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

TÍTULO

Identificación de anomalías en el tráfico de red utilizando algoritmos de aprendizaje automático. Caso de estudio: CENAIM-ESPOL

AUTOR

Chóez Baque, Francisco Alexis

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

Ing. Alicia Andrade Vera, Mgtr.
COORDINADORA DEL PROGRAMA

Firmado electrónicamente por:
DANIEL IVAN
QUIRUMBAY YAGUAL

Lsi. Daniel Quirumbay Yagual, Mgtr.
TUTOR

Firmado electrónicamente por:
EDISSON POMPILIO
QUINTUNA PADILLA

Ing. Edisson Quintuña Padilla, Mgtr.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
ALBERT GIOVANNY
ESPINAL SANTANA

Ing. Albert Espinal Santana, Ph.D.
DOCENTE ESPECIALISTA

Firmado electrónicamente por:
MARIA MARGARITA
RIVERA GONZALEZ

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 FRANCISCO ALEXIS CHOEZ BAQUE, como requerimiento para la obtención del título de Magíster en Ciberseguridad.

TUTOR

Firmado electrónicamente por:
DANIEL IVAN
QUIRUMBAY YAGUAL

Lsi. Daniel Quirumbay Yagual, Mgtr.

Santa Elena, 15 de diciembre de 2024

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

DECLARACIÓN DE RESPONSABILIDAD

Yo, **FRANCISCO ALEXIS CHOEZ BAQUE**

DECLARO QUE:

El trabajo de Titulación, Identificación de anomalías en el tráfico de red utilizando algoritmos de aprendizaje automático. Caso de estudio: CENAIM-ESPOL 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, 15 de diciembre de 2024

EL AUTOR

Firmado electrónicamente por:
FRANCISCO ALEXIS CHOEZ BAQUE

Francisco Alexis Chóez Baque

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 Identificación de anomalías en el tráfico de red utilizando algoritmos de aprendizaje automático. Caso de estudio: CENAIM-ESPOL, presentado por el estudiante, Chóez Baque Francisco Alexis fue enviado al Sistema Antiplagio COMPILATIO, presentando un porcentaje de similitud correspondiente al 8%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.

TUTOR

Firmado electrónicamente por:
DANIEL IVAN
QUIRUMBAY YAGUAL

Lsi. Daniel Quirumbay Yagual, Mgtr.

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

AUTORIZACIÓN

Yo, **Chóez Baque Francisco Alexis**

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 líneas patrimoniales de mi trabajo de propuesta metodológica y tecnológica avanzada con fines de difusión pública, además apruebo la reproducción de este trabajo de propuesta metodológica y tecnológica avanzada 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, 15 de diciembre de 2024

EL AUTOR

Firmado electrónicamente por:
FRANCISCO ALEXIS CHOEZ BAQUE

Francisco Alexis Chóez Baque

AGRADECIMIENTO

Este trabajo no hubiera sido posible sin la ayuda y el apoyo de varias personas e instituciones. En primer lugar, quiero agradecer a mi tutor Lsi. Daniel Quirumbay. Agradezco a la Universidad Estatal Península de Santa Elena y sus docentes por compartirme el conocimiento necesario para la ejecución de este proyecto. Asimismo, extiendo mi gratitud al Centro Nacional de Acuicultura e Investigaciones Marinas, y a la Gerencia de Tecnologías de la ESPOL, por brindarme la información esencial para la realización de esta tesis. Finalmente quiero agradecer a mi familia, especialmente a mis padres, por su apoyo emocional durante todo el proceso de maestría, y a mis amigos y amigas, por su comprensión y paciencia, especialmente cuando debí priorizar este trabajo sobre otros compromisos.

Francisco Alexis, Chóez Baque

DEDICATORIA

A mi familia, por ser el pilar fundamental en mi vida y brindarme su apoyo incondicional. A mis padres, Noemí y Evacio, en especial, por su amor y aliento, sin los cuales este logro no habría sido posible.

A mi hermano y hermanas, a mis cuñados, y también a mis sobrinos.

A mis amigos, por su comprensión y paciencia a lo largo de este proceso, por estar siempre presentes y brindarme palabras de aliento.

Este trabajo es el reflejo de su apoyo y cariño, lo dedico con todo mi amor y gratitud.

Francisco Alexis, Chóez Baque

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El estudio "Identificación de anomalías en el tráfico de red utilizando algoritmos de aprendizaje automático. Caso de estudio: CENAIM-ESPOL" trata de recopilar y analizar el tráfico de red generado, en periodos definidos, en el switch principal de la institución, para prevenir la explotación de vulnerabilidades mediante el entrenamiento de modelos con algoritmos de aprendizaje automático. Se utilizaron algoritmos supervisados y no supervisados, para identificar patrones anómalos. Según los resultados obtenidos la combinación K-Means y SVM fue la mejor para la detección de anomalías. Esto sugiere que su aplicación podría ser muy útil para mejorar la seguridad en la red. Además, al integrar la API de Virus Total, se validó si las anomalías detectadas eran realmente amenazas, lo que permitió tomar decisiones fundamentadas sobre los potenciales riesgos. Se concluye que aplicar técnicas de Machine Learning mejora la capacidad de identificación de amenazas en el tráfico de red del CENAIM-ESPOL.

Palabras claves: Aprendizaje Automático, Anomalía, Red.

ABSTRACT

The study "Identification of anomalies in network traffic using machine learning algorithms. Case study: CENAIM-ESPOL" intends to collect and analyze the network traffic generated, in defined periods, in the institution's main switch, to prevent the exploitation of vulnerabilities by training models with machine learning algorithms. Supervised and unsupervised algorithms were used to identify anomalous patterns. According to the results obtained, the combination of K-Means and SVM models was the best for the detection of anomalies. This suggests that its application could be very useful to improve network security. In addition, by integrating the Virus Total Application Programming Interface (API), it was validated whether the anomalies detected were indeed threats, which allowed to take the right decisions about the potential risks. In conclusion, applying Machine Learning techniques improves the ability to identify threats in CENAIM-ESPOL's network traffic.

Keywords: Machine Learning, Anomaly, Network

INTRODUCCIÓN

En la actualidad, la seguridad de las redes informáticas es una de las preocupaciones más importantes, tanto a nivel personal como institucional, debido al aumento exponencial de dispositivos conectados a Internet y el tráfico de red generado por ellos. De acuerdo con las proyecciones de (CISCO, 2020), se espera que el número de dispositivos conectados globalmente crezca significativamente, lo que incrementa también la superficie de ataque para los ciberdelincuentes. Este aumento en el tráfico y la complejidad de las redes genera un desafío considerable en términos de ciberseguridad, haciendo crucial la detección oportuna de anomalías para prevenir ataques cibernéticos.

El presente trabajo tiene como objetivo analizar el tráfico de red, a través de algoritmos de aprendizaje automático (Machine Learning), para prevenir la explotación de vulnerabilidades que comprometan la seguridad de la red del CENAIM-ESPOL. Se propone la implementación de un sistema con modelos entrenados con algoritmos de aprendizaje automático supervisado y no supervisado que permitan identificar patrones inusuales o amenazas potenciales antes de que se produzcan incidentes graves en la red. Además, se busca evaluar la efectividad de distintos algoritmos como DBSCAN, KMeans, Ramdom Forest y SVM en diferentes escenarios y entornos de red.

Este estudio tiene como alcance la evaluación y comparación de los algoritmos de aprendizaje automático, antes mencionados, para la detección de tráfico anómalo en redes informáticas, con especial enfoque en entornos como el del CENAIM-ESPOL, que enfrenta desafíos relacionados con la creciente cantidad de dispositivos conectados a sus redes internas. Este análisis permitirá identificar cuáles son las técnicas más prometedoras para la detección precisa de amenazas de ciberseguridad.

Este trabajo es importante no solo desde el punto de vista científico, al contribuir al desarrollo de nuevas técnicas para la ciberseguridad, sino también a nivel profesional, ya que ofrece herramientas que pueden ser implementadas para mejorar la protección de redes en entornos empresariales y gubernamentales. Se reduce el impacto por posibles

incidentes, también previene la pérdida de datos, esto porque se obtiene una respuesta más eficiente ante posibles ataques de ciberseguridad.

Esta investigación se organiza de la siguiente manera, para empezar, se presentará la fundamentación teórica del aprendizaje automático en la ciberseguridad. Luego se detallarán los algoritmos utilizados y sus aplicaciones. Al finalizar, se mostrarán los resultados obtenidos de los experimentos realizados en entornos reales, para concluir con un análisis de las ventajas y limitaciones de los métodos aplicados.

Planteamiento de la investigación (Fundamentación de la investigación)

El rápido crecimiento de los dispositivos conectados a redes de computadoras en entornos corporativos y domésticos ha generado un incremento significativo del tráfico de red que los ciberdelincuentes utilizan como campo de trabajo. Adicionalmente, el desarrollo de la ciberdelincuencia y sus modalidades avanzadas tales como phishing, malware, ataques de denegación de servicio, entre otros, han puesto en peligro la información personal y profesional de las empresas, así como su infraestructura crítica. Métodos tradicionales de monitoreo de red, basados en reglas predefinidas o análisis estadísticos simples, resultan ineficientes ante el volumen y complejidad de las amenazas actuales. Por ello, se requiere de enfoques más avanzados y adaptativos que permitan detectar anomalías de manera efectiva.

Mediante la definición este proyecto se determinan que varios algoritmos de ML se analizarán juntos con el fin de realizar una comparación y elegir el más efectivo. A través del análisis automático de grandes cantidades de datos, los algoritmos de ML pueden aprender a identificar patrones inusuales o maliciosos que podrían pasar desapercibidos para las herramientas tradicionales de seguridad de redes. En este contexto, la investigación se centra en la implementación y evaluación de modelos de aprendizaje automático supervisados y no supervisados para la detección de anomalías en el tráfico de la red del centro.

Se justifica este estudio por la necesidad de mejorar la capacidad de las organizaciones para prevenir y mitigar los frecuentes ataques cibernéticos. Un eficiente modelo de

detección de anomalías que se base en Machine Learning podría disminuir significativamente el tiempo de respuesta a los incidentes de seguridad y, por lo tanto, el impacto de un posible ataque. Además, podría radicar en una mejora de la administración de las redes que permitiría a los administradores optimizar sus recursos y disminuir los falsos positivos que causan las alertas innecesarias. Así, la relevancia de esta investigación es alta tanto para la academia como para los profesionales del área de Ciberseguridad.

Además, su aplicación tiene un impacto directo en el ámbito social, porque fortalece la protección de datos sensibles y asegura la continuidad de servicios críticos que dependen de redes informáticas.

Formulación del problema de investigación

¿Cómo el uso de algoritmos de aprendizaje automático puede mejorar la detección de paquetes anómalos en el tráfico de red y reducir la posibilidad de explotación de vulnerabilidades?

Objetivo General:

Implementar un sistema de análisis del tráfico de red basado en algoritmos de aprendizaje automático (Machine Learning), para la detección de comportamientos anormales y la prevención de la explotación de vulnerabilidades que comprometan la seguridad de la red del CENAIM-ESPOL.

Objetivos Específicos:

1. Analizar el tráfico de red actual con el fin de crear un conjunto de datos que facilite la detección de patrones de tráfico normal y anormal en la red del CENAIM-ESPOL.
2. Desarrollar un modelo de detección de anomalías que permita la evaluación del comportamiento en el tráfico de la red utilizando algoritmos de aprendizaje automático.

3. Evaluar las anomalías encontradas en la red del CENAIM-ESPOL con la Interfaz de Programación de Aplicaciones (API) de Virus Total para la verificación y envío de alertas tempranas a los administradores de la red.

Planteamiento hipotético

El uso de algoritmos de aprendizaje automático para la detección de anomalías en el tráfico de red en el CENAIM-ESPOL mejorará significativamente la capacidad de identificar y mitigar posibles amenazas a la seguridad de la red.

CAPÍTULO 1. MARCO TEÓRICO REFERENCIAL

1.1. Revisión de literatura

(Nassif et al., 2021) en el artículo "Machine Learning for Anomaly Detection: A Systematic Review", tiene como objetivo analizar las técnicas de aprendizaje automático aplicadas a la detección de anomalías y resumir las aplicaciones, técnicas, métricas de rendimiento y la clasificación de las anomalías. Mientras procesaban los datos, los autores encontraron 43 aplicaciones de detección de anomalías, 29 modelos de ML y 22 conjuntos de datos relevantes. La idea principal del estudio es que los algoritmos no supervisados son preferibles y que el aprendizaje automático es muy prometedor en la detección de anomalías.

(S. Wang et al., 2021) en su artículo “Machine Learning in Network Anomaly Detection: A Survey” describen las anomalías que pueden presentar una red y afirman que las redes siempre estarán expuestas a amenazas desconocidas. Proteger las redes de los accesos malintencionados es un desafío que la sociedad ha estado estudiando activamente durante muchos años. Los avances en las redes la hacen más vulnerables porque el número de dispositivos aumenta y los atacantes desarrollan formas más complejas y versátiles de infiltrarse. La detección de anomalías implica una actitud receptiva y flexible que cambia de acuerdo con la estructura de la red. Los autores analizan los problemas de la detección de anomalías en las redes heredadas y de próxima generación y cómo ML ha sido utilizado en estos casos. También se explican los procedimientos de diferentes algoritmos

de aprendizaje automático, así como sus metodologías y ventajas, y se ofrece una comparación de diversos modelos de aprendizaje automático utilizados en la detección de anomalías en la red.

(Toshniwal et al., 2020) en el artículo "Overview of Anomaly Detection techniques in Machine Learning", se explora el concepto de eventos raros en conjuntos de datos, definidos como aquellos que se desvían de la mayoría de los patrones regulares y que pueden representar actividades inusuales, como fraudes, intrusiones o eventos anómalos. Estas actividades, que son difíciles de identificar en grandes volúmenes de datos, se denominan anomalías, y su detección es crucial, ya que pueden indicar ataques en la red, caídas o aumentos repentinos en las ventas, propagación de enfermedades o ataques terroristas. Se discute diversas técnicas de detección de anomalías (AD), incluyendo métodos de clasificación, vecinos más cercanos, agrupación, estadísticas, espectros, teoría de la información y gráficos. Además, se señala que la disponibilidad limitada de datos reales puede complicar la elección del algoritmo de detección de anomalías adecuado, el cual debe basarse en factores como el tipo de datos de entrada, el tipo de anomalías, los datos de salida y el conocimiento del dominio.

(Eltanbouly et al., 2020) en el artículo "Machine Learning Techniques for Network Anomaly Detection: A Survey", analiza el creciente interés de los investigadores en el procesamiento distribuido de datos en la computación en la nube, el cual ha hecho que la red sea un objetivo atractivo y vulnerable para ciberdelincuentes que buscan explotar diferentes tipos de vulnerabilidades. Ante este panorama, han existido varias técnicas de detección de intrusiones en evolución continua para proteger los servicios distribuidos en la nube, mediante la detección de varios ataques en la red. En particular, en este artículo se aborda el uso de técnicas de aprendizaje automático en IDS, destacando los desarrollos y continuidad en este aspecto. Se presentan e inspeccionan en detalle algunos de los algoritmos representativos; además, se compara la precisión de la detección de intrusiones y la tasa de detección con diferentes conjuntos de datos y se examina la eficacia y las propiedades de estos métodos en la ciberseguridad de la nube.

(Elmrabit et al., 2020) en su artículo “Evaluation of Machine Learning Algorithms for Anomaly Detection”, se aborda el desafío crítico de la detección de ataques maliciosos en plataformas de redes inteligentes peer to peer, enfatizando que los comportamientos de los atacantes son dinámicos y mejoran con el pasar de los años. El estudio evalúa varios algoritmos de aprendizaje automático (ML) en su capacidad para detectar comportamientos anómalos, realizando la evaluación en los conjuntos de datos públicos: CICIDS-2017, UNSW-NB15 y conjuntos de datos de ciberataques del Sistema de Control Industrial (ICS). La realización de esta investigación fue posible gracias a la infraestructura computacional de alto rendimiento ALICE de la Universidad de Leicester. Basado en el análisis completo de los resultados de los experimentos realizados con respecto a los algoritmos de aprendizaje automático, se estableció que el algoritmo Random Forest mostró el mejor rendimiento en términos de precisión, exactitud, recuperación, F1-score y las curvas ROC en todos los conjuntos de datos que se usaron en los experimentos. Además, se destaca que otros algoritmos presentan un rendimiento similar al de RF, y se concluye que la selección del algoritmo de ML adecuado depende de los datos generados por el sistema de aplicación.

