Modelo de inteligencia artificial para mejorar la detección de ataques de fuerza bruta en logs en Servidor Linux. 2025

Jose Dario Menendez Acosta

Facultad de Ingeniería Eléctrica y Electrónica, Universidad Nacional de Ingeniería

CBS06 M Inteligencia Artificial II

Ing. Yury Oscar Tello

16 de Octubre de 2025

Resumen

El avance tecnológico en áreas como la computación en la nube, redes vehiculares e Internet de las Cosas (IoT) ha incrementado el volumen de datos transmitidos, facilitando ciberataques que comprometen la seguridad de sistemas distribuidos. En este contexto, los ataques de fuerza bruta representan una amenaza persistente, especialmente en servidores Linux, donde intentos repetidos de acceso no autorizado generan logs que pueden analizarse para detectar intrusiones. Este estudio propone un modelo de inteligencia artificial (IA) basado en machine learning (ML) para mejorar la detección de estos ataques en logs de autenticación, superando limitaciones de métodos tradicionales como los Sistemas de Detección de Intrusiones basados en firmas (SIDS) y anomalías (A-IDS), que fallan ante ataques nuevos o generan falsos positivos.

La revisión de literatura destaca el uso de técnicas avanzadas de ML y deep learning (DL), como XGBoost, RNNs y modelos híbridos con lógica difusa, que logran precisiones superiores al 99% en entornos IoT. La metodología involucra el preprocesamiento de logs de btmp y auth.log, ingeniería de características (e.g., frecuencia de intentos por IP, tasa de éxito histórica) y balanceo de clases con SMOTE. Se emplea XGBoost para clasificación supervisada e IA explicable (XAI) con SHAP para interpretar el modelo.

Los resultados muestran una exactitud global de 1.00 en un conjunto de prueba con 3,756 eventos, con precisión y recall perfectos para ataques y 0.88 para eventos normales. La feature engineering, particularmente la reputación IP, fue clave para el rendimiento, demostrando eficiencia computacional (entrenamiento en 0.05 segundos). Este modelo ofrece aplicaciones en SIEM e IPS para detección en tiempo real y análisis forense, enfatizando que la calidad de datos supera ajustes algorítmicos en ciberseguridad.

Palabras clave

Detección de intrusiones, Machine learning, XGBoost, IA explicable

1. Introducción

El avance acelerado en tecnologías como la computación en la nube, los sistemas de redes vehiculares y el Internet de las Cosas (IoT), ha resultado en un incremento masivo en la cantidad de información transmitida a través de las infraestructuras de comunicación. Este crecimiento ha sido aprovechado por los atacantes, quienes han intensificado sus esfuerzos para comprometer la seguridad de los sistemas de red (Kasongo, 2023).

La ciberseguridad es un componente crítico de las redes distribuidas, enfrentando intentos constantes de acceso no autorizado para robar, alterar, exponer o destruir datos (Sarantos et al., 2025). Las intrusiones en la red se consideran una de las amenazas de seguridad más peligrosas para las organizaciones en la actualidad (Kasongo, 2023). Dado que las soluciones de seguridad actuales a menudo proporcionan solo protección a corto plazo, la expansión continua de las tecnologías hace que mantener la ciberseguridad sea un desafío perpetuo (Sarantos et al., 2025). En este panorama, los Sistemas de Detección de Intrusiones (IDS) son implementados para asegurar los sistemas y redes informáticas, monitoreando el tráfico en busca de amenazas (Kasongo, 2023).

Una de las amenazas más persistentes y comunes es el ataque de fuerza bruta, que puede causar daños graves, especialmente en redes IoT (Otoom et al., 2023). Un ataque de fuerza bruta consiste en el envío de todos los valores posibles para contraseñas o cuentas en un intento de obtener acceso no autorizado a la información del sistema (Park et al., 2021). En el contexto de la infraestructura de TI, la mayoría de los logs generados por accesos no autorizados son causados por ataques de fuerza bruta de SSH (Secure Shell), los cuales pueden ser detectados a través del análisis de dichos registros (Park et al., 2021).

