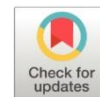


Modelo de aprendizaje automático para la detección de ciberataques en tráfico HTTP en entornos web

Machine learning model for detecting cyberattacks in HTTP traffic in web environments

- ¹ Jonathan Eduardo Mero Zambrano  <https://orcid.org/0009-0006-3663-5157>
Universidad Técnica de Manabí (UTM), Portoviejo, Ecuador.
Facultad de Ciencias Informáticas.
Ingeniería en Tecnologías de la Información
jmero4360@utm.ec
- ² Gabriel Eduardo Morejón López  <https://orcid.org/0000-0001-8902-4583>
Universidad Técnica de Manabí (UTM), Portoviejo, Ecuador
gabriel.morejon@utm.edu.ec
- ³ Wendy Viviana Obregón Martínez  <https://orcid.org/0000-0003-2402-6670>
Universidad Técnica de Manabí (UTM), Portoviejo, Ecuador.
wendy.obregon@utm.edu.ec



Artículo de Investigación Científica y Tecnológica

Enviado: 14/07/2026

Revisado: 13/08/2026

Aceptado: 11/09/2026

Publicado: 01/10/2026

DOI: <https://doi.org/10.33262/concienciadigital.v9i4.3767>

Cítese: Mero Zambrano, J. E., Morejón López, G. E., & Obregón Martínez, W. V. (2026). Modelo de aprendizaje automático para la detección de ciberataques en tráfico HTTP en entornos web. ConcienciaDigital, 9(4), 59-82. <https://doi.org/10.33262/concienciadigital.v9i4.3767>



CONCIENCIA DIGITAL, es una revista multidisciplinar, **trimestral**, que se publicará en soporte electrónico tiene como **misión** contribuir a la formación de profesionales competentes con visión humanística y crítica que sean capaces de exponer sus resultados investigativos y científicos en la misma medida que se promueva mediante su intervención cambios positivos en la sociedad. <https://concienciadigital.org>

La revista es editada por la Editorial Ciencia Digital (Editorial de prestigio registrada en la Cámara Ecuatoriana de Libro con No de Afiliación 663) www.celibro.org.ec

Esta revista está protegida bajo una licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 International. Copia de la licencia: <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>



Palabras claves:

Aprendizaje
automático;
ciberataques;
tráfico HTTP;
ciberseguridad;
Random Forest;
SVM;
XGBoost.

Keywords:

Machine learning;
cyberattacks;
HTTP traffic;
cybersecurity;
Random Forest;
SVM;
XGBoost.

Resumen

Introducción. El crecimiento acelerado de las aplicaciones web ha incrementado el tráfico HTTP y ha ocasionado que este protocolo se convierta en uno de los objetivos favoritos de quienes buscan vulnerar la seguridad de la información. **Objetivo.** Crear una solución tecnológica basada en un modelo de aprendizaje automático capaz de detectar ciberataques en tráfico HTTP. **Metodología.** La investigación fue aplicada y de enfoque mixto. Se trabajó con el HTTP DATASET CSIC 2010; se extrajeron ocho características cuantitativas y se evaluaron Random Forest, SVM y XGBoost mediante exactitud, precisión, recall y F1-score, con búsqueda en cuadrícula y validación cruzada estratificada de cinco particiones. **Resultados.** XGBoost obtuvo el mejor desempeño global, con exactitud de 91,1% y F1-score de 0,888, seguido por Random Forest, con exactitud de 90,7% y F1-score de 0,884. SVM obtuvo un recall de 0,627 y no detectó una gran cantidad de ataques. **Conclusión.** Random Forest y XGBoost mostraron capacidad para identificar patrones anómalos sin depender de firmas predefinidas y constituyen alternativas más seguras que SVM para un eventual sistema de detección en producción. **Área de estudio general:** Ingeniería y tecnología. **Área de estudio específica:** Ciberseguridad y aprendizaje automático. **Tipo de estudio:** Artículo original.

Abstract

Introduction. The accelerated growth of web applications has increased HTTP traffic and has made this protocol one of the main targets for those seeking to compromise information security. **Objective.** To create a technological solution based on a machine learning model capable of detecting cyberattacks in HTTP traffic. **Methodology.** This was an applied study with a mixed approach. The HTTP DATASET CSIC 2010 was used; eight quantitative features were extracted, and Random Forest, SVM, and XGBoost were evaluated through accuracy, precision, recall, and F1-score, using grid search and five-fold stratified cross-validation. **Results.** XGBoost achieved the best overall performance, with 91.1% accuracy and an F1-score of 0.888, followed by

Random Forest, with 90.7% accuracy and an F1-score of 0.884. SVM achieved a recall of 0.627 and failed to detect many attacks. **Conclusion.** Random Forest and XGBoost demonstrated the ability to identify anomalous patterns without relying on predefined signatures and constitute safer alternatives than SVM for a future production detection system. The findings support the viability of the proposed machine learning approach for strengthening cybersecurity in web environments. Future validation should use institutional HTTP traffic from the Technical University of Manabí and assess model performance under real operating conditions. **General area of study:** Engineering and technology. **Specific area of study:** Cybersecurity and machine learning. **Type of study:** Original article.

1. Introducción

El crecimiento acelerado de las aplicaciones web ha incrementado el tráfico HTTP lo que ha ocasionado que este protocolo se convierta en uno de los objetivos favoritos de ciertas personas que buscan vulnerar la seguridad de la información. Las aplicaciones web mantienen una buena parte de los trabajos académicos, trabajos administrativos y empresariales, los cuales están en exposición constante a internet por lo cual nos deja vulnerables para posibles ataques porque se aprovechan las debilidades presentes en las solicitudes y respuestas HTTP.

Durante bastante tiempo la seguridad de estos entornos ha dependido casi por completo de los mecanismos basados en firmas y reglas fijas. Esto genera un inconveniente: cuando aparece una nueva amenaza o una variante que altera sutilmente su estructura para evadir los controles, los sistemas tradicionales pierden eficacia. De ahí surge la necesidad de plantear enfoques adaptativos, que sean capaces de responder a un tráfico HTTP que no deja de transformarse.

Por lo puedo decir que, en este contexto, los modelos de aprendizaje automático representan una alternativa viable debido a su capacidad para analizar patrones presentes en grandes volúmenes de tráfico, para reconocer patrones complejos y los comportamientos anómalos que separan el tráfico legítimo del tráfico malicioso. A través de la información investigada se construye el presente trabajo, que será desarrollado bajo la modalidad de artículo académico-científico: que propone un modelo de detección de ciberataques que será sustentado en técnicas de aprendizaje

automático, con la finalidad de reforzar los mecanismos de seguridad informática en entornos web.

1.1. Planteamiento del problema

El crecimiento de las aplicaciones web y con el aumento incesante del tráfico HTTP podemos observar que se ha incrementado la superficie de exposición de los sistemas informáticos, favoreciendo la aparición de diferentes técnicas utilizadas para comprometer servicios digitales. La relación de vectores de ataque se ha vuelto tan avanzada, que los métodos tradicionales basados en seguridad no los detectan por sí mismos (Buczak & Guven, 2016; Sommer & Paxson, 2010), incluyendo la inyección SQL, el cross-site scripting (XSS), la denegación de servicio y la manipulación de solicitudes HTTP.