(Al-amri et al., 2021) en el artículo “A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data”, indica que es su objetivo ofrecer una visión completa de las técnicas de vanguardia para detectar anomalías en los flujos de datos generados por dispositivos IoT, abordando los principales desafíos en el procesamiento de dichos datos. Se revisa la literatura existente y se analizan aspectos clave como los tipos de anomalías, el modo de aprendizaje, el modelo de ventana y los conjuntos de datos utilizados. El procesamiento de datos en IoT enfrenta retos relacionados con la evolución de las características, la heterogeneidad de los datos, el uso de ventanas, la selección de parámetros y la complejidad, lo que afecta la precisión y la escalabilidad en grandes volúmenes de datos. Como principales conclusiones, se destacan los desafíos aún no resueltos, como la gestión de datos a gran escala y la heterogeneidad, subrayando la necesidad de mayores esfuerzos de investigación para mejorar las metodologías actuales y proponiendo direcciones futuras para optimizar el procesamiento

de datos IoT. Como conclusiones pueden destacarse los desafíos no resueltos, como la administración de datos a gran escala y la heterogeneidad, resaltando la necesidad de un mayor esfuerzo en investigación para mejorar las metodologías existentes y sugerir las vías futuras de mejora del procesamiento de datos IoT.

(Diro et al., 2021) en el artículo "A Comprehensive Study of Anomaly Detection Schemes in IoT Networks Using Machine Learning Algorithms", se trata de cómo la adopción del Internet de las cosas ha permitido grandes oportunidades de innovación en varias industrias, al mismo tiempo que presenta graves preocupaciones de seguridad debido a las vulnerabilidades inherentes al IoT. Puesto que estos dispositivos a menudo se implementan en un entorno muy diferente al de los sistemas de TI estándar, la protección de su seguridad está plagada de dificultades. Dado que los dispositivos son tanto distribuidos como heterogéneos y tienen recursos limitados, las herramientas de seguridad innovadoras pero simples como el antimalware y el escaneo de virus no son una opción viable. Ya que tanto el lado del dispositivo como el de la red plantean problemas únicos que obstaculizan la eficacia evitando amenazas, se puede argumentar que deben usarse sistemas de detección de anomalías en ambas situaciones. El artículo analiza las investigaciones anteriores y futuras centradas en el uso de algoritmos de aprendizaje automático para los sistemas de detección de anomalías IoT y sugiere que la colaboración con blockchain pueda mejorar aún más la efectividad de estos.

(Poornima & Paramasivan, 2020) en su investigación "Anomaly detection in wireless sensor network using machine learning algorithm", abordan la importancia crítica y los desafíos asociados con la seguridad en las redes de sensores inalámbricos (WSN), donde la detección de anomalía es crucial para garantizar las distintas amenazas que podrían perjudicar los nodos y llevar a mediciones inexactas. El conocimiento de estos datos anómalos es necesario para minimizar posibles falsas alarmas. Dado el rápido crecimiento de la detección de anomalías basada en aprendizaje automático recientemente, la mayoría de los métodos actuales operan en entornos estacionarios y demandar que todos los datos de formación se guardan en el nodo. En esa situación, el documento propuso una regresión de proyección ponderada localmente en línea (OLWPR) para la detección de

anomalías en WSN. OLWPR se base en métodos de regresión no paramétricos, donde las predicciones operan solo en una pequeña muestra de datos, lo que contribuye a una complejidad computacional que es igualmente importante en el caso de escenarios de WSN. PCA (Análisis de Componentes Principales) se utiliza para reducir la dimensionalidad y eliminar datos no relevantes. Finalmente, luego de la predicción, el umbral se establece dinámicamente mediante un método dedicado para descubrir desviaciones. Los resultados muestran que OLWPR logra una tasa de detección del 86% y una tasa de error muy baja del 16%.

1.2. Desarrollo teórico y conceptual

1.2.1 Ciberseguridad

Se llama ciberseguridad a proteger los sistemas informáticos y datos contra ataques maliciosos, esta disciplina también es llamada seguridad de la información electrónica. Esto abarca diversos ámbitos que van desde el empresarial hasta la informática móvil, y se clasifica en categorías específicas: como seguridad de red, seguridad de las aplicaciones, seguridad de la información, entre otras. La ciberseguridad es fundamental para resguardar la integridad y privacidad de la información en el entorno digital (AO Kaspersky Lab, 2024).

La ciberseguridad abarca un conjunto de elementos relacionadas con la prestación de protección en el ámbito del ciberespacio. El concepto de ciberseguridad está relacionado, entre otras cosas, con la protección del espacio de procesamiento de información y las interacciones que tienen lugar en las redes TIC (Rahman et al., 2020).

1.2.2 Machine Learning (Aprendizaje automático)

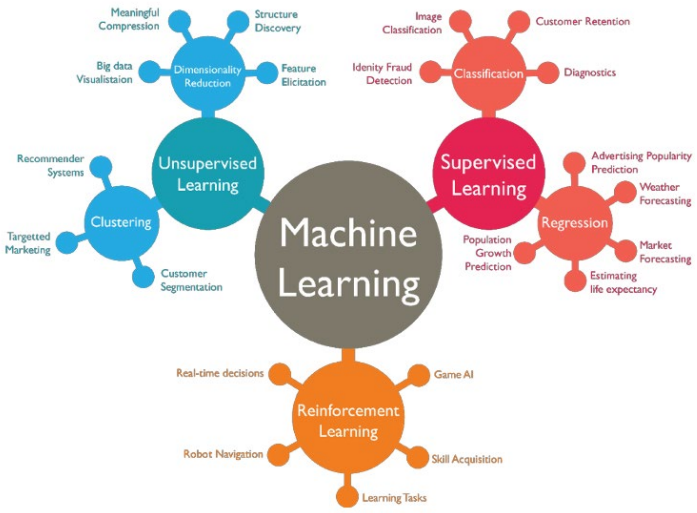
Machine Learning es una subdisciplina de la Inteligencia Artificial que explica cómo una computadora puede aprender a mejorar automáticamente a través de la experiencia. Machine learning está en la intersección de la informática y la estadística y es fundamental para la inteligencia artificial y el análisis de datos. Existen recientes avances debido al desarrollo de nuevos algoritmos y teorías como la abundancia de datos en línea y de computación accesible. Sobre la base de Machine learning, diferentes áreas, como la

medicina, la manufactura, la educación, las finanzas y las estrategias de investigación de tecnologías y comercialización, toman decisiones informadas y basada en evidencia (Jordan & Mitchell, 2015).

El Aprendizaje Automático o Machine Learning es una disciplina de la IA que aborda el desarrollo de algoritmos y métodos predictivos capaces de tomar decisiones basados en estructuras de conjuntos de datos de entrenamiento. La búsqueda de estas estructuras y patrones permite al modelo realizar predicciones y tomar de decisiones en base a pruebas. Existen varios tipos de aprendizaje automático: supervisado, donde se utilizan datos etiquetos para entrenar el modelo y predecir nuevas etiquetas; no supervisado, trabaja con datos no etiquetados para detectar patrones y agrupaciones; semisupervisado, combinación de datos etiquetados y no etiquetados; y aprendizaje por refuerzo; el modelo aprende a través de la retroalimentación basada en recompensas. Cada tipo tiene aplicaciones específicas en áreas como clasificación de imágenes, detección de spam, reconocimiento de voz, sistemas de recomendación, marketing, detección de anomalías, inteligencia artificial de videojuegos y robótica (Martin Ramos, 2021).

Figura 1

Diagrama de clasificación de técnicas ML y aplicaciones



Fuente: Martin Ramos, 2021

Para este proyecto, se ha seleccionado un modelo de Aprendizaje Automático basado en la combinación algoritmos de aprendizaje supervisado y no supervisado. Esta elección se justifica por la necesidad de identificar las anomalías primero agrupando los datos en grupos que representen los que son detectados como anomalías y los que no lo son. Dado que los ataques se pueden clasificar en categorías, el problema se aborda como uno de clasificación en lugar de regresión. Una vez que se han etiquetado las posibles anomalías se aplican algoritmos supervisados que permitan reconocer anomalías en nuevos datos.

1.2.3 Algoritmos de Machine Learning

(Sandoval Serrano & others, 2018) indican que, en el aprendizaje no supervisado, los algoritmos trabajan con datos no etiquetados, agrupándolos según sus características comunes sin tener información previa sobre las categorías a las que pertenecen. El objetivo es identificar patrones y estructuras en los datos que puedan indicar relaciones entre ellos, permitiendo así agruparlos en conjuntos similares.

Según (Y. Wang et al., 2022) el algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise – Agrupamiento Espacial Basado en Densidad con Ruido) es una técnica utilizada para detectar agrupaciones y ruido en bases de datos espaciales. Este algoritmo separa los grupos como el conjutno más amplio de puntos conectados por una determinada densidad. Entre sus ventajas destacan su sencillez a la hora de aplicarlo, útil para separar clústeres con caracteristica distintas, destacando valores atípicos.

K-means es un algoritmo de agrupamiento ampliamente utilizado para organizar grandes volúmenes de datos. Inicialmente propuesto por MacQueen en 1967, es uno de los algoritmos más simples dentro del aprendizaje no supervisado y se empleó para ofrecer una solución a un conjunto previamente conocido (Shafapourtehrany et al., 2022).

En la investigación de (Martin Ramos, 2021), se analizan varios algoritmos de aprendizaje supervisado de Machine Learning con el fin de seleccionar el más adecuado para su implementación en un sistema de detección de intrusiones. Se incluyen el Naive Bayes Classifier, que utiliza el teorema de Bayes para clasificar datos; el Decision Tree Classifier, basado en árboles de decisión invertidos; el Random Forest Classifier, que

emplea múltiples árboles de decisión; las Redes Neuronales Artificiales, inspiradas en el funcionamiento simplificado del cerebro; y las Máquinas de Vectores de Soporte, que separan elementos del dataset en el espacio utilizando hiperplanos de separación.

Los bosques aleatorios son una combinación de múltiples árboles de decisión de manera que cada árbol depende de los valores de un vector aleatorio de un muestreo independiente y con la misma distribución para todos los árboles del conjunto. (Breiman, 2001)

SVM (Vector Support Machine), creado por Vapnik en 2013, es un modelo supervisado de aprendizaje automático (ML) que se utiliza para el reconocimiento de patrones y el análisis de datos. Ha sido ampliamente aplicado en tareas de regresión y pronóstico en áreas como la agricultura, hidrología, meteorología y estudios ambientales (Fan et al., 2018).

1.2.4 Reporte de Clasificación

(Medium, 2024) indica que un reporte de clasificación es un resumen detallado que evalúa el desempeño de un modelo de clasificación. Proporciona un resumen detallado de métricas clave que permiten medir la efectividad del modelo en diferentes aspectos:

- **Precisión (Accuracy):** Refleja la proporción de predicciones correctas entre el total de predicciones realizadas, y que en casos donde las clases están balanceadas se puede utilizar como métrica.
- **Precisión específica (Precision):** Es una variante de precisión y evalúa qué porcentaje de las instancias predichas como positivas son verdaderamente positivas.
- **Sensibilidad (Recall):** Mide la capacidad del modelo de identificar correctamente todas las instancias positivas.
- **F1-Score:** Combina precisión y sensibilidad en una sola métrica que equilibran ambas. Es útil cuando se trabajan con clases desbalanceadas.

En relación con el caso de estudio, estas métricas permiten identificar la efectividad de los algoritmos en la detección de amenazas. Una baja precisión indica alta tasa de falsos positivos, al igual que un bajo valor en la sensibilidad (Recall) disminuirá la detección de anomalías reales. Su combinación con métodos como el uso de la API de Virus Total para la validación de los resultados será fundamental para evaluar el éxito de la solución.

1.2.5 Matriz de Confusión

En el ámbito de la inteligencia artificial y aprendizaje automático la Matriz de Confusión, según (Barrios Arce, 2019), se refiere a una herramienta para visualizar el rendimiento de un algoritmo de aprendizaje supervisado. Los resultados del modelo se muestran en la matriz de tal forma que las columnas contienen las predicciones dadas por el modelo para cada clase, y las filas contienen las instancias correspondientes a las clases reales. En términos prácticos, permite observar los aciertos y errores que el modelo está cometiendo durante el proceso de entrenamiento con los datos.

Figura 2

Matriz de Confusión Binaria

Fuente: (Barrios Arce, 2019)

1.2.6 Análisis estadístico

La visualización de las métricas mediante los gráficos estadísticos es vital para interpretar y visualizar los resultados de los algoritmos de Machine Learning (ML). Los gráficos que se utilizarán en este estudio son los siguientes:

- **Curva ROC (Receiver Operating Characteristic)**: Según (Martínez Pérez & Pérez Martin, 2023) es una técnica estadística utilizada para medir la eficiencia de una prueba diagnóstica dicotómica; es decir, específicamente, para visualizar la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos. En este diagrama, una curva más cercana a la esquina superior izquierda optimiza el modelo. El AUC se usa para el rendimiento del modelo y un AUC más cercano a 1 puntúa mejor el modelo para discriminar mejor las clases.
- **Matriz de confusión visualizada**: Muestra gráficamente el análisis de verdaderos positivos, falsos positivos, falsos negativos y verdaderos negativos. Esta visualización gráfica muestra los errores que el modelo podría estar cometiendo, identificando posibles problemas como las altas tasas de falsos positivos o falsos negativos.
- **Gráfico de líneas:** Este gráfico muestra la información como una serie de puntos o la conexión entre estos puntos, formando líneas. Es útil cuando los cambios son sutiles o para comparar datos que se cruzan entre sí. Se puede utilizar con variables categóricas, similar a un gráfico de barras. Es muy útil cuando se trata de series temporales. (IBM, 2021)

Estos gráficos no solo son una manera de evaluar los modelos, sino que también son una excelente forma de identificar patrones que podrían no ser evidentes con las métricas numéricas, el análisis de estos gráficos ayuda a ajustar los modelos y mejorar su capacidad de detección de anomalías.

1.2.7 Sistema de Detección de Intrusos

(Vargas Machuca Del Salto, 2021, p. 30) definen un sistema de detección de intrusos como un programa que alerta al administrador de la red u ordenador cuando descubre un

acceso no autorizado, emitiendo alertas o registros. Un IDS no reacciona activamente ante un ataque, solo indica que ha ocurrido un ataque. Hay dos tipos de IDS para propósitos generales son: HIDS, que monitoriza el tráfico de un host particular, y NIDS, que observa el tráfico de la red en general en busca de actividad sospechosa.

También (Sandoval Serrano & others, 2018, p. 58) menciona que los IDS mejoran la seguridad de la red y protegen los datos de una empresa al detectar actividad sospechosa. Junto con los auditores, los administradores utilizan esta división para detectar y responder a accesos no autorizados o al uso fraudulento de los recursos de información. Los infractores son personas o entidades que quieren acceder a la red privada y los IDS actúan como cortafuegos para proteger a la empresa de ellos.

En la presente investigación se realizará la ejecución de un NIDS, con algoritmos de Machine Learning que puedan predecir alguna anomalía y ser analizadas por medio de la API de Virus Total y, de ser comprobada, mostrar porque las direcciones asociadas a esas anomalías pueden ser maliciosas.

1.2.8 Virus Total

Según (Microsoft, 2024) Virus Total es una plataforma en la web que examina archivos y direcciones URL que generan sospechas, con el fin de identificar diferentes tipos de programas maliciosos y contenido dañino mediante la utilización de motores antivirus y herramientas de análisis de páginas web. Adicionalmente, les proporciona a los usuarios una interfaz de programación de aplicaciones (API) para acceder a los datos generados por Virus Total.

Trabaja con más de 70 escáneres antivirus, servicios de listas de bloqueo de URLs y dominios para inspeccionar archivos y enlaces en busca malware y contenido no deseado. Los usuarios pueden enviar archivos desde sus dispositivos a la interfaz web pública, cargadores de escritorio, extensiones de navegadores y, también a través de la API.

Los resultados se comparten con el remitente y otros socios examinadores para mejorar los sistemas de seguridad. Además, Virus Total ofrece características adicionales como la Comunidad Virus Total, que permite a los usuarios comentar y compartir notas sobre

archivos y URL para detectar contenido malicioso y falsos positivos. (VIRUSTOTAL, 2023).

Para este proyecto se utilizará la API que ofrece Virus Total con el fin de evaluar las anomalías que sean detectadas por los algoritmos de Machine Learning para determinar si se trata de una amenaza o no para la red corporativa.

1.2.9 Python

Python es un lenguaje interpretado y de código abierto, que se ejecuta más rápido que los lenguajes compilados. Es multiplataforma, por lo que los programas desarrollados con Python pueden ejecutarse sin problemas en diferentes sistemas operativos. Con una sintaxis sencilla el aprendizaje de este lenguaje es mucho más simple que otros lenguajes; como por ejemplo Java. Por otra parte, existe un conjunto de bibliotecas básicas extenso que se mantiene con ayuda de una gran comunidad de programadores Python. Estas características lo convierten en el lenguaje preferido por muchos programadores dedicados al desarrollo de sistemas basados en aprendizaje automático. (Pertuz, 2022, p. 15)

En este proyecto se utilizará Python para realizar la limpieza y ordenamiento de los datos recolectados del tráfico de red y también para la programación, pruebas y ejecución de los algoritmos de Machine Learning.