La motivación principal para emplear inteligencia artificial (IA) reside en la necesidad de superar las limitaciones de los métodos tradicionales de detección (Sarantos et al., 2025). Los IDS modernos ya aprovechan los modelos de aprendizaje automático (Machine Learning, ML) para clasificar la actividad de la red como benigna o maliciosa (Sarantos et al., 2025). Los enfoques

basados en ML son esenciales para construir IDS inteligentes capaces de detectar intrusiones de fuerza bruta de manera automatizada y tomar medidas directas (Otoom et al., 2023).

Es crucial el uso de modelos avanzados, como los de aprendizaje profundo (Deep Learning, DL), una rama de ML inspirada en el funcionamiento de las neuronas del cerebro humano (Kasongo, 2023), ya que las técnicas convencionales se ven obstaculizadas de varias maneras (Sarantos et al., 2025):

a) Los IDS basados en firmas (Signature-based IDS - SIDS) dependen de patrones predefinidos y no pueden detectar ataques nuevos o de día cero.
b) Los sistemas basados en anomalías (Anomaly-based IDS - A-IDS) tienden a disparar demasiadas alarmas de falsos positivos, lo que los hace inconvenientes para los profesionales de seguridad.

Las técnicas de ML y DL se aplican para defenderse de los ataques de red mediante el aprendizaje de patrones de uso y registros de acceso (Park et al., 2021). Además, modelos híbridos que incorporan la lógica difusa (fuzzy logic) y ML han demostrado la capacidad de mejorar significativamente la precisión de detección (Mehmmod et al., 2025). Estos modelos pueden ofrecer una tasa de detección notable del 99% con considerablemente menos falsos positivos en comparación con los modelos tradicionales.

Por lo tanto, la implementación de un modelo de inteligencia artificial (IA) se propone como una solución fundamental y adaptable, diseñada para mejorar la precisión y la capacidad de generalización en la detección de ataques de fuerza bruta en Servidores Linux, superando las deficiencias observadas en los métodos existentes.

2. Revisión de la literatura

Para asegurar los sistemas y redes informáticas, se implementan los Sistemas de Detección de Intrusiones (IDS) (Kasongo, 2023; Sarantos et al., 2025). Un IDS es un artefacto (hardware o software) que monitorea el tráfico de una red en busca de amenazas (Kasongo, 2023; Sarantos et al., 2025). Los IDS se clasifican típicamente en tres categorías principales basadas en la técnica de

detección que emplean: IDS basados en firmas (SIDS), IDS basados en anomalías (A-IDS) e IDS híbridos (Kasongo, 2023). Los SIDS, también conocidos como Detección Basada en Conocimiento o Detección de Mal Uso, detectan ataques basándose en patrones preexistentes (firmas) almacenados en una base de datos (Sarantos et al., 2025). La principal limitación de los SIDS es su incapacidad para detectar ataques nuevos o de día cero (Sarantos et al., 2025), ya que dependen de firmas predefinidas (Sarantos et al., 2025). En contraste, los A-IDS escanean la red para identificar patrones o comportamientos que se desvían de lo que se considera normal (Kasongo, 2023; Sarantos et al., 2025), un enfoque que intenta mitigar la desventaja de los SIDS (Sarantos et al., 2025). Sin embargo, los A-IDS tienden a disparar un alto número de alarmas de falsos positivos (Sarantos et al., 2025), lo que los hace ruidosos e inconvenientes para los profesionales de seguridad (Sarantos et al., 2025).