Los sistemas tradicionales siguen siendo útiles para detectar amenazas conocidas, pero presentan limitaciones ante ataques modificados. Por ello, este trabajo analiza modelos con capacidad de adaptación a partir de patrones extraídos del tráfico HTTP. En distintos escenarios, algoritmos como *Random Forest*, *Support Vector Machines* y varias arquitecturas de aprendizaje profundo han mostrado resultados favorables en la clasificación del tráfico HTTP legítimo y malicioso (Gniewkowski et al., 2021).

En Ecuador, específicamente en la Universidad Técnica de Manabí, se requiere crear un sistema automatizado e inteligente que analice el tráfico HTTP y detectar oportunamente los ataques cibernéticos. La institución sigue usando herramientas antiguas y realiza revisiones de forma manual no se dispone de herramientas capaces de analizar grandes volúmenes de tráfico HTTP o identificar anomalías en tiempo real. Lo que dificulta encontrar nuevas amenazas, además aumenta el riesgo en que la infraestructura tecnológica sufra daños y que los servicios en línea se paralicen o que se pierda información sensible.

A partir de lo anterior, el problema de investigación se concreta en la siguiente pregunta: ¿Cómo nos ayuda el uso de modelos de aprendizaje automático en el análisis del tráfico HTTP para mejorar la detección de ciberataques en la Universidad Técnica de Manabí?

Para orientar el desarrollo del estudio, se plantean las siguientes preguntas de investigación:

- 1.- ¿Qué tipo de atributos tiene el tráfico HTTP y cuáles son los tipos de ciberataques que afectan más a las aplicaciones web?
- 2.- ¿Cuáles son los algoritmos de aprendizaje automático más apropiados para identificar el tráfico HTTP malicioso?
- 3.- ¿Cuál sería la manera de aplicar un modelo de aprendizaje que identifique ciberataques utilizando información del tráfico HTTP?
- 4.- ¿Qué tan eficaz es el modelo sugerido en comparación con las técnicas de detección convencionales?

1.2. Justificación

La presente investigación busca identificar un modelo de aprendizaje automático que pueda detectar ciberataques en el tráfico HTTP, con el fin de Con el método detección fortalecer y clasificar automáticamente el tráfico malicioso fortaleciendo la seguridad de la información en entornos en línea para así ser capaz de identificar conductas y actividades anómalas que comprometen la privacidad y disponibilidad de los sistemas de información, abordar este problema nos resulta muy útil para la situación actual ya que los ciberataques contra aplicaciones web siguen en aumento, y los sistemas de detección tradicionales, basados en firmas y normativas éticas, presentan limitaciones que les impiden adaptarse rápidamente a los nuevos tipos de ataques. Lo cual tenemos que el aprendizaje automático nos, ofrece un enfoque más preciso y flexible el cual pueda detectar amenazas y así reducir los falsos positivos y reforzar la seguridad institucional.

El modelo que se propone se basa en el análisis del tráfico HTTP y la aplicación de algoritmos de aprendizaje automático seleccionados de la literatura científica especializada. Durante su desarrollo se incluye la recopilación y el preprocesamiento de los datos, el entrenamiento y la validación de los modelos, y la evaluación de su rendimiento mediante los métodos reconocidos en el campo de la ciberseguridad.

1.3. Antecedentes

Con el método detección de intrusiones de minería de datos y aprendizaje automático ha sido el objetivo de estudio de una manera constante desde hace más de una década. Buczak & Guven (2016) realizaron una revisión amplia de los métodos aplicados y llegaron a la conclusión de que los enfoques basados en datos superan a los sistemas de firmas estáticas cuando se trata de identificar amenazas que no habían sido documentadas antes.

Por otro lado, Sommer & Paxson (2010) pusieron más atención en los retos que enfrenta el aprendizaje automático al aplicarse a la detección de intrusiones en redes: también comentaron sobre la importancia de diseñar cuidadosamente las características y de validar los experimentos con rigor, ya que de lo contrario es fácil terminar con tasas altas de falsos positivos en entornos de producción.

Ring et al. (2019) compararon los principales conjuntos de datos usados en la detección de intrusiones basada en red, entre los cuales están KDD Cup 99 cuyas limitaciones documentaron, Tavallaee et al. (2009) señalaron sus fortalezas y debilidades que sirven de referencia para elegir data sets de tráfico HTTP en investigaciones posteriores.

Dentro del tráfico HTTP, Yu et al. (2018) propusieron DeepHTTP, un modelo basado en mecanismos de atención que trata las peticiones HTTP como secuencias de texto para detectar tráfico anómalo, y reportaron mejoras notables en la capacidad de identificar variaciones sutiles introducidas por los atacantes. Gniewkowski et al. (2021) siguieron una línea parecida con HTTP2vec, una técnica de representación vectorial de

solicitudes HTTP orientada a detectar tráfico anómalo, mientras que Ferriyan et al. (2022) exploraron el uso de Word2Vec para detectar tráfico malicioso cifrado.

Con estos aportes podemos identificar que los modelos basados en datos presentan ventajas frente a métodos estáticos, aunque requieren una adecuada selección de características para evitar errores de clasificación, en conjunto todos estos antecedentes nos muestran una tendencia clara hacia el uso de algoritmos de aprendizaje automático y aprendizaje profundo para superar las limitaciones de los sistemas tradicionales de detección de intrusiones, lo que respalda la pertinencia del presente trabajo de titulación.

1.4. Bases teóricas

Las bases teóricas de esta investigación permiten establecer los conceptos que sustentan el análisis del tráfico HTTP, los principales ciberataques dirigidos a aplicaciones web y la aplicación del aprendizaje automático para su detección. Estos fundamentos orientan la selección de las características del tráfico y de los algoritmos empleados en la fase experimental, así como la interpretación de las métricas utilizadas para valorar su desempeño.

1.4.1. El protocolo HTTP

El protocolo de transmisión de hipertexto es importante en la comunicación entre clientes y servidores en la web. Cada interacción consta de una solicitud del cliente y una respuesta del servidor, ambas construidas con un conjunto de componentes normalizados (Tanenbaum & Wetherall, 2012; Stallings, 2017).

Se analizaron los siguientes componentes fundamentales del protocolo HTTP.

Los componentes HTTP analizados fueron:

- Método HTTP: acción solicitada al servidor; GET, POST, PUT, DELETE
- URL: dirección del recurso solicitado; /login, /productos?id=10
- Headers HTTP: información adicional enviada entre cliente y servidor; User-Agent, Host, Accept
- Parámetros: datos enviados por el cliente; usuario=admin&clave=123
- Código de estado: respuesta emitida por el servidor; 200 OK, 404 Not Found

1.4.2. Ciberataques hacia aplicaciones web

Las aplicaciones web quedan expuestas a distintos vectores de ataque que aprovechan debilidades en el manejo de las solicitudes HTTP. Siguiendo el estándar OWASP Top 10 (OWASP Foundation, 2021), entre los ataques más relevantes para este estudio están los siguientes:

- Inyección SQL (SQLi): consiste en insertar código SQL malicioso en los parámetros de una solicitud para manipular la base de datos del servidor.
- Cross-Site Scripting (XSS): el XSS implica inyectar contenido ejecutable en una aplicación susceptible, buscando que el navegador del usuario ejecute comandos no autorizados.
- Inyección de comandos: ejecuta órdenes del sistema operativo de forma no autorizada, aprovechando áreas vulnerables de la aplicación.

- Recorrido de directorios: alguien entra a archivos y carpetas del servidor sin permiso al cambiar las rutas.
- Denegación de Servicio (DoS): el atacante llena el servidor con muchas peticiones en muy poco tiempo.
- Ataques de Fuerza Bruta: utilizan programas para probar miles de combinaciones de contraseñas hasta que logran entrar sin permiso.