1.2.10 Visual Studio Code

(Microsoft, 2024) indica que Visual Studio Code es editor de código fuente ligero pero potente que funciona en el escritorio y esta disponibles para los sistemas operativos Windows, macOS y Linux. Tiene compatibilidad integrada con JavaScript, TypeScript y Node.js, y dispone de un extenso ecosistema de extensiones que permite añadir compatibilidad con otros lenguajes y entornos de desarrollo, como C++, C#, Java, Python, Go y .NET.

En este proyecto se utilizará VS Code para la desarrollo y edición del código fuente de los algoritmos de Machine Learning (ML), también para realizar las respectivas pruebas y finalmente para la ejecución.

1.2.11 Wireshark

El analizador de protocolos de red más extendido es Wireshark. Tiene características eficientes y se ejecuta en todas las plataformas, incluidas Windows, Linux, OS X y UNIX. Profesionales de redes, expertos en seguridad, desarrolladores y educadores de todo el mundo lo utilizan con regularidad. Es de código abierto que utilizan regularmente expertos en seguridad, profesionales de redes, etc. Un equipo global de expertos en protocolos desarrolló y mantuvo esta herramienta. Wireshark anteriormente solía llamarse Ethereal (Jain & Anubha, 2021, p. 2).

Wireshark es un rastreador de paquetes gratuito que tiene algunos aspectos como análisis, solución de problemas de red, etc. Se utiliza para ver, capturar y analizar paquetes de datos. Tiene un protocolo inalámbrico refinado para ayudar a los administradores a solucionar problemas de redes inalámbricas. Puede capturar el tráfico desde el aire y descifrarlo en ese formato que ayuda a los administradores a rastrear los problemas que causan un rendimiento deficiente y una conectividad intermitente utilizando el soporte de controlador adecuado (Jain & Anubha, 2021, p. 2).

Con Wireshark se obtendrá los datos del tráfico de red que se genera en el switch core de la institución y con Python se le dará el formato necesario para que pueda ser interpretado de manera correcta por los algoritmos de Machine Learning (ML).

1.2.12 Herramientas de análisis de tráfico de red

Además de Wireshark existen otras herramientas que se utilizan de manera mecánica, es decir, automatizadas o con mínima intervención manual, para realizar estas funciones. A continuación, se presentan algunas de las herramientas más relevantes en este ámbito:

1. **tcpdump**: Es una herramienta de línea de comandos diseñada para examinar el tráfico de red (Chóez Quimis & Mero Suárez, 2023, p. 280). tcpdump puede ser

automatizado en servidores para realizar capturas continuas de tráfico y almacenar los datos para un análisis posterior, siendo ideal para monitorear patrones anómalos o tráfico sospechoso en redes críticas.

2. **nmap**: Es una herramienta que se utiliza para realizar escaneos de redes, también ofrece opciones para restringir o denegar accesos (Colque, 2021, p. 22). Este escanear de puertos tiene la funcionalidad para detectar servicios activos y explorar las redes.

3. **Snort**: Fue desarrollado por Martin Roesch (1998), ahora es gestionado por CISCO, es uno de los productos de seguridad más conocidos y utilizados. Es un sistema de detección de intrusos en red (NIDS) que se desarrolló como software de código abierto y está escrito en lenguaje C. Este sistema incluye un lenguaje que permite crear reglas personalizadas e incorpora una serie de filtros ajustables predefinidos que se adaptan a diversas necesidades. Su versión más reciente, Snort 3.0, puede configurarse como sniffer, registrador de paquetes y NIDS (Castillo Mendoza, 2022, p. 11).

4. **Suricata:** Es un motor de alto rendimiento para IDS, IPS y seguridad de red, desarrollado por el OISF. Esta aplicación de código abierto y multiplataforma pertenece a la comunidad Open Information Security Foundation (OISF), una fundación sin ánimo de lucro (Castillo Mendoza, 2022, p. 12).

5. **Nagios:** Es una herramienta de código abierto, utilizada para supervisar la infraestructura de las redes y alertar sobre algún inconveniente que se pueda presentar. Puede monitorear tanto los servidores, estaciones de trabajo y otros dispositivos de red, también servicios como HTTP, SMTP y SSH, entre otros. Los servidores Nagios recopilan información de los clientes y envían alertas en tiempo real cuando se detectan problemas en la red, lo que contribuye a mantener la red y sus servicios funcionando correctamente. (Sáez Agud, 2024, p. 8).

6. **PRTG Network Monitor:** Es un instrumento para garantizar la operación de los sistemas informáticos y prevenir fallos. Esta herramienta es adecuada para la

> administración de redes y se puede utilizar para monitorear el ancho de banda, tráfico de red y muchas otras características. Además, es fácil de instalar y configurar, y de utilizar (López Tello, 2022, p. 11).

Estas herramientas, usadas mecánica o automáticamente, son importantes para la detección y análisis de tráfico malicioso a nivel de redes corporativas. La integración de estas herramientas permite a los equipos de seguridad identificar amenazas en tiempo real, desarrolla una mejor visión de la red y ayuda a tomar decisiones basadas en los paquetes capturados de manera periódica.

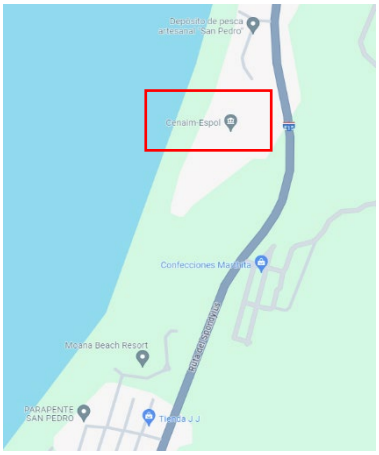
CAPÍTULO 2. METODOLOGÍA

2.1. Contexto de la investigación

Este proyecto se desarrollará en el Centro Nacional de Acuicultura e Investigaciones Marinas (CENAIM) que pertenece a la Escuela Superior Politécnica del Litoral (ESPOL) y se encuentra ubicado en la comuna San Pedro de la parroquia Manglaralto al norte de la provincia de Santa Elena en las siguientes coordenadas -1.9545382994322626, -80.72937811976382 o se puede seguir el siguiente enlace: https://maps.app.goo.gl/KjxbAP3HM1gyByi87.

Figura 3

Ubicación geográfica del CENAIM-ESPOL



Fuente: Google Maps

2.2. Diseño y alcance de la investigación

El diseño que se utilizará en esta investigación es no experimental transversal por que se llevarán a cabo recolección de datos en un periodo específico para luego ser evaluados, en caso de presentar anomalías, por medio de los algoritmos de Machine Learning (ML).

Mientras que el alcance será correlacional porque se estudiará la relación entre la implementación de los algoritmos de aprendizaje automático (ML) para la detección de paquetes anómalos en el tráfico de red y la mejora que se tendrá en cuanto a la capacidad de identificar anomalías.

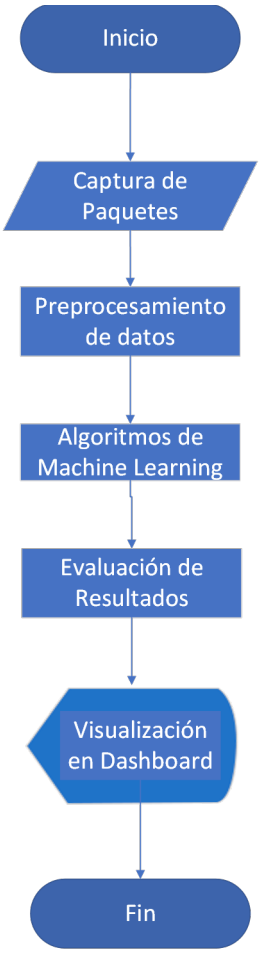
Para esta propuesta de un sistema de detección de anomalías se generó una arquitectura que integra varias etapas de procesamiento de datos, algoritmos de Machine Learning y una interfaz de visualización interactiva. Está compuesta por los siguientes bloques:

- Captura de datos del tráfico de red: Recolectar paquetes desde el switch core de la red, utilizando herramientas como Wireshark o tcpdump, estos datos se almacenan para luego ser procesados.
- Preprocesamiento de datos: Realizar limpieza a los datos capturados anteriormente, este proceso incluye normalización, conversión a características numéricas, eliminación de valores atípicos y de duplicados.
- Algoritmos de Machine Learning: Aquí se realiza el entrenamiento de los diferentes modelos, los cuales permitirán la identificación de anomalías en el tráfico de red.
- Evaluación de resultados: Luego de entrenar los modelos se procesan nuevos datos y se genera un informe con los resultados analizados por Virus Total, lo que permite generar las alertas necesarias.
- Interfaz de visualización: Esta es la etapa final donde se presentan los resultados a través de un dashboard, donde los usuarios pueden observar las anomalías detectadas y que tipo de riesgo representa para la red de la institución. Las direcciones IP's solo podrán ser visualizadas para los usuarios administradores, mientras que los usuarios estándares podrán subir datasets para generar los informes, pero no serán capaces de observar las IP's que los modelos detecten como anomalías.

A continuación, se muestra el diagrama de la arquitectura del sistema propuesto:

Figura 4

Diagrama de la Arquitectura del sistema propuesto



2.3. Tipo y métodos de investigación

Se aplicará el tipo de investigación Mixto, puesto que se analizan el número de paquetes (Cuantitativo) y sus características (Cualitativo).

El método de investigación a utilizar es el Hipotético-Deductivo porque a partir de una hipótesis se buscará comprobarla o negarla, y se llegará a conclusiones basadas en los resultados obtenidos.

2.4. Población y muestra

En el CENAIM-ESPOL, la red LAN y WI-FI conecta dispositivos como PC, laptops, celulares, tabletas, entre otros. Aproximadamente el 97% de los usuarios del centro tienen al menos un dispositivo que se conecta a la red, esto genera un tráfico considerable durante el horario laboral (08:00 a 16:30). Según datos facilitados por el personal de tecnologías del CENAIM, se estima que hay alrededor de 250 dispositivos conectados simultáneamente a la red.

En esta investigación, la población de estudio está conformada por los paquetes de tráfico de red generados en la institución durante el horario laboral. Dado que el tráfico de red es variable, se recolectarán datos en diferentes horarios y días de la semana (de lunes a viernes), con el fin de obtener una muestra más representativa. La muestra será no probabilística discrecional, centrándose en los horarios de mayor tráfico, previamente identificados. Sin embargo, para evitar sesgos y capturar la variabilidad del tráfico, se incluirán también datos de otros momentos del día.

Dado que la cantidad de paquetes es muy elevada, se trabajará con una muestra de 10 mil paquetes por archivo, recogidos en horarios puntuales entre las 08:00 y las 16:30, durante días laborables en semanas consecutivas. Esta decisión se tomó para representar los periodos de mayor actividad en la red, asegurando que los datos reflejen con precisión el comportamiento de la red del CENAIM-ESPOL.

2.5. Técnicas e instrumentos de recolección de datos

Para obtener los paquetes de datos del tráfico de la red utilizaremos herramientas que permitan la recolección de estos y generar archivos que puedan ser utilizados por los algoritmos de aprendizaje automático, en caso de no ser así se hará un proceso en el cual se pueda hacer una limpieza y ordenamiento de los datos para poder ser utilizados por los algoritmos.

A continuación, se mencionan varias herramientas que se pueden utilizar: **Nagios, PRTG Network Monitor, Pandora FM**S, entro otros. La mayoría de estos trabajan con licencia de pago, pero en su versión libre se puede realizar ciertas operaciones. En este proyecto

utilizaremos WireShark que es un sniffer gratuito y de código abierto que permite la recolección de todo el tráfico que pasa a través del switch core de la institución.

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

Wireshark, en últimos años, se ha vuelto una herramienta fundamental para el análisis de redes y protocolos. Gracias a su capacidad para capturar paquetes y examinarlos en tiempo real, sumada a su interfaz amigable y la gran comunidad de profesionales que ayudan con el soporte, se ha convertido en la mejor elección de las personas que trabajan en el ámbito de la seguridad de las redes de computadoras. Se puede tomar como uno de los mejores ejemplos de lo que es el software de código abierto y su colaboración con la comunidad. Seguirá siendo una herramienta confiable y esencial para el análisis de redes en el futuro, si se mantiene en un constante desarrollo. Mientras las redes evolucionan continuamente se vuelven más complejas, en este contexto Wireshark continuará como una herramienta indispensable para los profesionales que se encargan de mantener, optimizar y proteger la conectividad en el mundo digital. (Listopro Community, 2024).

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

Para realizar un análisis adecuado de los paquetes recolectados, primero se debe crear un conjunto de datos que pueda ser interpretado correctamente por los algoritmos de Machine Learning. Por esta razón, se realizó una lectura inicial, con Python, de los archivos .pcapng recolectados con Wireshark, se ejecuta un script, de elaboración propia, para realizar la limpieza y seleccionar los campos adecuados, obteniendo un DataFrame con las columnas necesarias para el análisis. Para este estudio, se seleccionaron los siguientes campos: Source, Destination, Protocol, Length, Port Source y Port Destination.

Antes de etiquetar los paquete normales y anómalos, se tomó una muestra del 1,5 % del dataset generado para realizar un análisis previo, con Virus Total y ApiVoid, de las IPs que podrían ser identificadas como anomalías para luego ser comparadas con los resultados de los modelos entrenados. Del análisis inicial se obtuvieron los siguientes resultados:

Figura 5

Resultados de análisis inicial (Source)

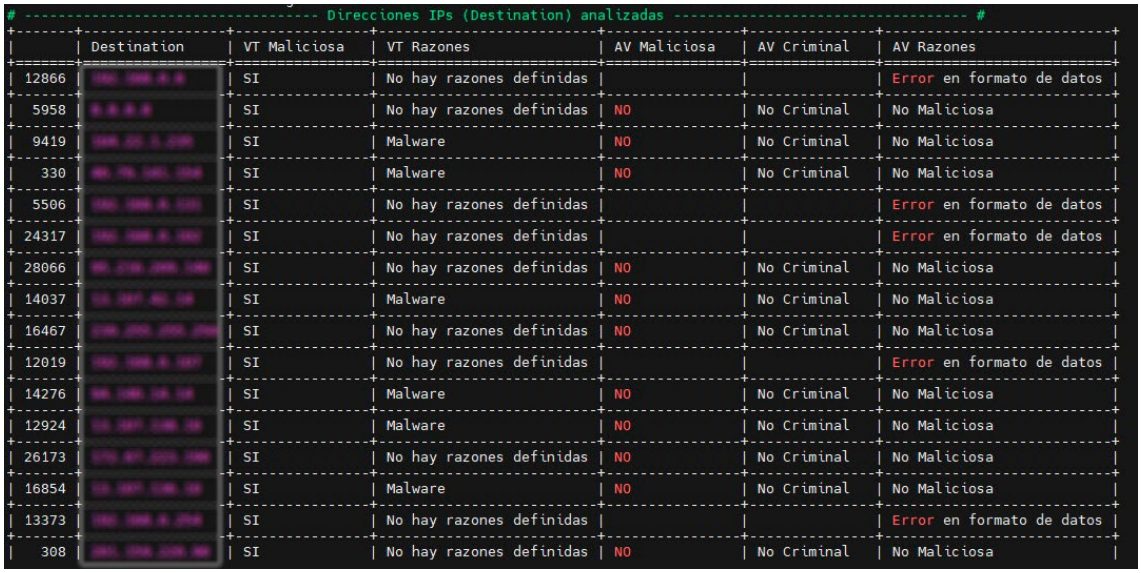
---------- Direcciones IPs (Source) analizadas ----------

	Source	VT Maliciosa	VT Razones	AV Maliciosa	AV Criminal	AV Razones
15941	[redacted]	SI	No hay razones definidas			Error en formato de datos
8333	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
6958	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
5181	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
10892	[redacted]	SI	No hay razones definidas			Error en formato de datos
11264	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
14497	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
3191	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
12627	[redacted]	SI	No hay razones definidas			Error en formato de datos
14279	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
28123	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa

El dataset de Source tiene: 11 registros

Figura 6

Resultados de análisis inicial (Destination)



---------- Direcciones IPs (Destination) analizadas ----------

	Destination	VT Maliciosa	VT Razones	AV Maliciosa	AV Criminal	AV Razones
12866	[redacted]	SI	No hay razones definidas			Error en formato de datos
5958	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
9419	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
330	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
5506	[redacted]	SI	No hay razones definidas			Error en formato de datos
24317	[redacted]	SI	No hay razones definidas			Error en formato de datos
28066	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
14037	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
16467	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
12019	[redacted]	SI	No hay razones definidas			Error en formato de datos
14276	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
12924	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
26173	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa
16854	[redacted]	SI	Malware	NO	No Criminal	No Maliciosa
13373	[redacted]	SI	No hay razones definidas			Error en formato de datos
308	[redacted]	SI	No hay razones definidas	NO	No Criminal	No Maliciosa

Dado que los datos recolectados no cuentan con etiquetas que indiquen si son anomalías o no, se procedió a analizar el DataFrame utilizando algoritmos no supervisados, que permiten identificar qué paquetes podrían ser anomalías. De esta manera, se etiquetaron los paquetes, en una nueva columna (Anomalía), como 1 para aquellos que no presentaron anomalías, es decir, "normales", y con -1 para los paquetes que fueron detectados como anomalías por los algoritmos. En este caso, se emplearon los algoritmos no supervisados DBSCAN (Density-Based Spatial Clustering of Applications with Noise – Agrupamiento Espacial Basado en Densidad con Ruido) y K-Means.