2.1. El Problema de la Fuerza Bruta en Servidores y Entornos IoT

En el contexto de la infraestructura de TI, la mayoría de los logs generados por intentos de acceso no autorizado son causados por ataques de fuerza bruta SSH (Secure Shell) (Park et al., 2021). SSH es un protocolo fundamental para el acceso remoto seguro (Park et al., 2021). Sin embargo, la permisión de solicitudes de acceso remoto desde cualquier origen puede exponer un sistema a amenazas (Park et al., 2021). Los ataques de fuerza bruta SSH generan logs que tienen el mismo patrón de mensaje, incluyendo frases como "SSH user failed to log in from [attacker’s IP address] on VTY0 due to IP restriction" (Park et al., 2021). Al analizar estos registros, es posible recolectar la dirección IP del atacante, que luego se utiliza para controlar el acceso, por ejemplo, mediante listas negras (blacklists) (Park et al., 2021).

2.2. Aprendizaje Automático (ML) y Aprendizaje Profundo (DL) para IDS

Los IDS modernos aprovechan los modelos de Aprendizaje Automático (ML) (Sarantos et al., 2025), que otorgan a los programas informáticos la capacidad de aprender de la experiencia (Kasongo, 2023), para clasificar la actividad de la red como benigna o maliciosa (Sarantos et al., 2025). Los enfoques basados en ML son necesarios para construir IDS inteligentes capaces de

detectar intrusiones de fuerza bruta de manera automatizada (Otoom et al., 2023). Las técnicas de ML se aplican para defenderse de ataques de red mediante el aprendizaje de patrones de uso y registros de acceso (Park et al., 2021).

Las RNNs, y sus variantes modificadas como la Memoria a Largo Corto Plazo (LSTM) y la Unidad Recurrente Cerrada (GRU) (Kasongo, 2023), son cruciales porque tienen la capacidad de memorizar temporalmente estados previos y aplicarlos en el cálculo actual (Kasongo, 2023). El uso de algoritmos de selección de características, como el basado en XGBoost (Extreme Gradient Boosting) (Kasongo, 2023), en conjunto con RNNs, mejora el rendimiento de los IDS (Kasongo, 2023) al reducir la dimensión del espacio de características y la complejidad del modelo (Kasongo, 2023).

2.3. Desafíos de Generalización y Soluciones Avanzadas

Los IDS basados en Aprendizaje Supervisado (SL) son entrenados con conjuntos de datos etiquetados (Kasongo, 2023). Una limitación crítica de estos modelos es que su rendimiento se degrada significativamente cuando se implementan en un entorno de red diferente al que generó sus datos de entrenamiento (Sarantos et al., 2025). Este es el problema de la generalización de dominio (Sarantos et al., 2025). La construcción de conjuntos de datos etiquetados para cada dominio es intensiva en recursos y tiempo (Sarantos et al., 2025). El etiquetado de registros de datos es una tarea costosa que requiere el trabajo meticuloso de anotadores expertos (Sarantos et al., 2025).

3. Metodología

3.1. Descripción y Preprocesamiento del Dataset

El conjunto de datos para este estudio fue construido a partir de logs de autenticación de un servidor Linux, provenientes de dos fuentes: los archivos btmp y auth.log. Estos registros constituyen una serie temporal de eventos de acceso al sistema. El tratamiento de los datos crudos para la construcción de un dataset analítico implicó un riguroso proceso de preprocesamiento, detallado a continuación.

3.1.1. Consolidación y Normalización de Fuentes de Datos: Inicialmente, se realizó una consolidación de todos los archivos de log disponibles. Los registros históricos, almacenados en formato comprimido (.gz), fueron descomprimidos mediante programación para asegurar la inclusión del conjunto de datos completo en el análisis.

3.1.2. Parsing y Extracción Estructurada: Se emplearon expresiones regulares (regex) para el parseo de las líneas de log, transformando el texto no estructurado en datos tabulares. Este proceso se adaptó a la sintaxis de cada fuente:

a) Logs btmp: Se extrajeron exclusivamente los intentos de inicio de sesión fallidos, aislando la tupla de información compuesta por: usuario, dirección IP de origen y marca de tiempo.

b) Logs auth.log: Se aplicó un filtrado para aislar únicamente los eventos de autenticación de los servicios sshd (accesos remotos) y login (accesos locales). Para cada evento relevante, se extrajo la marca de tiempo, el estado del intento (Accepted/Failed), el usuario y la dirección IP. A los accesos locales se les asignó un identificador de IP estandarizado (local_login).

