matemática Aplicada): https://dcain.etsin.upm.es/~carlos/bookAA/03.1_Clustering-K-Means.html

Dunsin, D., & Chahien, M. (17 de Enero de 2025). *Aprendizaje por refuerzo para una investigación eficiente y eficaz del malware durante la respuesta a incidentes cibernéticos*. Obtenido de https://www.sciencedirect.com/science/article/pii/S2667295225000030

García, B. (10 de Junio de 2021). *Aprendizaje Automático para Detección de Tráfico IoT Anómalo, Spam y Malware*. Obtenido de https://riull.ull.es/xmlui/bitstream/handle/915/24203/Aprendizaje%20automatico%20para%20deteccion%20de%20trafico%20iot%20anomalo%2C%20spam%20y%20malware.pdf?sequence=1

Güven, M. (2024). *Análisis dinámico de malware mediante un entorno sandbox, registros de tráfico de red e inteligencia artificial.* Obtenido de https://www.researchgate.net/publication/384379416_Dynamic_Malware_Analy

sis_Using_a_Sandbox_Environment_Network_Traffic_Logs_and_Artificial_Intelligence

Herrera, J., & Hernández, M. (2023). *Deployment of Ransomware Detection Using Dynamic Analysis and Machine Learning*. Obtenido de https://www.researchgate.net/publication/371999265_Deployment_of_Ransomware_Detection_Using_Dynamic_Analysis_and_Machine_Learning

IBM. (9 de Mayo de 2024). *¿Qué es XGBoost?* Obtenido de International Business Machines Corporation: https://www.ibm.com/es-es/think/topics/xgboost

Jácome, G., & González, V. (2022). *Detección de amenazas de seguridad en una red corporativa utilizando algoritmos de machine learning* . Obtenido de https://dialnet.unirioja.es/descarga/articulo/8955447.pdf

Marín, J. (2024). *Integración de soluciones de ciberseguridad en software libre para la protección de las PYMES aplicado para la funación de atención al discapacitado (F.A.D)*. Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/29255/2/TTS2075.pdf

Mite, E. (2025). *Identificación de modelos de aprendizaje automático para la detección de ransomware en empresas basado en bibliografía*. Obtenido de Universidad Politécnica Salesiana: https://dspace.ups.edu.ec/bitstream/123456789/29949/1/UPS-GT006113.pdf

Nieto, A. (2021). *Algoritmos de aprendizaje automático: un estudio de su difusión y utilización*. Obtenido de https://oa.upm.es/68484/

Obaido, G., & Mienye, P. (2024). *Aprendizaje automático supervisado en el descubrimiento y desarrollo de fármacos: algoritmos, aplicaciones, desafíos y perspectivas*. Obtenido de https://www.sciencedirect.com/science/article/pii/S2666827024000525

Ocampo, O., & Mendoza, V. (2021). *Identificación de brechas en gestión de la innovación en empresas de Alimentos y Bebidas en caldas* . Obtenido de https://dialnet.unirioja.es/descarga/articulo/8055081.pdf

Pérez, Á. (2024). *El algoritmo SVM y sus aplicaciones empresariales*. Obtenido de OBS Business School: https://www.obsbusiness.school/blog/el-algoritmo-svm-y-sus-aplicaciones-empresariales

Recordon, A., & Ruiz, S. (2021). *Detección y Clasificación de Zero-Day Malware a través de Data Mining y Machine Learning*. Obtenido de https://sedici.unlp.edu.ar/bitstream/handle/10915/140918/Documento_completo.pdf?sequence=1

Rojas, B., & Rodríguez, C. (14 de 01 de 2021). *Aprendizaje por defensa reactiva: el nuevo modelo de entrenamiento contra malware*. Obtenido de https://dspace.ulead.ac.cr/items/f6c63df0-4f07-40fd-bcf1-a3d3eadc3ecf

Salem, A., & Azzam, S. (2024). *Avance de la ciberseguridad: una revisión exhaustiva de las técnicas de detección impulsadas por la IA*. Obtenido de https://journalofbigdata.springeropen.com/articles/10.1186/s40537-024-00957-y

Salman, H. (June de 2024). *Random Forest Algorithm Overview*. Obtenido de Babylonian Journal of Machine Learning: https://www.researchgate.net/publication/382419308_Random_Forest_Algorithm_Overview

SCRIBD. (11 de Marzo de 2022). *Tipos de malware*. Obtenido de https://es.scribd.com/presentation/564105496/ACTIVIDAD-11-INFOGRAFIA

Villao, J., & López, E. (2024). *Revisión de literatura sobre el uso de Machine Learning enfocados a la seguridad de la información para minimizar vulnerabilidades en el sector educativo* . Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/29275/1/UPS-GT005918.pdf

Villavicencio, W. (2023). *Estudio de técnicas de aprendizaje automático utilizadas en ciberseguridad*. Obtenido de http://dspace.ups.edu.ec/handle/123456789/26498

Zainal, A., & Ghaleb, F. (2022). *Malware Detection Issues, Challenges, and Future Directions: A Survey*. Obtenido de https://www.researchgate.net/publication/362958068_Malware_Detection_Issues_Challenges_and_Future_Directions_A_Survey

Zhong, R., & Salehi, C. (2023). *Machine learning for drilling applications: A review*. Obtenido de ScienceDirect: https://www.sciencedirect.com/topics/engineering/general-machine-learning