Los algoritmos DBSCAN y K-Means fueron seleccionados por sus características que los hacen especialmente adecuados para el análisis de anomalías en conjuntos de datos sin etiquetas.

DBSCAN es eficaz en presencia de ruido y valores atípicos. Los otros algoritmos no funcionan tan bien en presencia de ruido, lo que hace que DBSCAN sea un algoritmo eficaz. Al tomar puntos mínimos en los grupos, se garantiza que no se incluyan el ruido y los valores atípicos (Singh et al., 2022).

Por otro lado, según (Naoui Mohammed Anouar AND Lejdel, 2020), K-Means es uno de los métodos de agrupamiento por particiones más utilizados; se destaca por su simplicidad, naturaleza estadística y alta escalabilidad. Asimismo, su tiempo de ejecución crece de manera lineal en función de cualquier variable del problema, proporcionando una visión general de cómo se distribuyen los paquetes en grupos "normales" o anómalos.

Una vez etiquetados los paquetes como anómalos (-1) y no anómalos (1), se procede a analizar el conjunto de datos utilizando algoritmos supervisados. Para este trabajo, se han seleccionado Random Forest (Bosque Aleatorio) y SVM (Support Vector Machine – Máquina de Vectores de Soporte) debido a su robustes, precisión y eficacia en escenarios donde los datos son complejos y las clases pueden ser desequilibradas.

Los modelos de bosques aleatorios son más estables y reproducibles utilizando varios tamaños y subconjuntos de datos, y superan a otros métodos (Redes Neuronales Artificiales, entre otros) en términos de éxito y tasa de predicción (Josso et al., 2023).

Por su parte SVM ha demostrado un gran éxito en diversas aplicaciones de clasificación, como la detección de intrusiones, el reconocimiento de voz e imágenes, la clasificación de texto y el análisis químico, entre otros. Su versatilidad radica en su excelente capacidad de aprendizaje, lo que lo hace adecuado para una amplia variedad de problemas. Entre sus principales ventajas se encuentran un tiempo de predicción reducido y la capacidad de garantizar una solución óptima global en la clasificación de objetivos (Mustafa Abdullah & Mohsin Abdulazeez, 2021).

Anteriormente se explicó cuáles son los algoritmos supervisados y no supervisados que se utilizarán, así como los datos que se analizarán. Dado que se emplearán 4 algoritmos (2 supervisados y 2 no supervisados), se realizará una combinación de estos. El orden de utilización de los algoritmos será el siguiente: DBSCAN – Random Forest, DBSCAN – SVM, K-Means – Random Forest, K-Means – SVM.

Con estos algoritmos se busca determinar si las anomalías corresponden a algún tipo de amenaza digital, como malware, phishing, spam o adware. Para identificar si pertenecen a alguna de estas categorías, se utilizará la API de Virus Total.

Primero, se generó el código Python con ayuda del IDE VisualStudio Code y se realizaron pruebas con archivos pequeños de 100 paquetes, luego se procede a ejecutar los algoritmos desde Python en su versión 3.12, se ajustaron las funciones y se desarrolló el control de errores.

Es importante recordar que cada archivo inicia con 10 mil paquetes, tomados en dos periodos distintos del día, uno en la mañana y otro en la tarde, pero tras realizar la limpieza en el DataFrame, este número se reduce debido a la presencia de datos no numéricos, principalmente en las columnas de las direcciones IP de origen y destino. Entre estos casos, pueden encontrar direcciones IPv6 o dominios, como cisco.com, entre otros.

Una vez obtenido un solo dataframe de los archivos procesados, se etiqueta las anomalías con DBSCAN y K-Means, en este punto se puede destacar que K-Means etiquetó más anomalías que DBSCAN. Es decir, se obtiene dos dataframe uno con anomalías etiquetadas por DBSCAN y otro por K-Means.

Cuando los algoritmos estuvieron desarrollados y probados, se procedió a ejecutarlos con el dataframe generado anteriormente.

Para la primera parte se utiliza el dataframe con anomalías etiquetas por DBSCAN y se realiza el análisis con RF (Random Forest) y SVM respectivamente.

A continuación, se muestran los resultados obtenidos, en el entrenamiento, al aplicar RF en el dataframe que tiene anomalías etiquetada por DBSCAN.

Figura 7

Matriz de Confusión DBSCAN-RF - Entrenamiento

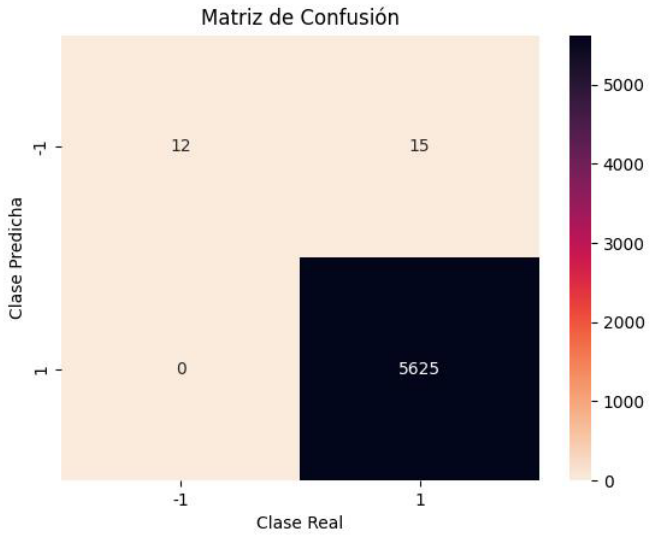


Figura 8

Reporte de Clasificación DBSCAN-RF - Entrenamiento

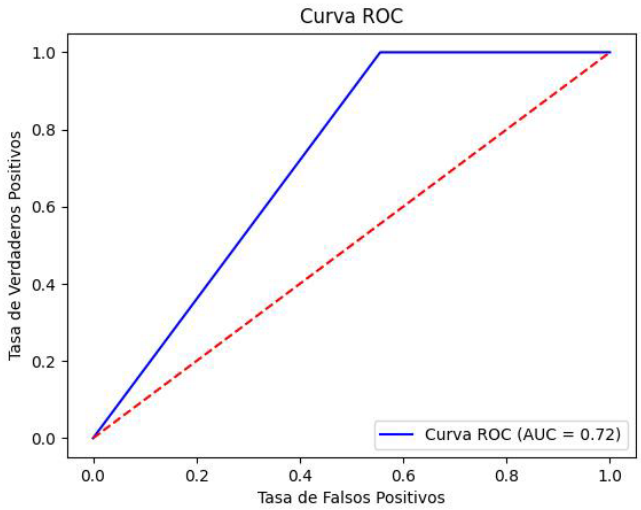
Reporte de Clasificación:

	precision	recall	f1-score	support
-1	1.00	0.44	0.62	27
1	1.00	1.00	1.00	5625
accuracy			1.00	5652
macro avg	1.00	0.72	0.81	5652
weighted avg	1.00	1.00	1.00	5652

En la matriz de confusión (Figura 4) se observa que existen 15 falsos positivos (FP) y ningún falso negativo (FN). El modelo presenta, en promedio macro (Figura 5), un recall (sensibilidad) del 72% y una media armónica (F1-score) del 81%. Estos resultados indican que el modelo tiene un buen rendimiento, aunque tiene margen de mejora en la detección de anomalías ya que tiene un recall del 44%.

Figura 9

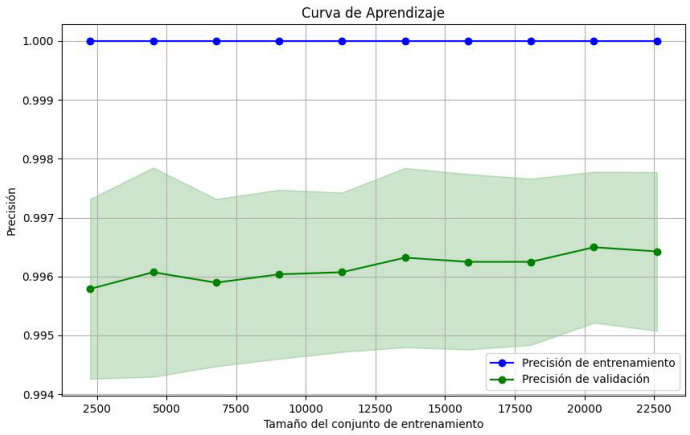
Curva ROC DBSCAN-RF - Entrenamiento



La curva ROC (Figura 6) indica que el modelo tiene dificultades para separar correctamente las clases. Con un AUC del 0.72 el modelo necesita ajustes para mejorar su capacidad predictiva.

Figura 10

Curva de Aprendizaje DBSCAN-RF



En la curva de entrenamiento (Figura 7) se puede notar mucha separación entre las líneas que indican la precisión de entrenamiento y validación (o prueba), lo que indica que el

modelo está sobreajustado al conjunto de entrenamiento, esto podría generar que el modelo no se aplique bien a nuevos datos.

Ahora se muestran los resultados obtenidos, en el entrenamiento, al aplicar SVM en el dataframe que tiene anomalías etiquetada por DBSCAN.

Figura 11

Matriz de Confusión DBSCAN-SVM - Entrenamiento

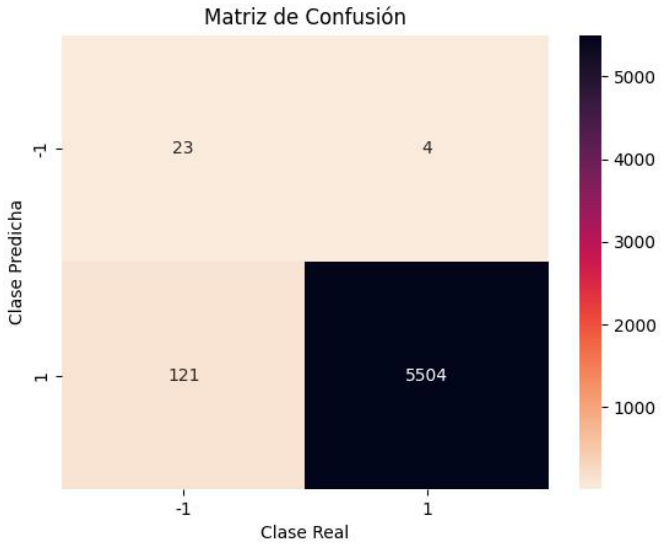


Figura 12

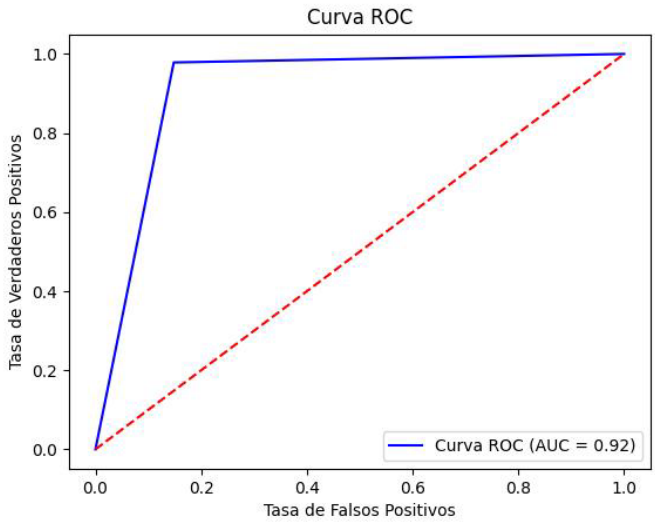
Reporte de Clasificación DBSCAN-SVM - Entrenamiento

Reporte de Clasificación:	precision	recall	f1-score	support
-1	0.16	0.85	0.27	27
1	1.00	0.98	0.99	5625
accuracy			0.98	5652
macro avg	0.58	0.92	0.63	5652
weighted avg	1.00	0.98	0.99	5652

La matriz de confusión (Figura 8) indica que el modelo detecto 4 falsos positivos (FP) y 121 falsos negativos (FN). Al tener una precisión del 16%, en la clases -1, indica que existe un desequilibrio de clases lo afecta negativamente la precisión y aunque el recall (sensibilidad) es alto porque el modelo identifica correctamente el 85% de los ejemplos de la clase -1, la precisión baja.

Figura 13

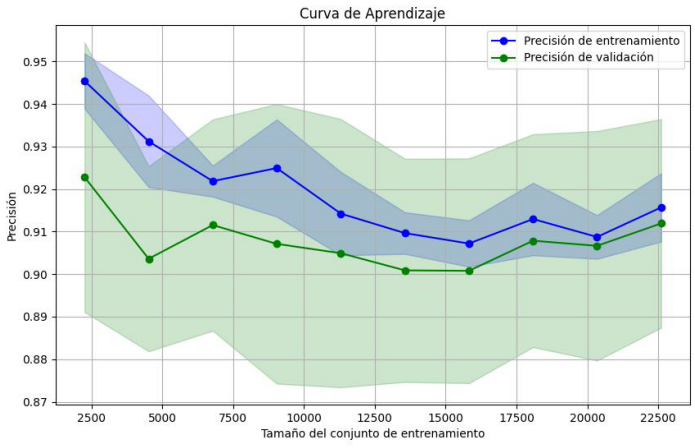
Curva ROC - DBSCAN-SVM - Entrenamiento



La curva ROC (Figura 10), de este modelo muestra que tiene un buen rendimiento con un AUC (Área bajo la curva) de 0.92, esto indica que el modelo tiene una capacidad predictiva fuerte, separando correctamente las clases en la mayoría de los casos.

Figura 14

Curva de aprendizaje - DBSCAN-SVM



En la Figura 11 se muestra la curva de aprendizaje de las combinaciones de algoritmos DBSCAN-SVM, está indica que el modelo tiene un comportamiento equilibrado y buena

capacidad de generalización, aunque podría haber margen de mejora en su precisión general.

Luego de analizar el dataframe en el que se etiquetaron anomalías con DBSCAN, se procede a realizar el mismo proceso con el dataframe etiquetado con el algoritmo no supervisado K-Means.

De la misma manera, primero se procede a entrenar el modelo con Random Forest (RF), y se obtuvieron los siguientes resultados.

Figura 15

Matriz de Confusión KMeans-RF - Entrenamiento

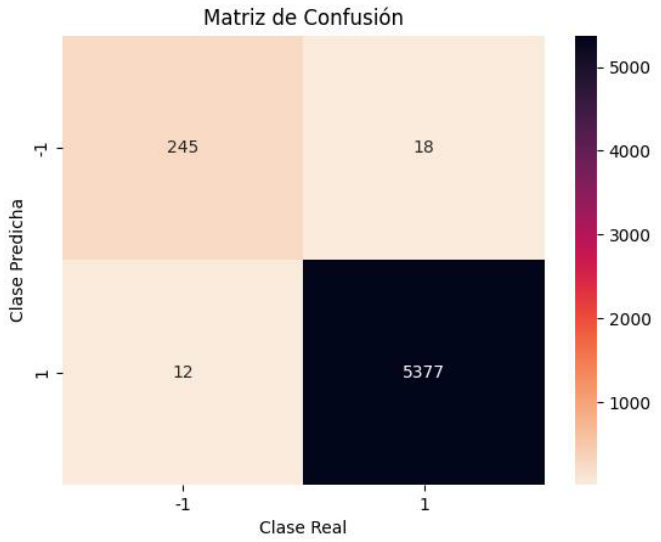


Figura 16

Reporte de Clasificación KMeans-RF - Entrenamiento

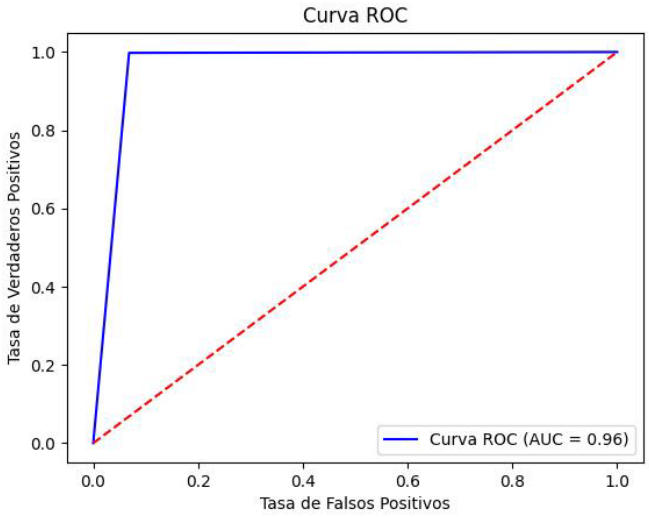
Reporte de Clasificación:

	precision	recall	f1-score	support
-1	0.95	0.93	0.94	263
1	1.00	1.00	1.00	5389
accuracy			0.99	5652
macro avg	0.97	0.96	0.97	5652
weighted avg	0.99	0.99	0.99	5652

Con este modelo se puede notar una mejora en precisión (99%), además se puede notar que los falso positivos y falsos negativos se mantienen bajos en comparación con los valores verdaderos. El recall de las anomalías (-1) es del 93% indicando que detecta la mayoría de las anomalías correctamente.