Estudiar estos ataques ayuda a encontrar patrones en el tráfico maligno: el uso de caracteres especiales en los parámetros, el uso de URLs codificadas o cambiadas, el hecho de que se repitan muchas veces las solicitudes de autenticación y se modifiquen las cabeceras HTTP para que el atacante oculte su identidad. Dentro de este estudio analizar, este tipo de ataque resulta relevante debido a que sus patrones se reflejan en parámetros y contenido de solicitudes HTTP analizadas por los algoritmos.

1.4.3. Aprendizaje automático aplicado a la ciberseguridad

Dentro de la rama de la ciberseguridad el aprendizaje automático nos ayuda a construir modelos que sean capaces de reconocer los patrones maliciosos a partir de los datos y mejorar su desempeño sin que cada tarea deba ser programada explícitamente. En la rama de la ciberseguridad, estas técnicas se usan para clasificar el tráfico de red en comportamiento legítimo o malicioso (Buczak & Guven, 2016).

Entre los algoritmos de aprendizaje supervisado más citados en la literatura para detectar tráfico HTTP malicioso tenemos los siguientes:

- Los algoritmos de aprendizaje automático analizados fueron:
- *Random Forest*: algoritmo de ensamble basado en múltiples árboles de decisión, robusto ante datos mixtos y ruido; su implementación está ampliamente soportada en bibliotecas como scikit-learn (Scikit-learn Developers, 2024). Si se usa muchos árboles, la cantidad de memoria que se necesita serán mayor los tiempos para hacer las predicciones serán más largos.
- *Support Vector Machine* (SVM): busca hiperplanos óptimos que separen las clases en espacios con muchas dimensiones (Cortes & Vapnik, 1995). No se escala bien cuando hay grandes volúmenes de datos.
- *XGBoost*: optimiza árboles de decisión de forma secuencial usando *boosting* y tiene regularización integrada (Chen & Guestrin, 2016). Necesita ajuste manual de varios hiperparámetros.
- *Deep Learning* (MLP): redes neuronales que se multicapa y extraen características complejas sin que intervenga un humano (Goodfellow et al., 2016). Baja interpretabilidad y alta demanda de datos y de hardware especializado.

Al momento de comparar estos algoritmos según precisión, *recall*, *F1-score*, tiempo de entrenamiento e interpretabilidad, se determinó que *Random Forest*, SVM y *XGBoost* son los candidatos más adecuados para la fase experimental de este trabajo de titulación. Las arquitecturas de aprendizaje profundo quedaron descartadas por su baja interpretabilidad y su alto costo de infraestructura, dos factores poco compatibles con un

sistema de inspección ágil pensado para datos tabulares y cadenas estructuradas extraídas de las peticiones HTTP.

1.4.4. Métricas de evaluación del desempeño

Evaluar modelos de clasificación en tareas de ciberseguridad se apoya en métricas derivadas de la matriz de confusión, que contrasta las predicciones del modelo con los valores reales del conjunto de datos:

- Precisión: porcentaje de predicciones positivas que resultaron ser correctas y reduce los falsos positivos.
- Sensibilidad o *recall*: porcentaje de ataques verdaderos identificados correctamente; reduce los falsos negativos.
- *F1-score*: media armónica entre precisión y *recall*, muy útil sobre todo en conjuntos de datos desequilibrados, como como es común. Suceder entre tráfico auténtico y dañino.
- Exactitud (accuracy): relación total de aciertos sobre el total de observaciones analizadas.

Estas métricas se emplearon en la fase experimental del presente trabajo para comparar el desempeño de los algoritmos *Random Forest*, SVM y *XGBoost* frente a los métodos tradicionales de detección basados en firmas.

Los términos clave considerados en el estudio fueron:

- Tráfico HTTP: conjunto de solicitudes y respuestas intercambiadas entre un cliente y un servidor web mediante el protocolo HTTP.
- Ciberataque: acto intencionado destinado a poner en riesgo la confidencialidad, integridad o disponibilidad de un sistema computacional.
- Aprendizaje automático: conjunto de métodos que posibilitan a un sistema identificar patrones a partir de datos sin necesidad de ser programado de manera específica.
- Falso positivo: cuando recibe una solicitud de tráfico verdadero, el modelo lo detecta como un ataque.
- Falso negativo: es cuando el modelo no se da cuenta de un ataque real, dejando que el ataque llegue al servidor.
- Conjunto de datos: es un grupo de información que se utiliza para entrenar y probar un modelo de aprendizaje automático.
- Validación cruzada (*cross-validation*): es una manera de validar el modelo en el que se divide el conjunto de entrenamiento en varias partes, este método nos sirve para verificar cómo funciona el modelo y como trabaja con datos nuevos.

1.5. Marco conceptual

Este proyecto de titulación se sitúa dentro de la normativa ecuatoriana actual relacionada con la seguridad informática y la protección de datos, así como de las directrices institucionales de la Universidad Técnica de Manabí para los trabajos de titulación.

Ley Orgánica de Protección de Datos Personales del Ecuador, en lo referente al tratamiento responsable de la información al recolectar y usar conjuntos de datos de tráfico HTTP.

Regulación del régimen académico del Consejo de Educación Superior (CES), que establece los procedimientos de titulación mediante el enfoque de artículo académico-científico.

Normativa interna de la Universidad Técnica de Manabí y de la Facultad de Ciencias Informáticas para la elaboración y presentación de trabajos de titulación.

Guía NIST SP 800-94 - *Guide to Intrusion Detection and Prevention Systems*, utilizada como referencia técnica internacional para los sistemas de detección de intrusiones (Scarfone & Mell, 2007).

2. Metodología

Esta sección describe los materiales, técnicas y procedimientos empleados para implementar el modelo de detección de ciberataques en tráfico HTTP, correspondiente al desarrollo del trabajo de titulación. El diseño metodológico conserva el tipo de investigación aplicada y el enfoque mixto planteados en el anteproyecto, combinando el análisis cuantitativo del desempeño de los algoritmos mediante métricas estadísticas con la interpretación cualitativa de los resultados frente a los enfoques tradicionales de detección.

2.1. Tipo y enfoque de la investigación

La siguiente investigación es de tipo aplicada, el objetivo principal es crear una solución tecnológica, un modelo de aprendizaje automático que pueda detectar ciberataques en tráfico HTTP.

El enfoque de la investigación es mixto. la parte cuantitativa mide el funcionamiento de los modelos para lo cual se usan métricas estadísticas como la de precisión, *recall*, *F1-score* y la exactitud, mientras que la parte cualitativa podemos entender de una mejor manera los resultados y a compararlos con los mecanismos de detección basados en firmas.

2.2. Conjunto de datos

El desarrollo experimental se apoyó en el HTTP DATASET CSIC 2010 (Torrano-Giménez et al., 2013), elaborado por el Instituto de Seguridad de la Información del Consejo Superior de Investigaciones Científicas (CSIC, España). Este conjunto de datos contiene solicitudes HTTP reales generadas contra una aplicación web de comercio electrónico e incluye tanto tráfico legítimo como tráfico malicioso inyección SQL, desbordamiento de buffer, divulgación de archivos, inyección CRLF, Cross-Site Scripting (XSS), Server-Side Include y manipulación de parámetros, en línea con los vectores de ataque descritos en la sección 1.4.2.