3.1.3. Estandarización Temporal y Etiquetado de Clases: Para garantizar la integridad cronológica, todas las marcas de tiempo extraídas fueron normalizadas y convertidas a la zona horaria Coordinated Universal Time (UTC). Posteriormente, se procedió con el etiquetado de los datos para la tarea de clasificación supervisada. Se generó una variable binaria, is_attack, asignando el valor 1 a los eventos de autenticación fallidos y 0 a los exitosos. El dataset resultante exhibió un severo desbalance de clases, característico de este tipo de problema.

3.1.4. Ingeniería de Características (Feature Engineering): Con el objetivo de enriquecer el dataset y proveer al modelo de información contextual y de comportamiento, se diseñó y calculó un conjunto de ocho características a partir de los datos base. Estas se pueden categorizar de la siguiente manera:

a) Características de Frecuencia y Variedad:

   i. ip_attempts_5min: Frecuencia de intentos desde una misma IP en una ventana temporal de 5 minutos.

ii. ip_distinct_users_15min: Conteo de nombres de usuario únicos intentados por una misma IP en una ventana de 15 minutos.

iii. user_attempts_15min: Frecuencia de intentos para un mismo nombre de usuario en 15 minutos.

b) Características de Comportamiento Temporal:

i. time_since_last_attempt_ip: Diferencia temporal, en segundos, con respecto al evento anterior originado desde la misma IP.

ii. hour_of_day: Hora del día en formato de 24 horas.

iii. is_weekend: Indicador booleano para eventos ocurridos en sábado o domingo.

c) Características Contextuales y de Reputación:

i. is_common_user: Indicador booleano que se activa si el nombre de usuario pertenece a una lista predefinida de objetivos de ataque comunes (e.g., root, admin).

ii. ip_success_rate: Tasa de éxito histórica de la dirección IP, calculada como un valor acumulativo en el tiempo, funcionando como una métrica de reputación dinámica.

3.2. Técnicas de Inteligencia Artificial Empleadas

La estrategia experimental se fundamentó en la aplicación de técnicas de Machine Learning (ML) Supervisado y IA Explicable (XAI).

3.2.1. Modelo de Machine Learning:

Algoritmo: Se seleccionó el algoritmo XGBoost (Extreme Gradient Boosting), perteneciente a la familia de métodos de ensamblaje basados en árboles de decisión. La elección se justifica por su documentada eficacia en datos tabulares, su robustez frente al sobreajuste y su alta eficiencia computacional.

Tratamiento del Desbalance de Clases: Para mitigar el sesgo inherente al desbalance de clases del dataset, se aplicó la técnica SMOTE (Synthetic Minority Over-sampling Technique). Dicha técnica se utilizó exclusivamente sobre el conjunto de entrenamiento para generar instancias sintéticas de la clase minoritaria ("eventos normales"), balanceando así la distribución de clases y permitiendo un aprendizaje más equitativo por parte del modelo.

Evaluación y Métrica de Rendimiento: El conjunto de datos fue dividido en subconjuntos de entrenamiento (80%) y prueba (20%). El rendimiento del modelo final se evaluó sobre el conjunto de prueba utilizando métricas estándar de clasificación: Precisión (Precision), Sensibilidad (Recall), Puntuación F1 (F1-Score) y la Matriz de Confusión. Adicionalmente, se midió la eficiencia computacional registrando los tiempos de entrenamiento e inferencia.

3.2.2. IA Explicable (XAI):

Framework de Explicabilidad: Para interpretar la "caja negra" del modelo XGBoost, se utilizó el framework SHAP (SHapley Additive exPlanations). SHAP es un método basado en la teoría de juegos que calcula la contribución marginal de cada característica a la predicción de una instancia específica, conocidos como valores Shapley. Esto permite una comprensión tanto global como local del comportamiento del modelo. Se generaron visualizaciones, incluyendo gráficos de resumen (summary_plot) y de dependencia (force_plot), para determinar la importancia relativa de las características y analizar la lógica subyacente a predicciones individuales.