Figura 17

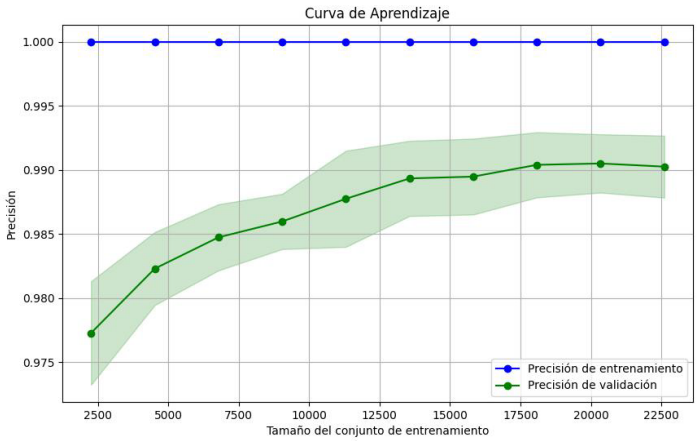
Curva ROC - KMeans-RF - Entrenamiento



Según se ve en la Figura 14, la curva ROC indica que el modelo está funcionando muy bien con un AUC de 0.96, con pocos falsos positivos y alta tasa de verdaderos positivos.

Figura 18

Curva de Aprendizaje - KMeans-RF



La curva de aprendizaje (Figura 15) indica que el modelo parece funcionar bien, sin embargo, se debe verificar que no esté sesgado hacia la clase mayoritaria porque parece haber problemas de clases desbalanceadas.

Para finalizar con el entrenamiento se utiliza la combinación K-Means y SVM, obteniendo los resultados que se muestran a continuación.

Figura 19

Matriz de Confusión KMeans-SVM - Entrenamiento

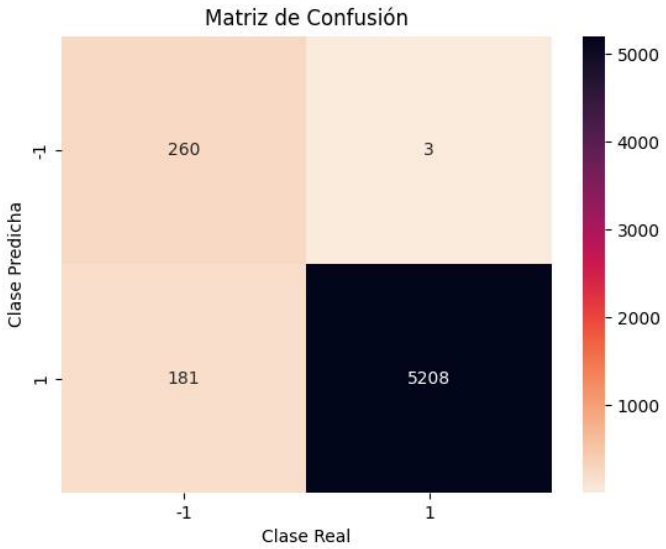


Figura 20

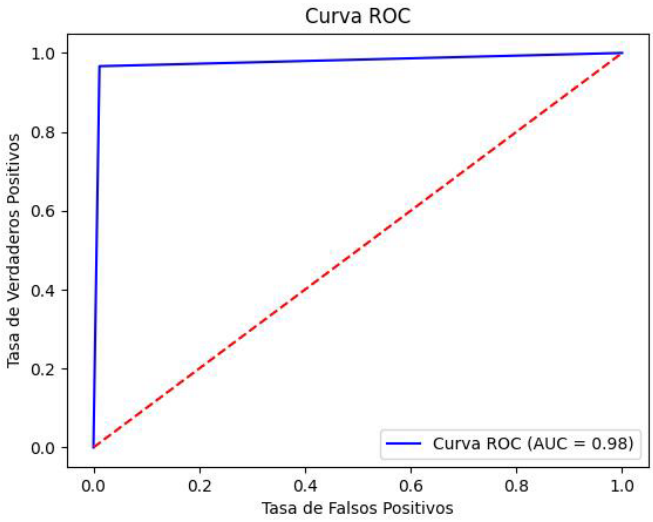
Reporte de Clasificación KMeans-SVM - Entrenamiento

```
Reporte de Clasificación:
              precision    recall  f1-score   support

          -1       0.59      0.99      0.74       263
           1       1.00      0.97      0.98      5389

    accuracy                           0.97      5652
   macro avg       0.79      0.98      0.86      5652
weighted avg       0.98      0.97      0.97      5652
```

Según se observa en las Figuras 16 y 17 el modelo funciona bastante bien para la clase 1, pero para clasificar la clase -1 es algo limitado. Esto podría ser una problemática porque la clase -1 es muy importante en el análisis de este proyecto.

Figura 21

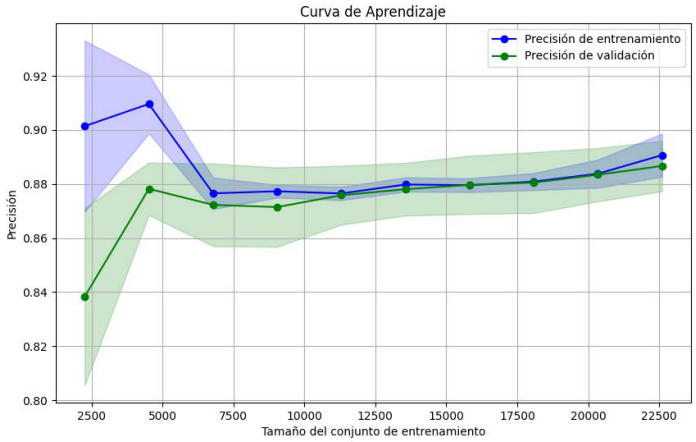
Curva ROC - KMeans-SVM - Entrenamiento



Se puede observar en la Figura 18 la curva ROC de este modelo, el cual tiene un AUC de 0.98 lo que representa un modelo efectivo para discriminar entre las clases. Observando la curva se puede notar que está cerca del vértice superior izquierdo, lo que indica un excelente rendimiento del modelo, ya que logra una alta sensibilidad (recall)

Figura 22

Curva de entrenamiento - KMeans-SVM



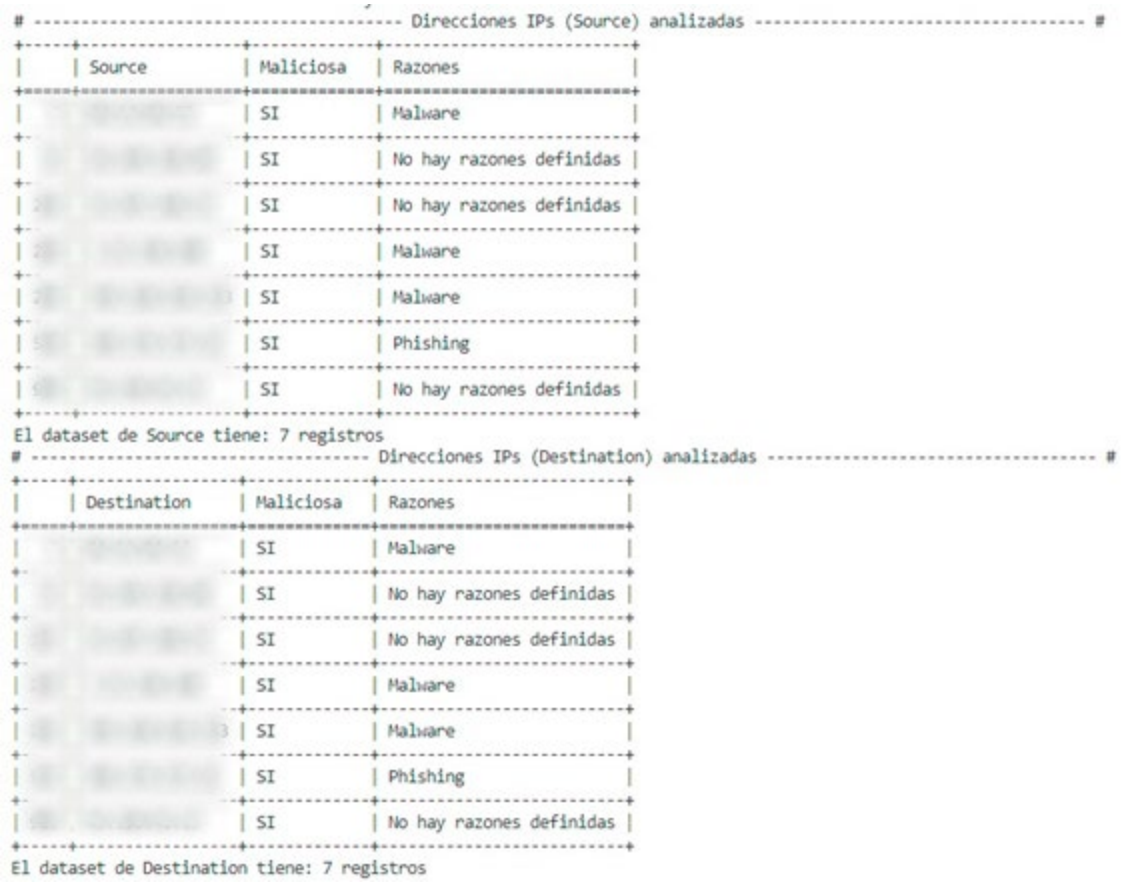
La curva de aprendizaje (Figura 19) muestra, en un principio, las curvas separadas con un conjunto de datos pequeño, pero a medida que los datos aumentan empiezan a converger gradualmente, indicando que el modelo mejora su capacidad para generalizar a medida que incrementa el tamaño del conjunto de datos. La precisión se estabiliza cerca del 0.90 y también las curvas se estabilizan muy cerca entre sí, esto implica que el modelo no está sobreajustando. Según lo observado se puede concluir que este modelo está entrenándose correctamente y que el tamaño del conjunto de datos es adecuado para obtener una generalización aceptable.

Aplicando modelos entrenados a nuevos datos

Una vez finalizado el proceso de entrenamiento se realiza una prueba inicial utilizando datos de tráfico de red (.pcap) que se sabe contienen información maliciosa, para verificar si los modelos entrenados funcionan correctamente. Se utilizó el archivo "2024-09-11-XLoader-infection-traffic.pcap" obtenido de (Malware-Traffic-Analysis.net, 2024), se generó un dataset con los mismo campos que el conjunto de datos de entrenamiento. Se aplica el modelo entrenado con K-Means-SVM para analizar el archivo de tráfico de red y se obtuvieron los resultados que se muestran en la Figura 20.

Figura 23

Resultados K-Means-SVM Tráfico - XLoader

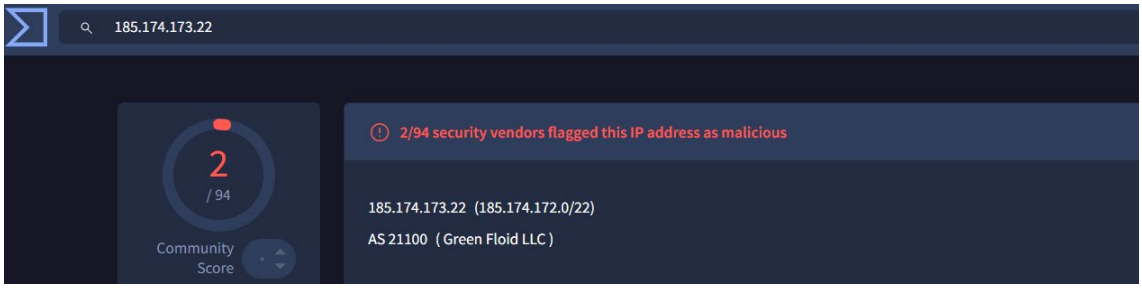


```
# ------------------------------------ Direcciones IPs (Source) analizadas ---------------------------------- #
+-----+-----------------+-------------+--------------------------+
|     | Source          | Maliciosa   | Razones                  |
+=====+=================+=============+==========================+
|     [redacted]        | SI          | Malware                  |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | No hay razones definidas |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | No hay razones definidas |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | Malware                  |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | Malware                  |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | Phishing                 |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | No hay razones definidas |
+-----+-----------------+-------------+--------------------------+
El dataset de Source tiene: 7 registros
# ---------------------------------- Direcciones IPs (Destination) analizadas ---------------------------------- #
+-----+-----------------+-------------+--------------------------+
|     | Destination     | Maliciosa   | Razones                  |
+=====+=================+=============+==========================+
|     [redacted]        | SI          | Malware                  |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | No hay razones definidas |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | No hay razones definidas |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | Malware                  |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | Malware                  |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | Phishing                 |
+-----+-----------------+-------------+--------------------------+
|     [redacted]        | SI          | No hay razones definidas |
+-----+-----------------+-------------+--------------------------+
El dataset de Destination tiene: 7 registros
```

Como se puede observar, las direcciones IP que se muestran son aquellas que podrían ser maliciosas. En algunos casos, se especifica su categoría (malware, phishing), y cuando Virus Total detecta que una IP es maliciosa pero no tiene una categoría definida, muestra el mensaje “No hay razones definidas”. Para realizar una comprobación adicional, estas IPs pueden analizarse manualmente en la página de Virus Total (https://www.virustotal.com/).

Figura 24

Resultados de búsqueda en Virus Total - XLoader



Se puede notar que una de las direcciones IP detectada como anomalía por el modelo entrenado es, efectivamente, clasificada como maliciosa por Virus Total.

También se utilizó el archivo "2024-10-03-SmartLoader-to-Lumma-Stealer.pcap" obtenido de (Malware-Traffic-Analysis.net, 2024) para realizar más pruebas del funcionamiento de los modelos.

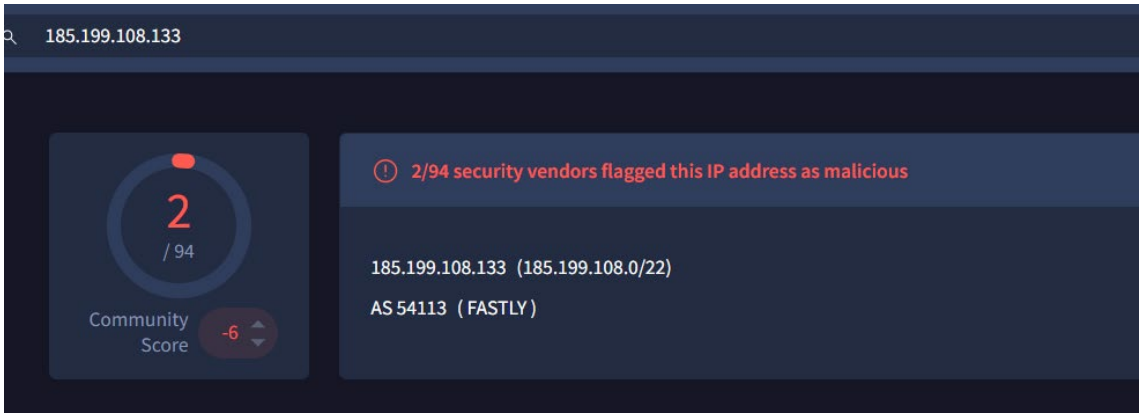
Figura 25

Resultados K-Means-SVM - SmartLoader

```
# ------------------------------------ Direcciones IPs (Source) analizadas ---------------------------------- #
+----+-----------------+-------------+-----------+
|    | Source          | Maliciosa   | Razones   |
+====+=================+=============+===========+
|    [redacted]        | SI          | Malware   |
+----+-----------------+-------------+-----------+
|    [redacted]        | SI          | Malware   |
+----+-----------------+-------------+-----------+
|    [redacted]        | SI          | Malware   |
+----+-----------------+-------------+-----------+
|    [redacted]        | SI          | Malware   |
+----+-----------------+-------------+-----------+
El dataset de Source tiene: 4 registros
# ---------------------------------- Direcciones IPs (Destination) analizadas -------------------------------- #
+-----+-----------------+-------------+-----------+
|     | Destination     | Maliciosa   | Razones   |
+=====+=================+=============+===========+
|     [redacted]        | SI          | Malware   |
+-----+-----------------+-------------+-----------+
|     [redacted]        | SI          | Malware   |
+-----+-----------------+-------------+-----------+
|     [redacted]        | SI          | Malware   |
+-----+-----------------+-------------+-----------+
El dataset de Destination tiene: 3 registros
```

Figura 26

Resultados de búsqueda en Virus Total 2



Como se puede notar, en ambos casos, las direcciones IP que fueron determinadas como anomalías por los modelos entrenados también fueron verificadas como maliciosas en Virus Total, confirmando su naturaleza maliciosa. Esto permite concluir que los modelos están funcionando correctamente, por lo que se procede al análisis de los datos recolectados.