Se trabajó con los archivos `normalTrafficTraining.txt` y `anomalousTrafficTest.txt`, que en conjunto suman 12.000 solicitudes HTTP distribuidas según la **Tabla 1**.

Tabla 1

Distribución de clases del conjunto de datos.

Clase de tráfico	N.º de instancias	Proporción
Legítimo (normal)	6000	50%
Malicioso (anómalo)	6000	50%
Total	12000	100%

2.3. Extracción de características

Para convertir las solicitudes HTTP en vectores numéricos que los algoritmos de aprendizaje automático pudieran procesar, se desarrolló un analizador (parser) en Python capaz de leer el formato crudo del dataset método, URL, cabeceras HTTP y cuerpo de la solicitud y extraer ocho características cuantitativas relevantes para detectar tráfico malicioso, como se muestra en la **Tabla 2**.

Tabla 2

Características extraídas de cada solicitud HTTP

Característica	Descripción
Longitud de la URL	Número de caracteres de la URL solicitada
Número de parámetros	Cantidad de parámetros enviados en la solicitud
N.º de caracteres especiales	Conteo de símbolos como <, >, ', ", %,;, ()
Palabras clave SQL	Presencia de términos como select, union, drop, or 1=1
Longitud de la solicitud	Tamaño total de la URL más el cuerpo de la petición
Entropía de la URL	Medida de aleatoriedad/ofuscación de la URL (Shannon)
Proporción de caracteres codificados	Porcentaje de secuencias %XX dentro de la URL
Método HTTP	Método de la solicitud (GET, POST, PUT, DELETE) codificado

2.4. Selección y configuración de los algoritmos

A partir del análisis comparativo realizado, se seleccionaron *Random Forest*, *Support Vector Machine* (SVM) y *XGBoost* como los algoritmos más adecuados para la fase experimental, por su equilibrio entre precisión, interpretabilidad y costo computacional. En la etapa correspondiente se implementaron y configuraron Random Forest y SVM; la incorporación de XGBoost y el ajuste fino de hiperparámetros mediante validación cruzada quedaron pendientes para el transcurso de la siguiente etapa. La configuración inicial empleada para cada algoritmo en esta fase se presenta en la **Tabla 3**.

Tabla 3

Configuración inicial de los algoritmos implementados.

Algoritmo	Hiperparámetros configurados
Random Forest	200 árboles (n_estimators = 200); profundidad máxima = 14
SVM	Núcleo RBF; parámetro de regularización C = 2.0; características estandarizadas

2.5. Procedimiento experimental

Dado el alto costo computacional de entrenar un modelo SVM con núcleo RBF sobre el conjunto completo de datos, se construyó una muestra estratificada balanceada de 12 000 solicitudes (6 000 legítimas y 6 000 maliciosas), respetando la proporción de clases mediante un muestreo independiente por categoría. La muestra se dividió en un 75% de datos para entrenamiento y un 25% de datos para pruebas, se aseguró que los dos subconjuntos tuvieran la misma distribución.

Para el modelo SVM, primero normalizamos las variables numéricas para que tuvieran media cero y varianza unitaria. Se tuvieron que normalizar las variables ya que el modelo es muy sensible a la escala de datos usamos el modelo *Random Forest*, no se necesitó normalización porque funciona con árboles. El entrenamiento sobre el conjunto de datos completo, la incorporación de *XGBoost* y el ajuste de hiperparámetros mediante validación cruzada se explican en la sección de resultados.

2.6. Herramientas y recursos tecnológicos

El desarrollo experimental se llevó a cabo con el lenguaje de programación Python 3.12 (Python Software Foundation, 2024) y la biblioteca de aprendizaje automático scikit-learn (Scikit-learn Developers, 2024), utilizada tanto para el preprocesamiento y la partición estratificada de los datos como para implementar, entrenar y validar los modelos *Random Forest* y SVM. Para *XGBoost* (sección de resultados) se empleó la biblioteca *xgboost* (Chen & Guestrin, 2016). Como recurso adicional se utilizó un computador personal con acceso a internet para obtener el conjunto de datos y consultar documentación técnica especializada, en concordancia con el diseño metodológico previsto en el anteproyecto.

2.7. Técnicas e instrumentos de evaluación

El desempeño de los modelos se evalúa con las métricas descritas en la sección 1.4.4 (precisión, recall, F1-score y exactitud), calculadas a partir de la matriz de confusión obtenida sobre el conjunto de prueba. En la sección de resultados se incorpora además la validación cruzada estratificada (5-fold) como instrumento para estimar la estabilidad y la capacidad de generalización de cada modelo. Comparar el desempeño de Random Forest, SVM y XGBoost, y contrastar los resultados frente a los mecanismos tradicionales de detección basados en firmas, en respuesta a la pregunta de investigación planteada en la sección 1.1.

2.8. Preparación y organización de los datos

La preparación de los datos comenzó con la lectura de las solicitudes contenidas en los archivos `normalTrafficTraining.txt` y `anomalousTrafficTest.txt` del HTTP DATASET CSIC 2010. Cada registro fue tratado como una solicitud HTTP compuesta por método, URL, cabeceras y cuerpo. El analizador desarrollado en Python transformó esa información en una representación numérica formada por ocho características: longitud de la URL, número de parámetros, número de caracteres especiales, presencia de palabras clave SQL, longitud total de la solicitud, entropía de la URL, proporción de caracteres codificados y método HTTP. Esta transformación permitió que los algoritmos seleccionados procesaran información originalmente expresada como texto y cadenas estructuradas.

Las características fueron elegidas porque reflejan comportamientos presentes en los ataques descritos en el estudio. Las solicitudes maliciosas pueden presentar modificaciones en la longitud, uso repetido de caracteres especiales, parámetros alterados, palabras vinculadas con consultas SQL, codificación de fragmentos de la URL y cambios en la estructura completa de la petición. Por esta razón, el modelo no dependió únicamente de una palabra o firma específica, sino del comportamiento conjunto de las variables extraídas de cada solicitud. El procedimiento mantuvo la diferenciación entre tráfico legítimo y malicioso disponible en el conjunto de datos.

2.9. Entrenamiento, prueba y validación

Para el entrenamiento inicial se construyó una muestra estratificada balanceada de 12 000 solicitudes, integrada por 6 000 registros legítimos y 6 000 maliciosos. El muestreo independiente por categoría permitió conservar una representación equivalente de las dos clases. La muestra se dividió en 75% para entrenamiento y 25% para prueba. Posteriormente, la evaluación final utilizó el conjunto completo de 61 065 solicitudes, compuesto por 36 000 solicitudes legítimas y 25 065 maliciosas. La partición estratificada produjo 45 798 registros de entrenamiento y 15 267 registros de prueba, manteniendo las proporciones de 58,9% de tráfico legítimo y 41,1% de tráfico malicioso.

El modelo *Random Forest* se entrenó con el conjunto completo de entrenamiento. En el caso de SVM con núcleo RBF, el ajuste se realizó sobre una muestra estratificada de 12 000 solicitudes debido a su mayor costo computacional. Una vez establecidos sus mejores hiperparámetros, el modelo se evaluó sobre las mismas 15 267 solicitudes de prueba utilizadas para los demás algoritmos. *XGBoost* se incorporó como tercer modelo y se entrenó con el conjunto completo. De este modo, la comparación final se realizó sobre un único conjunto de prueba y con las mismas métricas para los tres algoritmos.