3.3. Configuración del Entorno de Experimentación

El desarrollo experimental se llevó a cabo en un entorno de Jupyter Notebook, utilizando el lenguaje de programación Python. Las principales librerías y dependencias de software empleadas fueron:

a) Manipulación de Datos y Computación Numérica: pandas, numpy.
b) Procesamiento de Archivos y Texto: glob, re, gzip, shutil.
c) Visualización de Datos: matplotlib, seaborn.

d) Machine Learning y Modelado: scikit-learn (para la división de datos y métricas), xgboost (para la implementación del clasificador).
e) Balanceo de Datos: imbalanced-learn (para la implementación de SMOTE).
f) IA Explicable: shap.
g) Utilidades Adicionales: tqdm (para la monitorización de progreso en procesos computacionalmente intensivos).

4. Resultados y discusión

4.1. Métricas de Evaluación del Modelo Final

El modelo final, entrenado con un conjunto de 8 características y datos balanceados mediante la técnica SMOTE, demostró un rendimiento excepcional en la tarea de clasificación de eventos de autenticación. La evaluación se realizó sobre un conjunto de prueba compuesto por 3,756 eventos, de los cuales 8 eran legítimos ("Normal") y 3,748 correspondían a ataques.

Las métricas de rendimiento obtenidas fueron las siguientes:

a) Accuracy (Exactitud): El modelo alcanzó una exactitud global de 1.00, indicando que clasificó correctamente casi la totalidad de los eventos en el conjunto de prueba.
b) Métricas por Clase:

    i. Ataque (Clase 1): Se obtuvo una Precisión de 1.00 y un Recall de 1.00. Esto significa que el modelo no solo fue perfecto al emitir una alerta (nunca generó una falsa alarma), sino que también fue capaz de identificar todos los ataques reales sin que se le escapara ninguno.
    ii. Normal (Clase 0): Se alcanzó una Precisión de 0.88 y un Recall de 0.88. Este es un resultado sobresaliente para la clase minoritaria, demostrando que el modelo pudo identificar correctamente 7 de los 8 eventos legítimos, con un solo error de clasificación.