Para el análisis con nuevos datos se recolectaron paquetes en varios días del mes de noviembre, luego de realizar la captura se ejecutó el script para generar un dataset limpio que pueda ser interpretado de manera correcta por los modelos entrenados, en este caso ya no se etiquetó los posibles paquetes con anomalías.

Se generaron varios datasets para probar el funcionamiento de los modelos con diferentes cantidades de registros. En las siguientes páginas se mostrará los resultados obtenidos al aplicar los modelos entrenados en diferentes datasets.

Primero se presentan gráficos del entrenamiento de los modelos y luego los gráficos de los dataset analizados con los diferentes modelos entrenados.

Tabla 1

Distribución de anomalías en entrenamiento

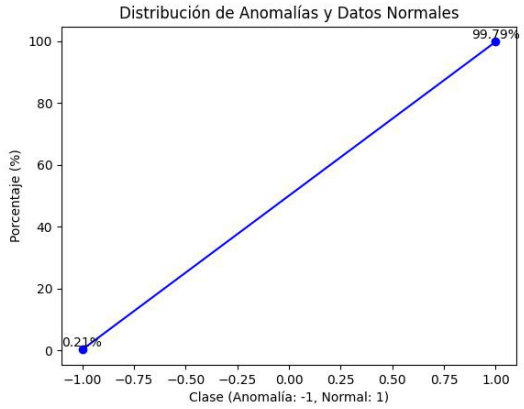
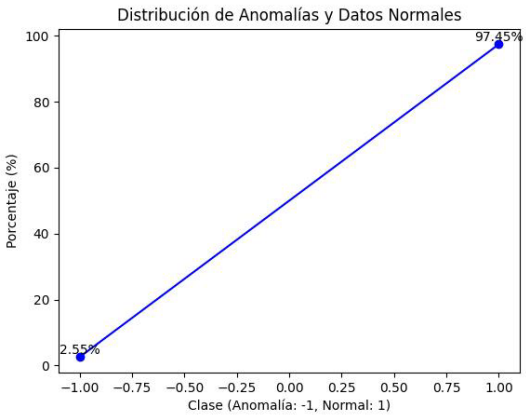
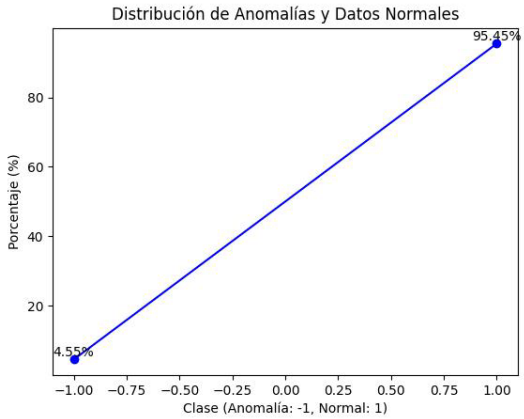
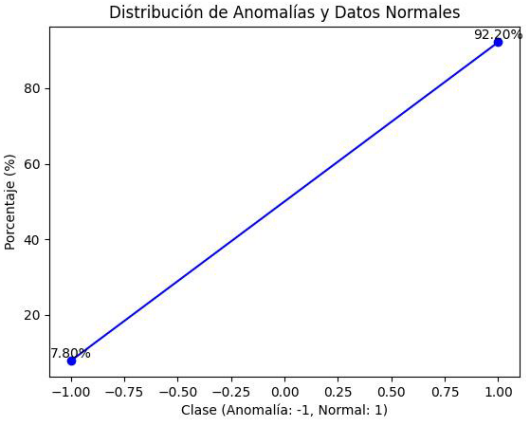
DBSCAN-RF	DBSCAN-SVM
	
KMEANS-RF	**KMEANS-SVM**
	

Tabla 2

Distribución de anomalías en producción (2377 registros)

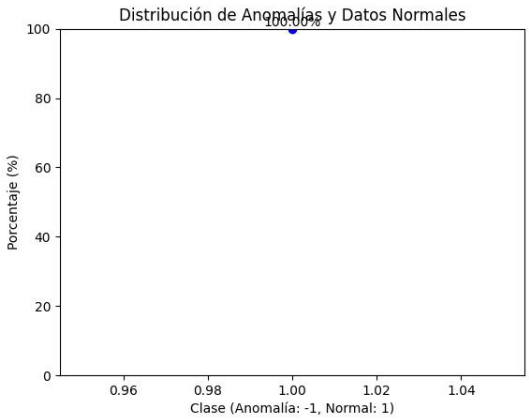
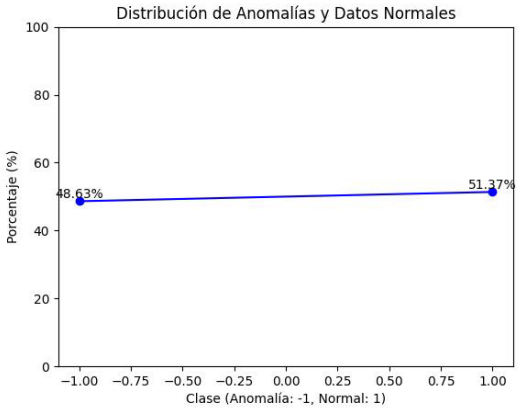
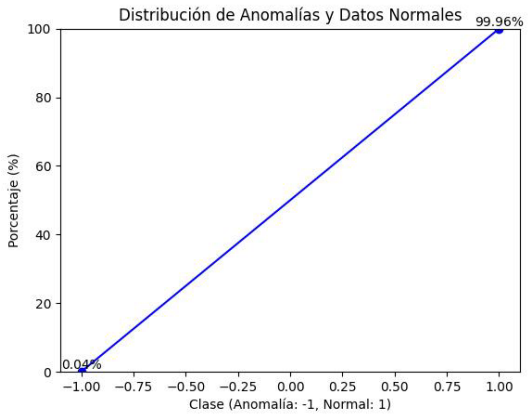
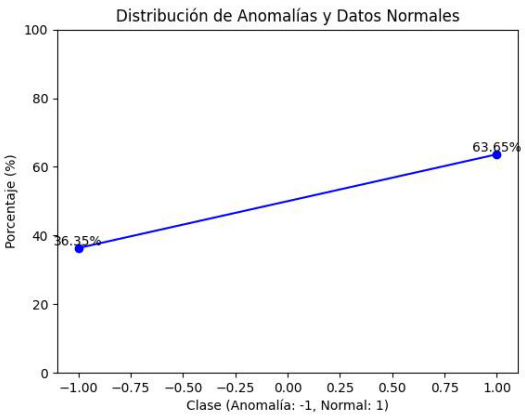
DBSCAN-RF	DBSCAN-SVM
	
KMEANS-RF	**KMEANS-SVM**
	

Tabla 3

Distribución de anomalías en producción (3946 registros)

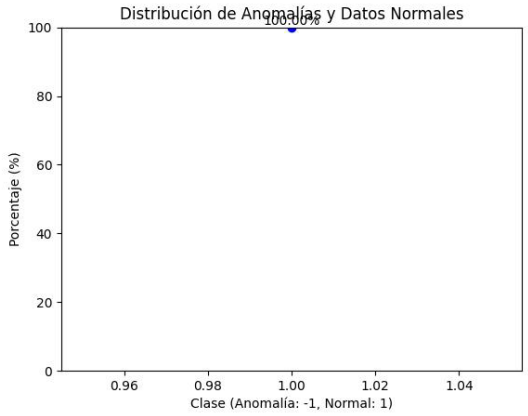
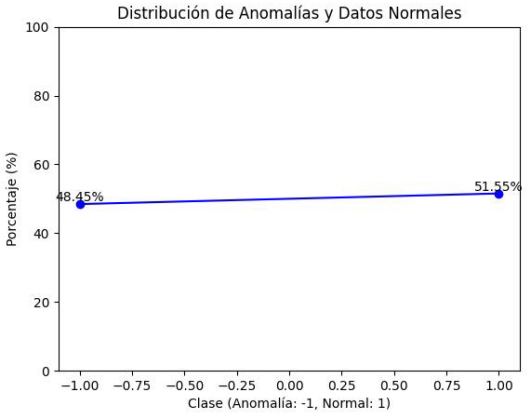
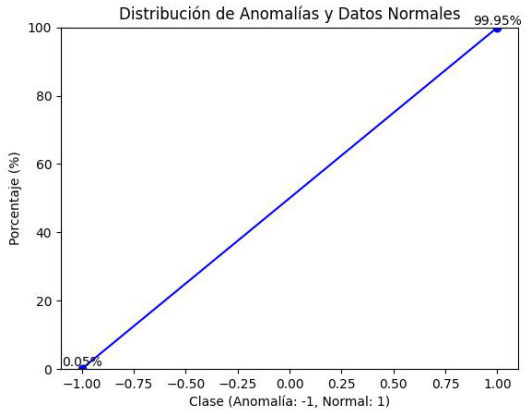
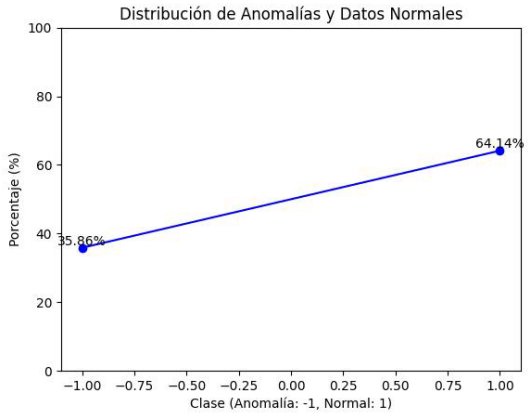
DBSCAN-RF	DBSCAN-SVM
	
KMEANS-RF	**KMEANS-SVM**
	

Tabla 4

Distribución de anomalías en producción (15118 registros)

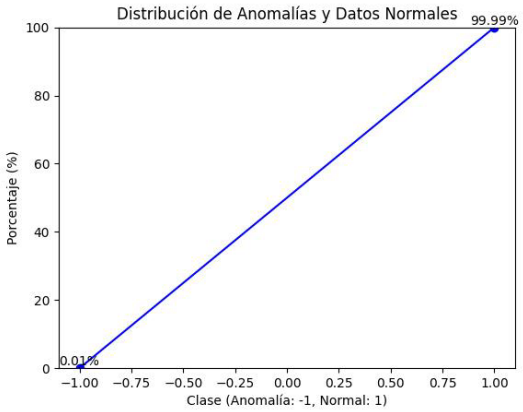
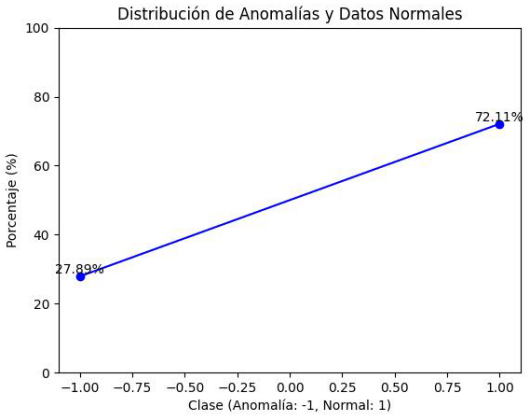
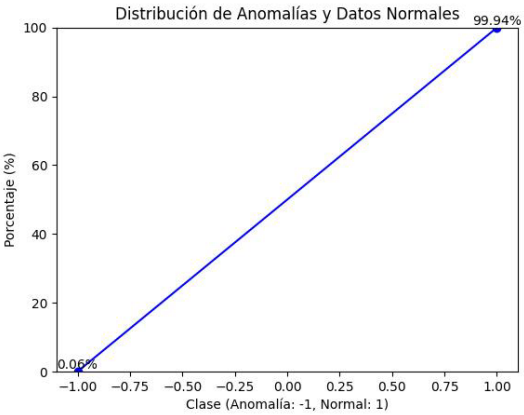
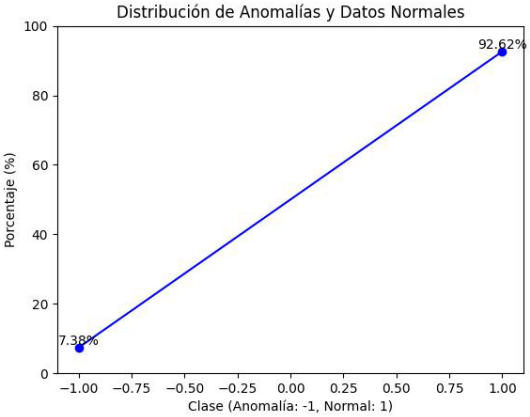
DBSCAN-RF	DBSCAN-SVM
	
KMEANS-RF	**KMEANS-SVM**
	

Tabla 5

Distribución de anomalías en producción (20405 registros)