Antes de entrenar SVM, las variables numéricas se estandarizaron para que tuvieran media cero y varianza unitaria, debido a la sensibilidad de este algoritmo frente a las diferencias de escala. *Random Forest* no necesitó esta normalización porque trabaja con árboles de decisión. El ajuste de los hiperparámetros se efectuó mediante *GridSearchCV*

y validación cruzada estratificada de cinco particiones. La métrica utilizada para optimizar la búsqueda fue *F1-score*, seleccionada por su capacidad de equilibrar precisión y *recall* frente al desbalance existente entre las clases.

2.10. Criterios de comparación de los modelos

La evaluación consideró exactitud, precisión, *recall* y *F1-score*. La exactitud permitió identificar la proporción total de predicciones correctas; la precisión mostró cuántas predicciones positivas correspondieron realmente a ataques; el *recall* indicó la proporción de solicitudes maliciosas identificadas; y el *F1-score* reunió precisión y *recall* en una sola medida. También se revisó el promedio y la desviación estándar del *F1-score* obtenidos mediante validación cruzada, el tiempo de ajuste y entrenamiento y la matriz de confusión de cada modelo. Las matrices permitieron diferenciar verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.

La comparación concedió especial importancia a los falsos negativos porque representan ataques que el modelo clasifica como tráfico legítimo. En un entorno web, este tipo de error puede permitir que una solicitud maliciosa llegue al servidor. Los falsos positivos también fueron considerados, debido a que pueden provocar el bloqueo de tráfico legítimo. El análisis conjunto de estas medidas permitió valorar no solo la exactitud general, sino también la seguridad, estabilidad y viabilidad computacional de *Random Forest*, *SVM* y *XGBoost*.

2.11. Criterios de inclusión, exclusión y consideraciones éticas

Se incluyeron las solicitudes HTTP válidas disponibles en los archivos *normalTrafficTraining.txt* y *anomalousTrafficTest.txt* del HTTP DATASET CSIC 2010 que pudieron ser transformadas mediante las ocho características definidas. Para la evaluación final se utilizaron los 61 065 registros descritos en el conjunto de datos. No se incorporaron registros externos ni se modificaron las etiquetas de tráfico legítimo y malicioso. El estudio no involucró participantes humanos ni recopiló datos personales, debido a que trabajó con un conjunto de datos público y generado en un entorno controlado. El procesamiento se limitó al objetivo académico de comparar los tres algoritmos seleccionados.

3. Resultados

Este capítulo presenta los resultados obtenidos, a la evaluación exhaustiva del modelo de detección de ciberataques en tráfico HTTP. A diferencia de la fase preliminar descrita en la sección de metodología que trabajó con una muestra estratificada de 12 000 solicitudes, en esta etapa se completó el entrenamiento sobre el conjunto de datos íntegro (61 065 solicitudes HTTP), se incorporó el algoritmo *XGBoost* como tercer candidato, se ajustaron los hiperparámetros mediante búsqueda en cuadrícula (*GridSearchCV*) con validación cruzada estratificada de 5 particiones (5-fold), y se realizó el análisis comparativo final entre los tres algoritmos, nos muestra que *XGBoost* es más fiable para esta tarea, ya que alcanza la mayor precisión y el mejor rendimiento.

3.1. Entrenamiento sobre el conjunto de datos completo

El conjunto de datos completo 36 000 solicitudes legítimas y 25 065 maliciosas, 61 065 en total se dividió mediante muestreo aleatorio estratificado en 75% para entrenamiento (45 798 solicitudes) y 25% para prueba (15 267 solicitudes), conservando en ambos subconjuntos la proporción original de clases (58,9% legítimo / 41,1% malicioso). En esta fase el modelo *Random Forest* se entrenó con la totalidad de las 45 798 solicitudes de entrenamiento, y no con una muestra reducida.

Para el algoritmo SVM, dado el elevado costo computacional del núcleo RBF frente al volumen completo de datos, el ajuste de hiperparámetros se realizó sobre una muestra estratificada de 12 000 solicitudes extraída del conjunto de entrenamiento. El modelo final, ya con los mejores hiperparámetros determinados, se entrenó sobre dicha muestra y se evaluó sobre el conjunto de prueba completo de 15 267 solicitudes, de manera que la comparación final entre algoritmos se realizara sobre el mismo conjunto de prueba.

3.2. Ajuste de hiperparámetros mediante validación cruzada

Se aplicó la técnica de búsqueda en cuadrícula (*GridSearchCV*) combinada con validación cruzada estratificada de 5 particiones (*StratifiedKfold*, $k = 5$) sobre el conjunto de entrenamiento, optimizando la métrica *F1-score* por su idoneidad frente al desbalance de clases. La **Tabla 4** resume el espacio de búsqueda explorado y los mejores hiperparámetros obtenidos para cada algoritmo.

Tabla 4

Espacio de búsqueda e hiperparámetros óptimos por algoritmo.

Algoritmo	Espacio de búsqueda	Mejores hiperparámetros
Random Forest	n_estimators: {150, 250}; max_depth: {14, 20}	n_estimators = 250; max_depth = 20
SVM	C: {1.0, 2.0, 5.0}; gamma: {scale, auto}; núcleo RBF	C = 5.0; gamma = auto
XGBoost	n_estimators: {150, 250}; max_depth: {4, 6}; learning_rate: {0.1, 0.2}	n_estimators = 250; max_depth = 6; learning_rate = 0.1

El ajuste y entrenamiento final de *Random Forest* tomó 63,6 segundos; el de SVM, 57,7 segundos (incluyendo la búsqueda sobre la muestra de 12 000 solicitudes y el reentrenamiento final); y el de *XGBoost*, 17,6 segundos, una diferencia que evidencia la eficiencia computacional de este último algoritmo incluso al operar sobre el conjunto de entrenamiento completo.

3.3. Resultados de Random Forest, SVM y XGBoost

Luego de encontrar los mejores valores para los hiperparámetros, se comparó como fue el funcionamiento de los tres modelos usando un conjunto de prueba de 15 267 solicitudes, en ese conjunto de prueba se incluyeron 9 000 solicitudes legítimas y 6 267

solicitudes maliciosas. Se muestran la exactitud, precisión, *recall* y el *F1-score* de cada modelo, también se muestra el *F1-score* promedio de la validación cruzada de cinco pliegues que se hizo en el conjunto de entrenamiento, los datos obtenidos **Tabla 5** vemos qué tan estable son los modelos.

Tabla 5

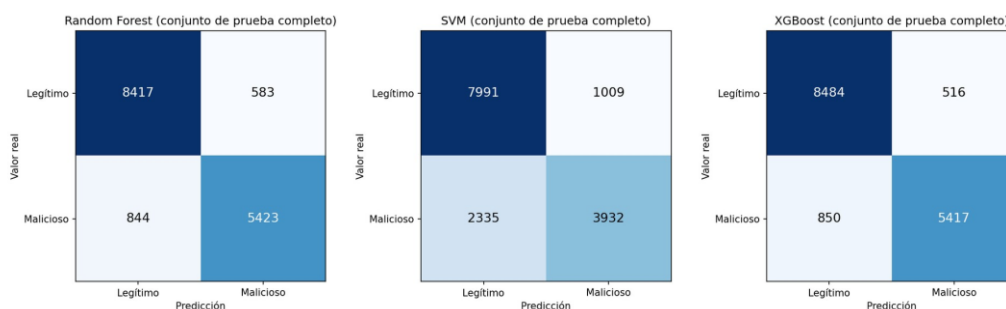
Métricas de desempeño obtenidas por Random Forest, SVM y XGBoost sobre el conjunto de prueba completo.