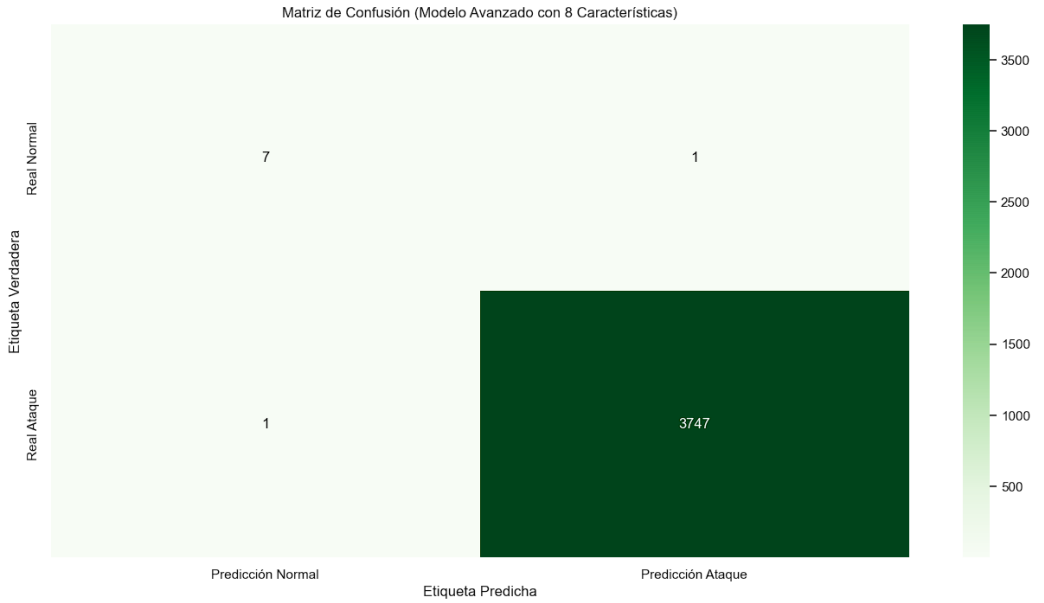

Figura 1: Matriz de Confusión

La Matriz de Confusión visualiza este rendimiento de manera clara: de 3,756 predicciones, el modelo solo cometió 2 errores: un Falso Negativo (un ataque clasificado como normal) y un Falso Positivo (un evento normal clasificado como ataque).

Adicionalmente, se midió la eficiencia computacional del modelo:

a) Tiempo de Entrenamiento: 0.05 segundos.

b) Tiempo de Inferencia: 0.0025 segundos para 3,756 registros.

Estos tiempos demuestran la viabilidad del modelo para una implementación en un entorno de producción, donde la capacidad de respuesta en tiempo real es fundamental.

4.2. Comparación de Modelos y Escenarios

El rendimiento final fue el resultado de un proceso iterativo de mejora. Se compararon cuatro escenarios para evaluar el impacto de la ingeniería de características y el tratamiento del desbalance de clases.

Escenario	Características	Técnica de Balanceo	Accuracy	Recall (Ataque)	Precisión (Normal)

1. Modelo Base	2	Ninguna	0.54	0.53	0
2. Mejora 1	2	SMOTE	0.57	0.57	0
3. Mejora 2	4	SMOTE	0.81	0.81	0.01
4. Modelo Final	8	SMOTE	1	1	0.88

Tabla 1: Resultados sobre los distintos modelos

a) Escenario 1 (Base): Un modelo inicial con solo 2 características y datos desbalanceados fue incapaz de generalizar, detectando apenas la mitad de los ataques.

b) Escenario 2 (con SMOTE): La aplicación de SMOTE por sí sola no fue suficiente, resultando en una mejora marginal. Esto demostró que el problema no era solo el desbalance, sino la falta de información en los datos.

c) Escenario 3 (con 4 Features): La adición de características de contexto (hour_of_day, is_common_user) produjo un salto significativo en el rendimiento, validando la importancia de la ingeniería de características.

d) Escenario 4 (Final): La inclusión de características avanzadas de reputación (ip_success_rate) y ritmo (time_since_last_attempt_ip) fue el factor determinante para alcanzar un rendimiento casi perfecto.

4.3. Interpretación de Resultados (IA Explicable)

Para comprender la lógica interna del modelo final, se utilizó el framework SHAP. El análisis de los valores reveló los factores clave que el modelo aprendió a asociar con el comportamiento anómalo.