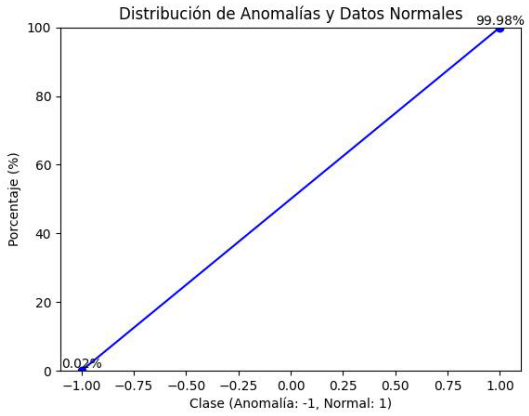
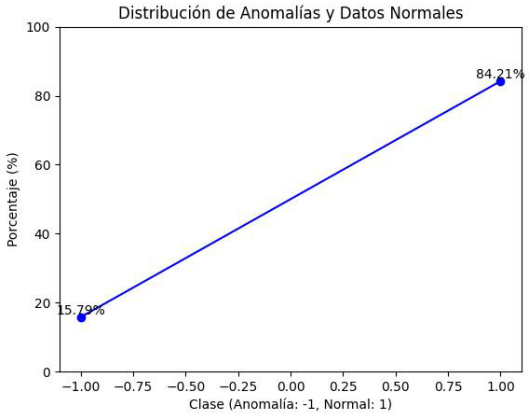
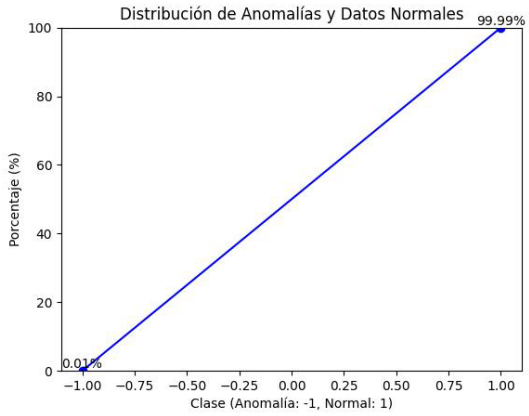
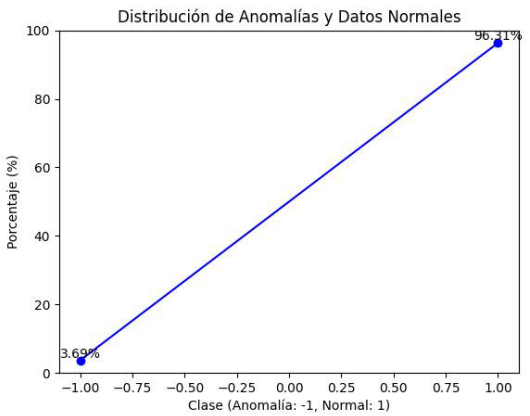
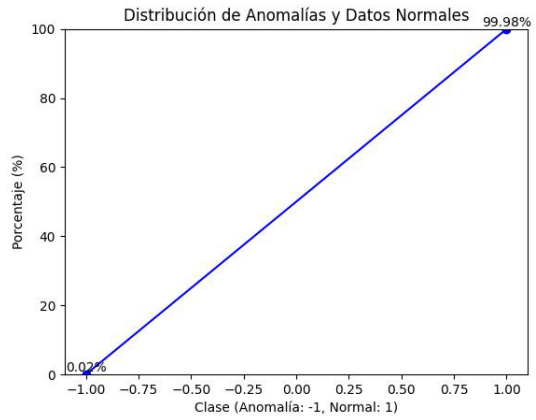
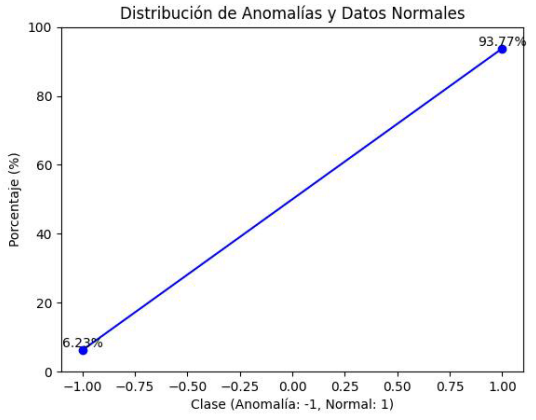
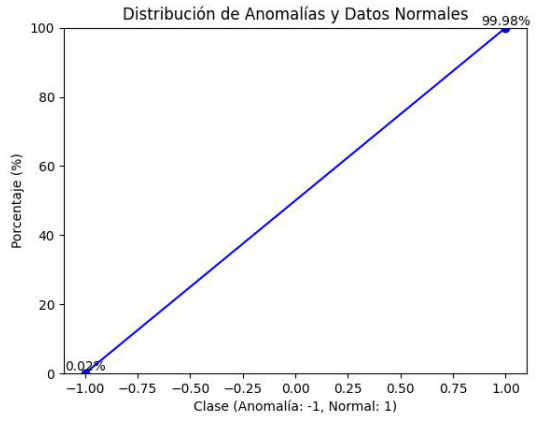
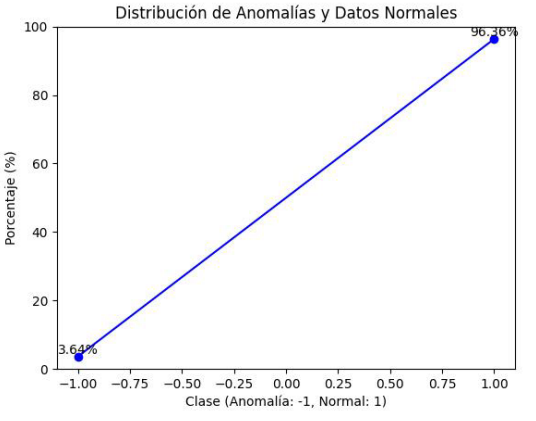
DBSCAN-RF	DBSCAN-SVM
	
KMEANS-RF	**KMEANS-SVM**
	

Tabla 6

Distribución de anomalías en producción (33086 registros)

DBSCAN-RF	DBSCAN-SVM
	
KMEANS-RF	**KMEANS-SVM**
	

Para realizar un análisis tomamos como base las gráficas de líneas de los algoritmos de entrenamientos para compararlos con la aplicación de los modelos en producción o con nuevos datos. Se puede notar que el modelo que analiza los nuevos datos con DBSCAN-RF o DBSCAN-SVM tienen problemas con datasets que tienen una cantidad baja de registros, pero a medida que aumentan los registros va tomando la misma forma que el entrenamiento.

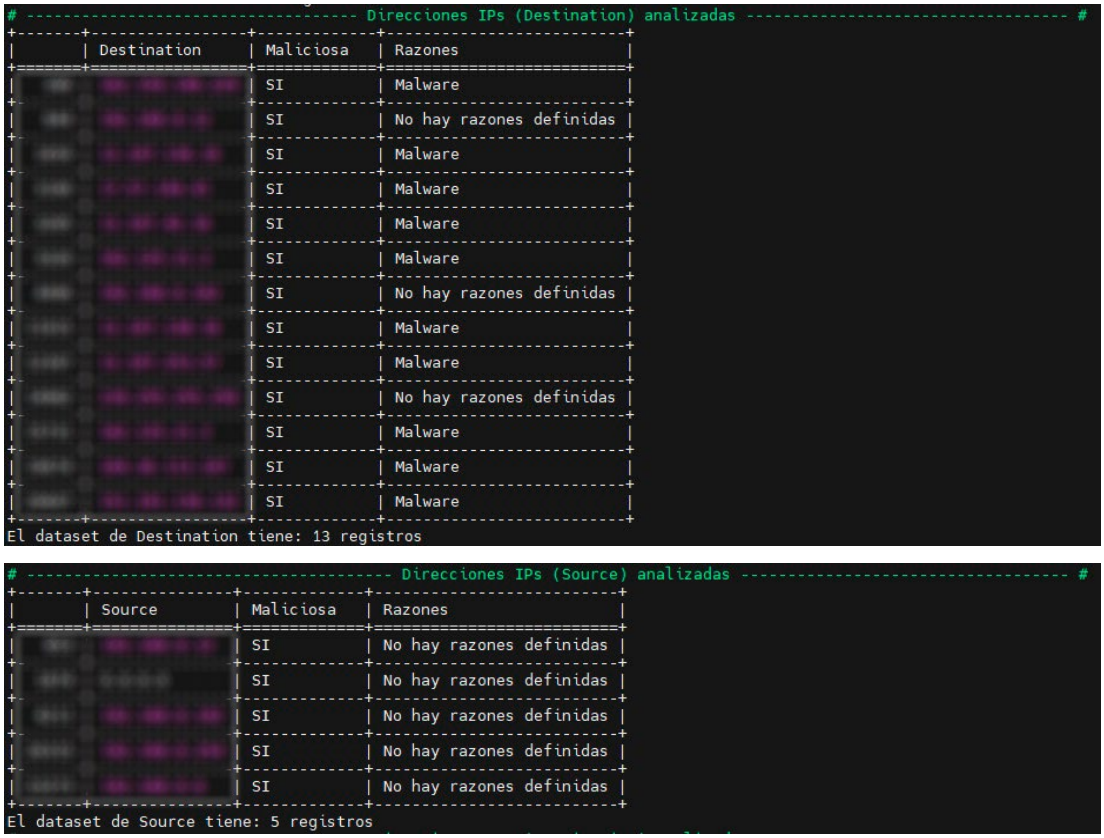
En cuanto a K-Means-RF se mantiene a lo largo de todas las pruebas y de las anomalías detectadas se puede corroborar que un porcentaje de ellos son detectados como amenazas reales por la API de Virus Total.

Finalmente analizando K-Means-SVM se puede notar que para datasets de pocos registros detecta casi la mitad de los datos como anomalías, pero con los datasets que tienen una cantidad de registros similar al dataset de entrenamiento, la línea es muy parecida a la obtenida en el entrenamiento.

Luego de analizar nuevos datos con los modelos entrenados se puede decir que el mejor modelo de detección es K-Means-SVM puesto que ha detectado la mayor cantidad de anomalías que son verificadas como amenazas a través de la API de virus total. A continuación, se muestra un listado con las IP que fueron detectadas como amenazas.

Figura 27

Listado de IPs analizadas con Virus Total



Para realizar este análisis se utilizó un servidor de las siguientes características 512 cores, 2TB RAM, 128TB de almacenamiento, y se obtuvo acceso al mismo gracias a un proyecto en colaboración de ESPOL y CEDIA. Cabe mencionar que no se obtuvo acceso a todos los recursos de este servidor, pero se pudo solicitar cierta parte de estos para realizar las tareas de procesamiento, con el siguiente comando:

salloc -p PARTITION -n 1 -c #CPUs --mem=#RAMGB --gres=gpu:GPUTypeConsole:#GPUs --time=hh:mm:ss

CONCLUSIONES

El uso de Machine Learning mejora significativamente la capacidad de identificación de amenazas en el tráfico de red de CENAIM-ESPOL.

La limpieza de los datos se realizó de tal manera que estos tuvieran la claridad y precisión necesarias para ser interpretados de manera correcta por los algoritmos de Machine Learning (ML), aumentando de esta manera la efectividad en la identificación de patrones maliciosos y también mejorando el rendimiento de los modelos a entrenar.

El modelo que mostró mayor eficacia en la detección de anomalías que resultaron ser maliciosas en el tráfico de red del CENAIM-ESPOL fue la combinación de los algoritmos K-Means-SVM. Con esto, se pueden tomar medidas para fortalecer la seguridad de la red.

Al integrar la API de Virus Total en el análisis de detección de anomalías se logró una validación más sólida de las amenazas detectadas. Con esto se espera aumentar el nivel de confianza en la detección de anomalías y una mayor cantidad de información sobre las amenazas que ayude a la hora de tomar decisiones relacionadas con la seguridad de la red.

RECOMENDACIONES

Establecer procesos para la limpieza de datos incluyendo pasos concretos que permitan asegurar su claridad y precisión. Se debe incluir el manejo de valores faltantes, la normalización de formatos, entre otros. Con esto se garantiza que el conjunto de datos utilizados en el análisis con los algoritmos de Machine Learning estén en condiciones óptimas, lo que mejorará la identificación de patrones maliciosos.

Generar un cronograma de pruebas periódicas y ajustes en el mejor modelo para adaptarlo a amenazas y patrones de tráfico de red que puedan surgir. También, se recomienda implementar un sistema de monitoreo continuo para detectar cambios en el comportamiento de la red, esto asegura que el modelo se mantenga entrenado, siendo así eficaz frente a la evolución de las amenazas.

Automatizar la integración de la API de Virus Total en el trabajo de análisis de anomalías. Esto permitirá agilizar y hacer más eficiente la validación de las amenazas detectadas, obteniendo así una respuesta más rápida ante posibles incidentes de seguridad y optimizando la toma de decisiones en la gestión de la seguridad de la red.

REFERENCIAS

Al-amri, R., Murugesan, R. K., Man, M., Abdulateef, A. F., Al-Sharafi, M. A., & Alkahtani, A. A. (2021). A Review of Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT Data. *Applied Sciences*, *11*(12), 5320. https://doi.org/10.3390/app11125320

Breiman, L. (2001). Bosques aleatorios. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Castillo Mendoza, J. I. (2022). *Análisis de los sistemas de detección de intrusos (IDS) Open Source y Software Propietario.* Babahoyo: UTB-FAFI. 2022.

Chóez Quimis, I. J., & Mero Suárez, K. V. (2023). Evaluación de diversas herramientas tecnológicas para el análisis de tráfico de la red de datos en la carrera de tecnología de la información. *Revista Científica Arbitrada Multidisciplinaria PENTACIENCIAS*, *5*(5), 275–288. https://doi.org/10.59169/pentaciencias.v5i5.734

Colque, M. S. S. I. J. (2021). NMAP COMO UNA HERRAMIENTA PARA LA SEGURIDAD DE REDES. *REVISTA CIENCIA Y TECNOLOGIA INFORMATICA*, *2*(2), 22–25.

Diro, A., Chilamkurti, N., Nguyen, V.-D., & Heyne, W. (2021). A Comprehensive Study of Anomaly Detection Schemes in IoT Networks Using Machine Learning Algorithms. *Sensors*, *21*(24), 8320. https://doi.org/10.3390/s21248320

Elmrabit, N., Zhou, F., Li, F., & Zhou, H. (2020). Evaluation of Machine Learning Algorithms for Anomaly Detection. *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 1–8. https://doi.org/10.1109/CyberSecurity49315.2020.9138871

Eltanbouly, S., Bashendy, M., AlNaimi, N., Chkirbene, Z., & Erbad, A. (2020). Machine Learning Techniques for Network Anomaly Detection: A Survey. *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 156–162. https://doi.org/10.1109/ICIoT48696.2020.9089465

Fan, J., Yue, W., Wu, L., Zhang, F., Cai, H., Wang, X., Lu, X., & Xiang, Y. (2018). Evaluation of SVM, ELM and four tree-based ensemble models for predicting daily reference evapotranspiration using limited meteorological data in different climates of China. *Agricultural and Forest Meteorology*, *263*, 225–241. https://doi.org/10.1016/j.agrformet.2018.08.019

Jain, G., & Anubha. (2021). Application of SNORT and Wireshark in Network Traffic Analysis. *IOP Conference Series: Materials Science and Engineering*, *1119*(1), 012007. https://doi.org/10.1088/1757-899X/1119/1/012007

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260. https://doi.org/10.1126/science.aaa8415

Josso, P., Hall, A., Williams, C., Le Bas, T., Lusty, P., & Murton, B. (2023). Application of random-forest machine learning algorithm for mineral predictive mapping of Fe-Mn crusts in the World Ocean. *Ore Geology Reviews*, *162*, 105671. https://doi.org/10.1016/j.oregeorev.2023.105671

López Tello, I. J. (2022). *Análisis y medición del tráfico en redes ip inalámbricas mediante herramientas prtg y mrtg*.

Martin Ramos, F. (2021). *Diseño de un Sistema de Detección de Intrusiones basado en Machine Learning para tráfico de red real*. https://oa.upm.es/69370/

Martínez Pérez, J. A., & Pérez Martin, P. S. (2023). La curva ROC. *SEMERGEN, Soc. Esp. Med. Rural Gen.(Ed. Impr.)*, e101821–e101821. https://pesquisa.bvsalud.org/portal/resource/pt/ibc-215631

Mustafa Abdullah, D., & Mohsin Abdulazeez, A. (2021). Machine Learning Applications based on SVM Classification A Review. *Qubahan Academic Journal*, *1*(2), 81–90. https://doi.org/10.48161/qaj.v1n2a50

Naoui Mohammed Anouar AND Lejdel, B. A. N. D. A. M. (2020). Using K-means algorithm for regression curve in big data system for business environment. *Revista Cubana de Ciencias InformÃ!`ticas*, *14*, 34–48.

http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992020000200034&nrm=iso

Nassif, A. B., Talib, M. A., Nasir, Q., & Dakalbab, F. M. (2021). Machine Learning for Anomaly Detection: A Systematic Review. *IEEE Access*, *9*, 78658–78700. https://doi.org/10.1109/ACCESS.2021.3083060

Pertuz, C. M. P. (2022). *Aprendizaje automático y profundo en python*. Ra-Ma Editorial. https://books.google.es/books?hl=es&lr=&id=NEi9EAAAQBAJ&oi=fnd&pg=PA7&dq=Aprendizaje+autom%C3%A1tico+y+profundo+en+python&ots=b3w2cyq7Ja&sig=6X2nTKjv3DGE37OjJusTr97lP_I

Poornima, I. G. A., & Paramasivan, B. (2020). Anomaly detection in wireless sensor network using machine learning algorithm. *Computer Communications*, *151*, 331–337. https://doi.org/10.1016/j.comcom.2020.01.005

Rahman, N. A. A., Sairi, I. H., Zizi, N. A. M., & Khalid, F. (2020). The Importance of Cybersecurity Education in School. *International Journal of Information and Education Technology*, *10*(5), 378–382. https://doi.org/10.18178/ijiet.2020.10.5.1393

Sáez Agud, J. Á. (2024). *Implementación de un servidor Nagios para la monitorización de servicios en red y la actividad de sus usuarios*. Universitat Politècnica de València.