Algoritmo	Exactitud	Precisión	Recall	F1-score	F1 (CV 5-fold)
Random Forest	0,907	0,903	0,865	0,884	0,883 ± 0,003
SVM	0,781	0,796	0,627	0,702	0,701 ± 0,018
XGBoost	0,911	0,913	0,864	0,888	0,887 ± 0,003

La matriz de confusión complementa las métricas generales al mostrar la cantidad de aciertos y errores de cada modelo en las dos clases evaluadas, como se muestra en la **Figura 1**.

Figura 1

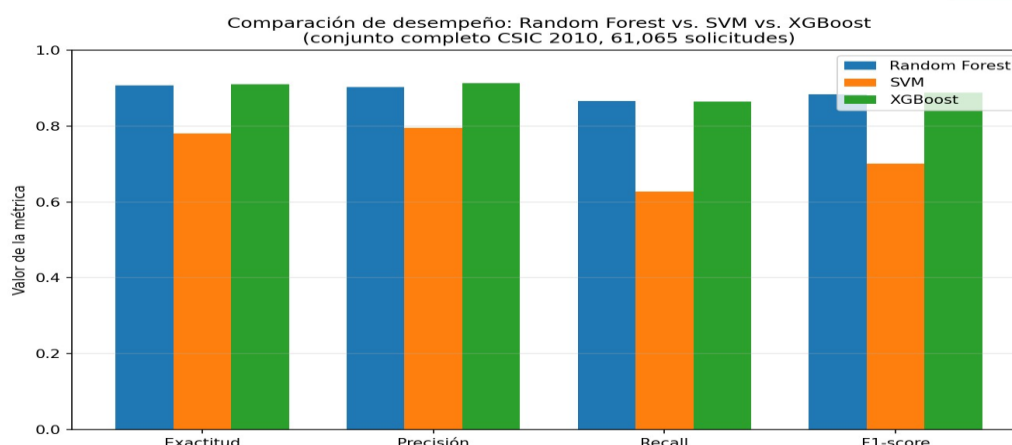
Matrices de confusión de Random Forest, SVM y XGBoost sobre el conjunto de prueba completo (15 267 solicitudes)



Como se muestra en la **Figura 2**, los resultados permiten comparar el desempeño de los tres modelos.

Figura 2

Comparación de desempeño entre Random Forest, SVM y XGBoost (conjunto completo CSIC 2010)



XGBoost como lo mencionan Chen & Guestrin (2016) obtuvo el mejor desempeño global (exactitud = 91,1%, *F1-score* = 0,888), seguido de cerca por *Random Forest* (exactitud = 90,7%, *F1-score* = 0,884). Los dos algoritmos de ensamble basados en árboles superaron a SVM, el *recall* de SVM fue 0,627 lo cual es bastante bajo, esto significa que no detectó una gran cantidad de ataques. Este resultado coincide con lo que se vio en el Capítulo II, donde se dice que SVM tiene una escalabilidad limitada frente a grandes cantidades de datos (Cortes & Vapnik, 1995), en cambio *Random Forest* y *XGBoost* son más fuertes ante conjuntos de datos extensos y ruidosos.

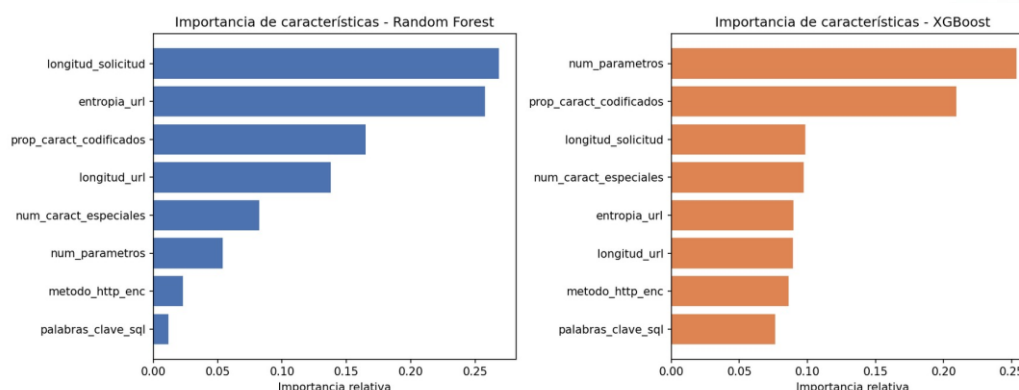
La desviación estándar de la validación cruzada respalda esta misma lectura: *Random Forest* y *XGBoost* obtuvieron $\pm 0,003$, frente a $\pm 0,018$ de SVM, lo que confirma que los modelos basados en árboles de decisión se comportan de forma más estable y consistente entre particiones de entrenamiento.

3.4. Análisis de importancia de características

Se analizó la importancia relativa de cada una de las ocho características extraídas (Tabla 5), para los modelos *Random Forest* y *XGBoost*, ya que ambos algoritmos permiten obtener esta medida de forma nativa. La Figura 3 presenta dicha comparación.

Figura 3

Importancia de las características en los modelos Random Forest y XGBoost (conjunto completo CSIC 2010)



En *Random Forest*, las características con mayor peso predictivo fueron la longitud de la solicitud (26,8%), la entropía de la URL (25,8%) y la proporción de caracteres codificados (16,5%), un resultado que coincide con lo obtenido en la fase preliminar. En *XGBoost*, por otro lado, las variables más influyentes fueron el número de parámetros (25,4%) y la proporción de caracteres codificados (20,9%), seguidas por la longitud de la solicitud (9,8%).

Esta discrepancia entre los algoritmos es comprensible al observar cómo cada uno forma sus árboles *bagging* en *Random Forest*, *boosting* secuencial en *XGBoost*, aunque ambos acuerdan en otorgar escasa importancia a la variable palabras clave sql por sí misma. Esto sugiere que los ataques presentes en el *dataset* CSIC 2010 se reconocen mejor a través de anomalías estructurales de la solicitud (longitud, entropía, codificación) que por la simple presencia de palabras clave SQL, las cuales aparecen ofuscadas o codificadas.

La evaluación realizada sobre el mismo conjunto de prueba permitió observar diferencias concretas entre los algoritmos. *XGBoost* presentó la exactitud y el *F1-score* más altos, mientras que *Random Forest* obtuvo valores muy cercanos y mantuvo un *recall* ligeramente superior. SVM mostró el rendimiento más bajo, especialmente en la identificación de solicitudes maliciosas. Estos datos confirmaron que los modelos basados en árboles ofrecieron el comportamiento más estable dentro de las condiciones experimentales descritas.

4. Discusión

Las matrices de confusión (**Figura 1**), identifican el tipo de error que predomina en cada modelo. En *Random Forest*, de las 6 267 solicitudes maliciosas del conjunto de prueba, 844 (13,5%) no fueron detectadas (falsos negativos), mientras que 583 solicitudes legítimas (6,5% de 9 000) se clasificaron por error como maliciosas (falsos positivos). En *XGBoost*, los falsos negativos ascendieron a 850 (13,6%) y los falsos

positivos a 516 (5,7%), un balance algo más favorable en cuanto a falsos positivos, lo que reduce el riesgo de bloquear tráfico legítimo. SVM, los falsos negativos subieron a 2 335 (37,3% de las solicitudes maliciosas), la principal debilidad de este modelo en el presente estudio.