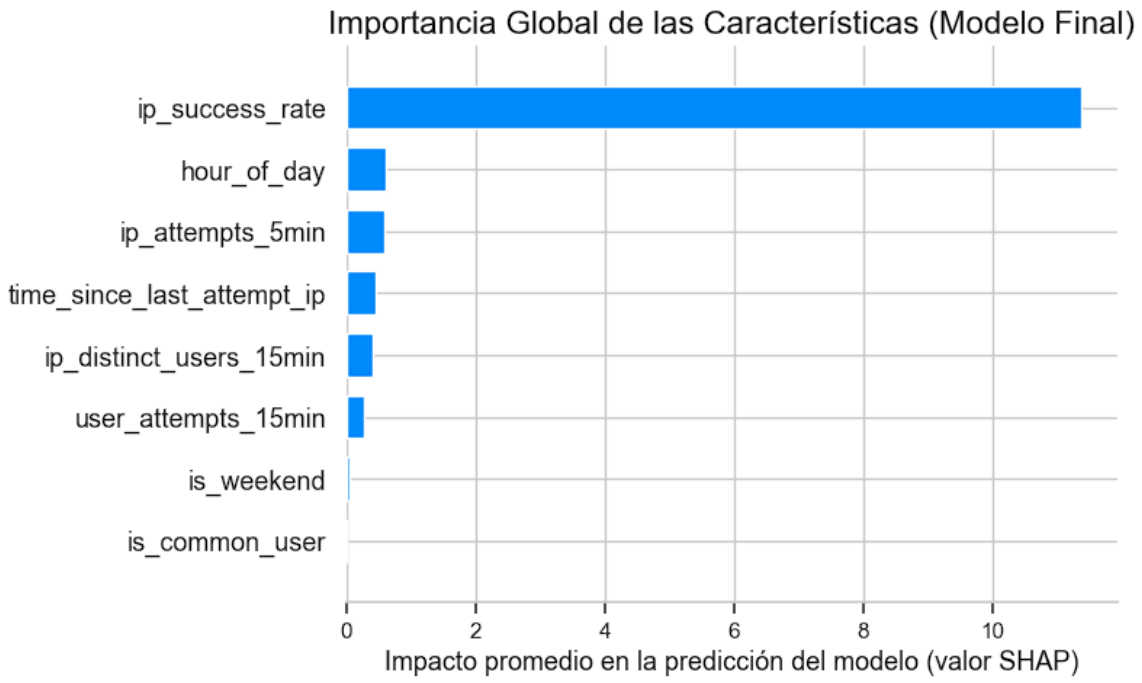

Figura 2: Gráfico de resumen SHAP

Importancia Global de las Características: El gráfico de resumen SHAP indica de manera concluyente que la característica más influyente fue ip_success_rate. Esto representa el hallazgo principal del estudio: el modelo determinó que la reputación histórica de una dirección IP es el predictor más potente de su intención futura, por encima de cualquier otra métrica de comportamiento instantáneo. Características como hour_of_day, ip_attempts_5min y

time_since_last_attempt_ip actuaron como factores secundarios de gran relevancia para refinar la predicción.