Desde la perspectiva de la seguridad informática, un falso negativo (ataque no detectado) representa un riesgo más grave que un falso positivo (usuario legítimo bloqueado), ya que compromete directamente la integridad y disponibilidad del sistema protegido. Bajo este criterio, *Random Forest* y *XGBoost* con *recall* de 0,865 y 0,864 respectivamente constituyen alternativas notablemente más seguras que SVM (*recall* = 0,627) para un eventual sistema de detección en producción.

Vale la pena señalar una limitación del estudio: el conjunto de datos CSIC 2010, aunque real y ampliamente utilizado en la literatura (Ring et al., 2019; Torrano-Giménez et al., 2013), proviene de una única aplicación web de comercio electrónico generada en un entorno controlado. Su capacidad de generalizarse a otros entornos web institucionales como los de la Universidad Técnica de Manabí debería comprobarse en trabajos futuros mediante tráfico HTTP capturado directamente de la infraestructura institucional.

4.1. Contraste frente a enfoques tradicionales de detección

Frente a los mecanismos tradicionales de detección basados en firmas y reglas estáticas limitados ante variantes de ataques no documentadas previamente (Buczak & Guven, 2016); Sommer & Paxson, 2010), los tres modelos de aprendizaje automático evaluados muestran capacidad de identificar patrones anómalos a partir de características estructurales de la solicitud HTTP, sin depender de firmas predefinidas. *Random Forest* y *XGBoost*, en particular, alcanzaron una exactitud superior al 90% y un F1-score por encima de 0,88 sobre el conjunto de prueba completo, lo que confirma la viabilidad del enfoque de aprendizaje automático planteado en el Capítulo II y responde de manera afirmativa a la pregunta de investigación formulada en la sección 1.1: la aplicación de modelos de aprendizaje automático al análisis del tráfico HTTP mejora de manera significativa la capacidad de detección de ciberataques frente a los enfoques tradicionales.

4.2. Interpretación del desempeño comparativo

Los resultados muestran que una exactitud elevada no debe interpretarse de manera aislada. Aunque los tres modelos consiguieron clasificar una parte importante del tráfico, la diferencia principal apareció en la capacidad para reconocer solicitudes maliciosas. *XGBoost* alcanzó una exactitud de 91,1% y un *F1-score* de 0,888, mientras que *Random Forest* obtuvo 90,7% y 0,884, respectivamente. La cercanía entre estos resultados demuestra que ambos modelos fueron consistentes; sin embargo, *XGBoost* redujo los falsos positivos a 516 frente a los 583 registrados por *Random Forest*.

El *recall* aporta una lectura complementaria. *Random Forest* alcanzó 0,865 y *XGBoost* 0,864, valores prácticamente equivalentes. Esta diferencia mínima indica que los dos algoritmos detectaron una proporción semejante de ataques. En cambio, SVM obtuvo

un *recall* de 0,627 y dejó sin detectar 2 335 solicitudes maliciosas. Desde la perspectiva de la seguridad informática, este resultado limita su utilidad para un sistema institucional, porque un falso negativo representa una amenaza que puede continuar hacia la aplicación web sin ser advertida.

La validación cruzada también respalda la selección de los modelos basados en árboles. *Random Forest* y *XGBoost* presentaron una desviación estándar de $\pm 0,003$, mientras que SVM alcanzó $\pm 0,018$. La menor variación indica que el rendimiento de los dos primeros modelos se mantuvo estable entre las cinco particiones del conjunto de entrenamiento. Esta estabilidad es importante porque reduce la posibilidad de que el desempeño observado dependa únicamente de una partición favorable de los datos.

El tiempo computacional añade otro elemento a la comparación. *XGBoost* completó el ajuste y entrenamiento en 17,6 segundos, *Random Forest* necesitó 63,6 segundos y SVM empleó 57,7 segundos considerando la búsqueda sobre la muestra y el entrenamiento final. Dentro del experimento realizado, *XGBoost* combinó el menor tiempo con la mayor exactitud y el *F1-score* más alto. Este comportamiento respalda su selección como el modelo más fiable para la tarea estudiada, sin descartar a *Random Forest* como una alternativa cercana y con resultados estables.

4.3. Interpretación de las características y de los errores

El análisis de importancia mostró que los algoritmos no organizaron las variables de la misma manera. En *Random Forest*, la longitud de la solicitud, la entropía de la URL y la proporción de caracteres codificados concentraron el mayor peso predictivo. En *XGBoost*, las variables más influyentes fueron el número de parámetros y la proporción de caracteres codificados, seguidas por la longitud de la solicitud. A pesar de estas diferencias, ambos modelos otorgaron poca importancia individual a la presencia de palabras clave SQL.

Este resultado permite interpretar que el tráfico malicioso del conjunto CSIC 2010 se distingue mejor mediante cambios estructurales de la solicitud que por una palabra aislada. Las palabras clave pueden aparecer modificadas, ofuscadas o codificadas; por ello, depender únicamente de su presencia reproduciría parte de las limitaciones de los mecanismos basados en firmas. La combinación de longitud, entropía, parámetros, caracteres especiales y codificación ofrece una descripción más amplia del comportamiento de cada solicitud HTTP.

La principal limitación permanece vinculada con el origen del conjunto de datos. CSIC 2010 corresponde a una aplicación de comercio electrónico generada en un entorno controlado. Por ello, aunque los resultados permiten comparar los algoritmos bajo las mismas condiciones, no garantizan por sí solos el mismo rendimiento frente al tráfico institucional de la Universidad Técnica de Manabí. La validación posterior con solicitudes obtenidas en ese entorno permitirá determinar si las características seleccionadas conservan su capacidad de discriminación ante condiciones operativas reales.

Esta comprobación permitiría valorar la respuesta del modelo frente a variaciones propias de los servicios web y del tráfico institucional analizado.

5. Conclusiones

- *Random Forest* y *XGBoost*, con *recall* de 0,865 y 0,864 respectivamente, constituyen alternativas notablemente más seguras que SVM (*recall* = 0,627) para un eventual sistema de detección en producción.
- El análisis confirmó que los mecanismos tradicionales conservan utilidad frente a amenazas conocidas, pero presentan limitaciones cuando las solicitudes maliciosas cambian su estructura. Los modelos evaluados permitieron reconocer patrones del tráfico HTTP a partir de variables cuantitativas y ofrecieron una capacidad de adaptación superior a la dependencia exclusiva de firmas estáticas.
- La detección basada en aprendizaje automático puede complementar los controles existentes al clasificar automáticamente solicitudes legítimas y maliciosas. Su valor principal se encuentra en el análisis simultáneo de varias características de la petición, lo que permite identificar comportamientos anómalos incluso cuando no aparece una firma conocida de forma literal.
- La comparación de los tres algoritmos permitió establecer que *XGBoost* alcanzó el mejor desempeño general, con una exactitud de 91,1% y un *F1-score* de 0,888. *Random Forest* obtuvo resultados cercanos, con una exactitud de 90,7% y un *F1-score* de 0,884. Ambos modelos presentaron una desviación estándar de $\pm 0,003$ durante la validación cruzada, lo que evidencia un comportamiento estable entre las particiones del conjunto de entrenamiento.
- SVM presentó la principal limitación del estudio al alcanzar un *recall* de 0,627 y registrar 2 335 falsos negativos. En una tarea de detección de ciberataques, esta cantidad de solicitudes maliciosas no identificadas representa un riesgo mayor que el bloqueo incorrecto de tráfico legítimo. En consecuencia, los resultados no respaldan a SVM como la primera opción para un sistema de detección basado en las características y el conjunto de datos empleados.
- Las variables de mayor influencia confirmaron que los ataques pueden reconocerse mediante anomalías estructurales de las solicitudes HTTP. La longitud de la solicitud, la entropía de la URL, el número de parámetros y la proporción de caracteres codificados tuvieron mayor participación que la presencia aislada de palabras clave SQL. El aporte de la investigación consiste en mostrar que la combinación de estas características con modelos basados en árboles permite detectar tráfico malicioso sin depender exclusivamente de firmas estáticas.
- Los objetivos planteados fueron atendidos mediante la identificación de características del tráfico HTTP, la selección y comparación de *Random Forest*, SVM y *XGBoost*, la implementación del proceso de entrenamiento y la

evaluación con métricas reconocidas. Los resultados sostienen la viabilidad del aprendizaje automático como apoyo para reforzar la detección de ciberataques en entornos web. La aplicación institucional requiere, como etapa posterior, validar el modelo con tráfico real de la Universidad Técnica de Manabí e implementar un prototipo de inspección en tiempo real.

- La elección final debe considerar tanto el desempeño estadístico como las condiciones técnicas de implementación. En este estudio, *XGBoost* reunió la mayor exactitud, el *F1-score* más alto y el menor tiempo de ajuste y entrenamiento, mientras que *Random Forest* mantuvo un comportamiento muy cercano y un *recall* ligeramente superior. Por esta razón, los dos modelos pueden continuar hacia una fase de validación institucional. En dicha fase será necesario observar el comportamiento frente a variaciones del tráfico, medir el consumo de recursos y analizar los errores de clasificación antes de adoptar una decisión de despliegue. Esta comprobación permitirá determinar si los resultados obtenidos con el conjunto CSIC 2010 se conservan en las aplicaciones web y condiciones operativas de la institución.

5.1. Agradecimientos

El autor expresa su agradecimiento al Ing. Gabriel Morejón, Mg., tutor del trabajo de titulación, y a la Ing. Wendy Obregón, cotutora, por su orientación y acompañamiento académico durante el desarrollo de la investigación.

6. Conflicto de intereses

El autor declara que no existe conflicto de intereses en relación con el artículo presentado.

7. Derechos de autor (copyright)

El autor conserva los derechos de autor y garantiza a la Revista Conciencia Digital el derecho de primera publicación de este artículo.

8. Declaración de contribución de los autores

Los autores declaran haber participado de manera equitativa y sustancial en la realización de esta investigación, bajo las responsabilidades según la [Taxonomía CRediT](#) para describir las contribuciones individuales de cada autor al trabajo.

Jonathan Eduardo Mero Zambrano realizó la conceptualización de la investigación, curación y tratamiento de datos, análisis formal e investigación, metodología, recursos y software, validación y visualización, administración del proyecto y adquisición de fondos, escritura del borrador, revisión, edición y discusión de resultado, revisión del análisis estadístico, interpretación de los resultados, revisión crítica del contenido y aprobación de la versión final del manuscrito.

Gabriel Eduardo Morejón López & Wendy Viviana Obregón Martínez, realizaron la supervisión académica y metodológica de la investigación, brindó orientación durante el desarrollo del ensayo, revisó el análisis y la interpretación de los resultados, efectuó la revisión crítica del manuscrito, realizó la corrección científica y académica y aprobó la versión final para su publicación.

Divulgación de la delegación a la IA generativa

Durante la revisión y adaptación formal del artículo se utilizó inteligencia artificial generativa como apoyo para la organización, el ajuste de la redacción y la adecuación a la plantilla de la revista. Esta herramienta no fue utilizada para crear datos, realizar el análisis ni generar los resultados de la investigación. El autor revisó el contenido final y asume la responsabilidad sobre su integridad y exactitud. Declaración presentada por Jonathan Eduardo Mero Zambrano.

9. Costos de financiamiento

La investigación fue financiada con recursos propios del autor y no recibió financiamiento externo.

10. Referencias bibliográficas

- Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys and Tutorials*, 18(2), 1153-1176. <https://doi.org/10.1109/COMST.2015.2494502>
- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297. <https://doi.org/10.1007/bf00994018>
- Ferriyan, A., Thamrin, A. H., Takeda, K., & Murai, J. (2022). Encrypted malicious traffic detection based on Word2Vec. *Electronics*, 11(5), 679. <https://doi.org/10.3390/electronics11050679>
- Gniewkowski, M., Maciejewski, H., Surmacz, T. R., & Walentynowicz, W. (2021). HTTP2vec: Embedding of HTTP requests for detection of anomalous traffic. *arXiv*. <https://arxiv.org/abs/2108.01763>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. <https://books.google.com.ec/books?id=omivDQAAQBAJ>
- OWASP Foundation. (2021). *OWASP Top 10 Web Application Security Risks*. OWASP. <https://owasp.org/www-project-top-ten/>
- Python Software Foundation. (2024). *Python 3.12 documentation*. Python. <https://docs.python.org/3.12/>

- Ring, M., Wunderlich, S., Grudl, D., Landes, D., & Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computer Security*, 86, 147-167. <https://doi.org/10.1016/j.cose.2019.06.005>
- Scarfone, K. A., & Mell, P. M. (2007). *Guide to intrusion detection and prevention systems (IDPS)*. National Institute of Standards and Technology. <https://csrc.nist.gov/pubs/sp/800/94/final>
- Scikit-learn Developers. (2024). *Ensembles: gradient boosting, random forests, bagging, voting, stacking*. User guide – Scikit learn. <https://scikit-learn.org/stable/modules/ensemble.html>
- Sommer, R., & Paxson, V. (2010). *Outside the closed world: on using machine learning for network intrusion detection* [Proceedings of the IEEE Symposium on Security and Privacy, 305-316]. <https://doi.org/10.1109/SP.2010.25>
- Stallings, W. (2017). *Network Security Essentials: applications and Standards* (6th ed). Pearson. <https://www.pearson.com/en-us/subject-catalog/p/network-security-essentials-applications-and-standards/P200000003333/9780134527338>
- Tanenbaum, A. S. & Wetherall, D. J. (2012). *Redes de computadoras 5e*. Pearson Education. https://www.pearsonenespanol.com/mexico/educacion-superior/tanenbaum_index
- Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). *A detailed analysis of the KDD CUP 99 data set* [Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, 1-6]. <https://doi.org/10.1109/CISDA.2009.5356528>
- Torrano-Giménez, C., Pérez-Villegas, A., & Álvarez Maraón, G. (2013). *HTTP dataset CSIC 2010*. Information Security Institute, Spanish National Research Council (CSIC). https://petescully.co.uk/wp-content/uploads/2018/04/http_dataset_csic_2010.pdf
- Yu, Y., Yan, H., Guan, H., & Zhou, H. (2018). DeepHTTP: Semantics-structure model with attention for anomalous HTTP traffic detection and pattern mining. *arXiv*. <https://doi.org/10.48550/arXiv.1810.12751>

El artículo que se publica es de exclusiva responsabilidad de los autores y no necesariamente reflejan el pensamiento de la **Revista Conciencia Digital**.



El artículo queda en propiedad de la revista y, por tanto, su publicación parcial y/o total en otro medio tiene que ser autorizado por el director de la **Revista Conciencia Digital**.