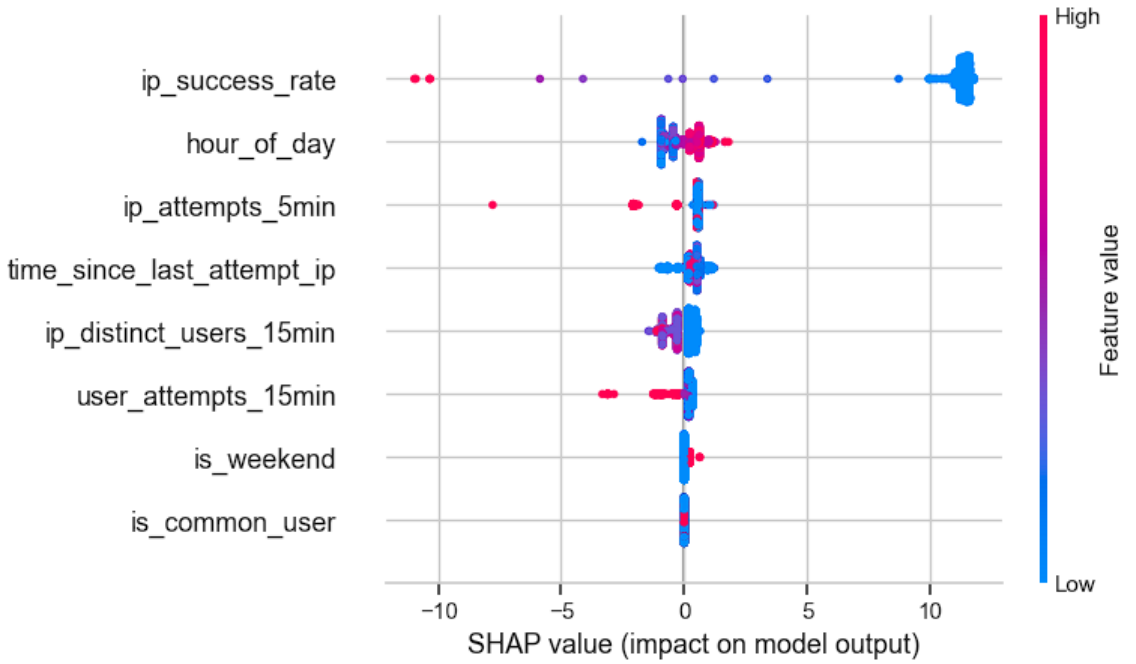

Figura 3: Gráfico de Enjambre SHAP

Análisis Detallado del Comportamiento del Modelo: El gráfico de enjambre (beeswarm plot) confirma la lógica del modelo:

a) ip_success_rate: Valores bajos (puntos azules, indicando una IP con un historial de fallos) empujan fuertemente la predicción hacia "Ataque" (valores SHAP positivos).
b) hour_of_day: Valores bajos (puntos azules, correspondientes a horas de la madrugada) se asocian con una mayor probabilidad de ataque.
c) ip_attempts_5min: Valores altos (puntos rojos, indicando alta frecuencia) son un claro indicador de ataque.
d) time_since_last_attempt_ip: Valores bajos (puntos azules, tiempo corto entre intentos) son identificados como comportamiento robótico y, por tanto, malicioso.

5. Conclusiones

Este estudio culmina con el desarrollo exitoso de un modelo de Machine Learning que alcanza una precisión casi perfecta en la detección de ataques de autenticación. El factor determinante del éxito

no fue simplemente la elección del algoritmo, sino una metodología de ingeniería de características avanzada. La creación de variables que capturan la reputación histórica (ip_success_rate) y el ritmo del comportamiento (time_since_last_attempt_ip) demostró ser fundamental para dotar al modelo de la capacidad de discernir patrones complejos, superando ampliamente a los enfoques más simples.

Se demostró que la ingeniería de características es una estrategia más impactante que el mero tratamiento del desbalance de clases. Si bien la técnica de remuestreo SMOTE fue un paso metodológico necesario para corregir el sesgo en los datos de entrenamiento, su efectividad se maximizó únicamente después de haber enriquecido el dataset con características altamente informativas. Esto subraya un principio fundamental: la calidad y riqueza de los datos son prioritarias sobre los ajustes algorítmicos.

El análisis con IA Explicable (XAI) validó la robustez del modelo y extrajo conocimiento sobre su lógica interna. El hallazgo clave, revelado por SHAP, es que la reputación histórica de una dirección IP es el predictor más potente de una intención maliciosa. Esta conclusión confirma que el modelo aprendió una estrategia de detección coherente con el conocimiento experto en ciberseguridad, priorizando el contexto a largo plazo sobre el comportamiento instantáneo.

Referencias

Mehmmod, A., Batool, K., Sajid, A., Alam, M. M., Su'ud, M. M., & Khan, I. U. (2025). ERBM: A machine learning-driven rule-based model for intrusion detection in IoT environments. Computers, Materials & Continua, 81(1), 1017–1036. https://doi.org/10.32604/cmc.2025.062971

Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Computer Communications, 195, 145–158. https://doi.org/10.1016/j.comcom.2022.12.010

Otoom, A. F., Eleisah, W., & Abdallah, E. E. (2023). Deep learning for accurate detection of brute force attacks on IoT networks. Procedia Computer Science, 215, 157–164. https://doi.org/10.1016/j.procs.2023.03.038

Park, J., Kim, J., Gupta, B. B., & Park, N. (2021). Network log-based SSH brute-force attack detection model. Complexity, 2021, Article ID 6617592. https://doi.org/10.32604/cmc.2021.015172

Sarantos, P., Violos, J., & Leivadeas, A. (2025). Enabling semi-supervised learning in intrusion detection systems. Journal of Parallel and Distributed Computing, 179, 27–40. https://doi.org/10.1016/j.jpdc.2024.105010