Sandoval Serrano, L. J., & others. (2018). Algoritmos de aprendizaje automático para análisis y predicción de datos. *Revista Tecnológica; No. 11*. http://redicces.org.sv/jspui/handle/10972/3626

Shafapourtehrany, M., Yariyan, P., Özener, H., Pradhan, B., & Shabani, F. (2022). Evaluating the application of K-mean clustering in Earthquake vulnerability mapping of Istanbul, Turkey. *International Journal of Disaster Risk Reduction*, *79*, 103154. https://doi.org/10.1016/j.ijdrr.2022.103154

Singh, H. V., Girdhar, A., & Dahiya, S. (2022). A Literature survey based on DBSCAN algorithms. *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 751–758. https://doi.org/10.1109/ICICCS53718.2022.9788440

Toshniwal, A., Mahesh, K., & R., J. (2020). Overview of Anomaly Detection techniques in Machine Learning. *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 808–815. https://doi.org/10.1109/I-SMAC49090.2020.9243329

Vargas Machuca Del Salto, A. G. (2021). *Sistema de detección de intrusos basado en machine learning* [Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas.]. https://repositorio.uta.edu.ec/handle/123456789/34028

Wang, S., Balarezo, J. F., Kandeepan, S., Al-Hourani, A., Chavez, K. G., & Rubinstein, B. (2021). Machine Learning in Network Anomaly Detection: A Survey. *IEEE Access*, *9*, 152379–152396. https://doi.org/10.1109/ACCESS.2021.3126834

Wang, Y., Li, Y., Xie, H., Wu, B., & Yang, Y. (2022). Cluster division in wind farm through ensemble modelling. *IET Renewable Power Generation*, *16*(7), 1299–1315. https://doi.org/10.1049/rpg2.12276

AO Kaspersky Lab. (2024). *¿Qué es la ciberseguridad?* Retrieved from ¿Qué es la ciberseguridad?: https://latam.kaspersky.com/resource-center/definitions/what-is-cyber-security

Barrios Arce, J. (2019, Julio 26). *La matriz de confusión y sus métricas.* Retrieved from Health Big Data: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

CISCO. (2020, Marzo 09). *Cisco Annual Internet Report (2018–2023) White Paper*. Retrieved from Cisco Annual Internet Report (2018–2023) White Paper: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

IBM. (2021, 12 07). *SPSS Statistics*. Retrieved from Gráficos de líneas: https://www.ibm.com/docs/es/spss-statistics/beta?topic=types-line-charts

Listopro Community. (2024, Febrero 2). *Wireshark.* Retrieved from Wireshark: La herramienta indispensable para el análisis de redes:

https://community.listopro.com/wireshark-la-herramienta-indispensable-para-el-analisis-de-redes/

Malware-Traffic-Analysis.net. (2024, Septiembre 9). *Malware-Traffic-Analysis.net.* Retrieved from https://www.malware-traffic-analysis.net/2024/09/11/index.html

Malware-Traffic-Analysis.net. (2024, Octubre 3). *Malware-Traffic-Analysis.net.* Retrieved from 2024-10-03 (THURSDAY): SMARTLOADER TO LUMMA STEALER: https://www.malware-traffic-analysis.net/2024/10/03/index.html

Medium. (2024, Abril 21). *Classification Report Explained — Precision, Recall, Accuracy, Macro average, and Weighted Average.* Retrieved from https://medium-com.translate.goog/@chanakapinfo/classification-report-explained-precision-recall-accuracy-macro-average-and-weighted-average-8cd358ee2f8a?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=rq&_x_tr_hist=true

Microsoft. (2024). *Microsoft Power Automate*. Retrieved from Virus Total: https://powerautomate.microsoft.com/es-es/connectors/details/shared_virustotal/virus-total/

Microsoft. (2024). *Visual Studio.* Retrieved from https://visualstudio.microsoft.com/es/

VIRUSTOTAL. (2023). *VTDOC*. Retrieved from Cómo funciona: https://docs.virustotal.com/docs/how-it-works

ANEXOS

Anexo 1: Solicitud aprobada por el director, para ejecutar el tema de tesis en CENAIM-ESPOL

San Pedro, 02 de agosto del 2024

Stanislaus Sonnenholzner. Ph. D.
Director del CENAIM-ESPOL

En su despacho. –

Reciba un cordial saludo y a la vez deseándole éxitos en sus labores diarias, por este medio permitame expresar lo siguiente:

Yo, Ing. Francisco Alexis Chóez Baque, con número de cedula 0923317440, laborando como Analista de Soporte Técnico en nuestra institución CENAIM-ESPOL, actualmente estoy realizando mis estudios de cuarto nivel en la Universidad Estatal Península de Santa Elena (UPSE), por lo tanto, mi petición es que me permita realizar mi trabajo de titulación denominado: **"IDENTIFICACION DE ANOMALIAS EN EL TRAFICO DE RED UTILIZANDO ALGORITMOS DE APRENDIZAJE AUTOMATICO. CASO DE ESTUDIO CENAIM-ESPOL".**

Previo a la obtención del título de Máster en Ciberseguridad.

Por la atención que brinde al presente quedo muy agradecido.

Atentamente,

Ing. Francisco Chóez Baque
Analista de Soporte Técnico
CENAIM-ESPOL

2-ago-2024

Anexo 2: Librerías utilizadas en Phyton

```
from dash import Dash, html, dcc, Input, Output, State, dash_table, no_update
import dash
import dash_bootstrap_components as dbc
import pandas as pd
import base64, os, io
from datetime import datetime
from werkzeug.utils import secure_filename
from waitress import serve
```

Anexo 3: Código fuente del dashboard

```
app = Dash(__name__, external_stylesheets=[
    dbc.themes.BOOTSTRAP,
    "https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css"
], suppress_callback_exceptions=True)

# Define el encabezado con título y logotipos
navbar = dbc.Navbar(
    dbc.Container([
        # Logotipo espol
        html.Img(src="/assets/img/logo_espol.png", height="100px"), #height="100px"

        # Título central
        html.P("Detección de anomalías", className="mx-auto h1", style={"color":"white"}),

        # Logotipo upse
        html.Img(src="/assets/img/logo_upse.png", height="100px") #className="img-fluid"
    ]),
    color="primary",  # Color de fondo
    dark=True,  # Estilo de texto claro
    className="mb-4"
)
```

Anexo 4: Código fuente del dashboard

```
# Definir el panel lateral como un componente Offcanvas
sidebar = dbc.Offcanvas(
    [
        html.H5("Menú Principal", className="display-6"),
        html.Hr(),

        # Primer menú (Dataset)
        dbc.Button(
            "Dataset",
            id="dataset-collapse",
            className="mb-2",
            color="primary",
            style={"width": "100%"},
        ),
        dbc.Collapse(
            dbc.Nav(
                [
                    dbc.NavLink("Generar Dataset", href="#", id="link-generar-dataset"),
                    dbc.NavLink("Cargar Dataset", href="#", id="link-cargar-dataset"),
                ],
                vertical=True,
                pills=True,
            ),
            id="collapse-dataset",
        ),

        html.Hr(),
```

```
        html.Hr(),

        # Segundo menú (Análisis)
        dbc.Button(
            "Analizar",
            id="toggle-production-collapse",
            className="mb-2",
            color="primary",
            style={"width": "100%"}
        ),
        dbc.Collapse(
            dbc.Nav(
                [
                    dbc.NavLink("DBSCAN-RF", href="#", id="link-dbscan-rf-prod"),
                    dbc.NavLink("DBSCAN-SVM", href="#", id="link-dbscan-svm-prod"),
                    dbc.NavLink("K-MEANS-RF", href="#", id="link-kmeans-rf-prod"),
                    dbc.NavLink("K-MEANS-SVM", href="#", id="link-kmeans-svm-prod"),
                ],
                vertical=True,
                pills=True,
            ),
            id="collapse-production",
        ),
    ],
    id="offcanvas-sidebar",
    #title="Panel Lateral",
    is_open=False,
    backdrop=True,
```

Anexo 5: Código fuente del dashboard

```
# Layout de la app
app.layout = html.Div([
    # Usamos html.Button en lugar de dbc.Button
    navbar, # Encabezado
    html.Button(html.I(className="fas fa-bars"),
                id="open-offcanvas",
                n_clicks=0,
                className="btn btn-primary mb-2"),  # Añadir las clases de Bootstrap
    sidebar,
    html.Div(id="content", className="p-4"), # Para contenido normal
])

# Callbacks para manejar la visibilidad de Offcanvas y Collapse
@app.callback(
    Output("offcanvas-sidebar", "is_open"),
    Input("open-offcanvas", "n_clicks"),
    [State("offcanvas-sidebar", "is_open")],
)
def toggle_offcanvas(n_clicks, is_open):
    if n_clicks:
        return not is_open
    return is_open


# Callback abrir y cerrar submenus
@app.callback(
    [Output("collapse-production", "is_open"),
     Output("collapse-dataset", "is_open")],
    [Input("toggle-production-collapse", "n_clicks"),
     Input("dataset-collapse", "n_clicks")],
    [State("collapse-production", "is_open"),
     State("collapse-dataset", "is_open")]
)
def toggle_submenus(produccion_clicks, dataset_clicks, is_production_open, is_dataset_open):
    # Determinar cuál botón fue presionado
    triggered_id = dash.callback_context.triggered[0]['prop_id'].split('.')[0]

    # Lógica para alternar el estado del submenú presionado y cerrar los otros

    if triggered_id == "toggle-production-collapse":
        return not is_production_open, False
    elif triggered_id == "dataset-collapse":
        return False, not is_dataset_open

    # Si no se presionó ningún botón, retornar los estados actuales
    return is_production_open, is_dataset_open
```

Anexo 6: Código fuente del dashboard

```
# ------------------------------ Para seleccionar los submenus ---------------------------------------- #
# Callback para selección de submenú
@app.callback(
    Output("content", "children"),
    [Input("link-dbscan-rf-prod", "n_clicks"),
     Input("link-dbscan-svm-prod", "n_clicks"),
     Input("link-kmeans-rf-prod", "n_clicks"),
     Input("link-kmeans-svm-prod", "n_clicks"),
     Input("link-generar-dataset", "n_clicks"),
     Input("link-cargar-dataset", "n_clicks")]
)
def display_table(*args):
    # Identificar qué enlace fue presionado
    triggered_id = dash.callback_context.triggered[0]['prop_id'].split('.')[0]

    # Para submenu de dataset
    if triggered_id == "link-generar-dataset":
        # Lógica para la opción de Generar Dataset
        return dbc.Card([
            dbc.CardHeader("Generar Dataset desde archivo .pcapng o .pcap"),
            dbc.CardBody([
                dcc.Upload(
                    id="upload-pcapng",
                    children=dbc.Button("Cargar nuevo archivo .pcapng o .pcap", color="primary"),
                    multiple=False
                ),
                html.Div(id="file-name-display", className="mt-2"),  # Aquí se mostrará el nombre del archivo
                html.Br(),
                html.Label("Nombre del Dataset:"),
                dbc.Input(id="dataset-name", type="text", placeholder="Escriba un nombre para el dataset"),
```

```
    elif triggered_id == "link-cargar-dataset":
        # Lógica para la opción de Cargar Dataset
        return dbc.Card([
            dcc.Store(id="initialize-table", data=True),  # Almacén para inicializar tabla
            dbc.CardHeader("Cargar CSV"),
            dbc.CardBody([
                dcc.Upload(
                    id="upload-csv",
                    children=dbc.Button("Cargar archivo .csv", color="primary"),
                    multiple=False,
                    accept=".csv" # solo .csv
                ),
                html.Div(id="file-name-display-csv", className="mt-2"),  # Aquí se muestra el nombre del archivo
                html.Br(),
                dbc.Button("Guardar CSV", id="save-csv", color="success", n_clicks=0),
                html.Div(id="save-message-csv", className="mt-3"),
                html.Br(),
                # DataTable para mostrar los archivos en dataset-prod
                dbc.Table(id="file-table", striped=True, bordered=True, hover=True)
            ])
        ])
```

Anexo 8: Código fuente del dashboard

```
    elif triggered_id == "link-dbscan-svm-prod":
        return dbc.Card([
            #prod_dbscan_rf()
            dbc.CardHeader("Producción DBSCAN-SVM"),
            dbc.CardBody([
                # Dropdown para seleccionar el dataset
                dbc.Label("Selecciona un dataset:", html_for="dataset-dropdown-db-sv"),
                dcc.Dropdown(
                    id="dataset-dropdown-db-sv",
                    options=[{"label": f, "value": f} for f in listar_csv("dataset-prod")],
                    placeholder="Selecciona un archivo .csv"
                ),
                html.Br(),
                dbc.Button("Procesar Dataset con DBSCAN-SVM", id="procesar-dataset-btn-db-sv", color="primary"),
                html.Br(),
                html.Div(id="dataset-info-db-sv", className="mt-3"),  # Contenedor para mensajes
                html.Br(),
                html.Div(id="img-lineas-db-sv", className="mt-3"), # Contenedor para gráfico de lineas
                html.Br(),
                # Contenedor para mostrar la tabla con IPs Source
                html.Div(id="tabla-dataset-source-db-sv", className="mt-3"),
                html.Br(),
                # Contenedor para mostrar la tabla con IPs Destination
                html.Div(id="tabla-dataset-dest-db-sv", className="mt-3")
            ])
        ])
```

```
    elif triggered_id == "link-kmeans-rf-prod":
        return dbc.Card([
            dbc.CardHeader("Producción Kmeans-RF"),
            dbc.CardBody([
                # Dropdown para seleccionar el dataset
                dbc.Label("Selecciona un dataset:", html_for="dataset-dropdown-km-rf"),
                dcc.Dropdown(
                    id="dataset-dropdown-km-rf",
                    options=[{"label": f, "value": f} for f in listar_csv("dataset-prod")],
                    placeholder="Selecciona un archivo .csv"
                ),
                html.Br(),
                dbc.Button("Procesar Dataset con DBSCAN-SVM", id="procesar-dataset-btn-km-rf", color="primary"),
                html.Br(),
                html.Div(id="dataset-info-km-rf", className="mt-3"),  # Contenedor para mensajes
                html.Br(),
                html.Div(id="img-lineas-km-rf", className="mt-3"), # Contenedor para gráfico de lineas
                html.Br(),
                # Contenedor para mostrar la tabla con IPs Source
                html.Div(id="tabla-dataset-source-km-rf", className="mt-3"),
                html.Br(),
                # Contenedor para mostrar la tabla con IPs Destination
                html.Div(id="tabla-dataset-dest-km-rf", className="mt-3")
            ])
        ])
```

Anexo 9: Código fuente del dashboard

```
elif triggered_id == "link-kmeans-svm-prod":
    return dbc.Card([
        dbc.CardHeader("Producción KMeans-SVM"),
        dbc.CardBody([
            # Dropdown para seleccionar el dataset
            dbc.Label("Selecciona un dataset:", html_for="dataset-dropdown-km-sv"),
            dcc.Dropdown(
                id="dataset-dropdown-km-sv",
                options=[{"label": f, "value": f} for f in listar_csv("dataset-prod")],
                placeholder="Selecciona un archivo .csv"
            ),
            html.Br(),
            dbc.Button("Procesar Dataset con DBSCAN-SVM", id="procesar-dataset-btn-km-sv", color="primary"),
            html.Br(),
            html.Div(id="dataset-info-km-sv", className="mt-3"),  # Contenedor para mensajes
            html.Br(),
            html.Div(id="img-lineas-km-sv", className="mt-3"), # Contenedor para gráfico de lineas
            html.Br(),
            # Contenedor para mostrar la tabla con IPs Source
            html.Div(id="tabla-dataset-source-km-sv", className="mt-3"),
            html.Br(),
            # Contenedor para mostrar la tabla con IPs Destination
            html.Div(id="tabla-dataset-dest-km-sv", className="mt-3")
        ])
    ])
# Si no se hace clic en ninguna opción, mostrar un mensaje
return html.Div("Selecciona una opción del Menú.", className="alert alert-info")
```

Anexo 10: Código fuente funciones – Leer archivo .pcapng y generar DataFrame

```
# ---------------------- Leer archivo ------------------------------ #
def leer_pcapng(p):
    # Abrir el archivo pcapng con PcapReader

    packets = PcapReader('archivos1/2024-09-11-XLoader-infection-traffic.pcap')

    # Listas para almacenar la información extraída
    data = []
    #count = 1

    # Recorrer todos los paquetes
    for pkt in packets:
        # if count > max_packets:
        #     break
        if IP in pkt:
            # Variables comunes
            source = pkt[IP].src
            destination = pkt[IP].dst
            protocol_name = pkt[IP].proto
            length = len(pkt)
            payload = len(bytes(pkt[IP].payload))

            # Inicialización por defecto
            sport = None
            dport = None
```

```
            if TCP in pkt:
                sport = pkt[TCP].sport
                dport = pkt[TCP].dport
            elif UDP in pkt:
                sport = pkt[UDP].sport
                dport = pkt[UDP].dport
            else:
                sport = 0
                dport = 0

            # Agregar datos a la lista
            data.append({
                'Source': source,
                'Destination': destination,
                'Protocol': protocol_name,
                'Length': length,
                'Payload': payload,
                'Port Source': sport,
                'Port Destination': dport
            })
        #count += 1

    # Crear DataFrame con pandas
    df = pd.DataFrame(data)
    df_muestra = df.sample(frac=p)

    # Cerrar el archivo pcapng
    packets.close()

    return df_muestra
```

Anexo 11: Código fuente de los algoritmos

```
# ---------------------------------- DBSCAN y RANDOM FOREST -------------------------------------------
def ejecutar_dbscan_rf(df):
    df_encoded = df
    # 2.- Codificar las columnas categóricas (Source, Destination, Protocol)
    # Utilizamos LabelEncoder para convertir IPs y Protocolos a números
    le = LabelEncoder()

    df_encoded['Source'] = le.fit_transform(df_encoded['Source'])
    df_encoded['Destination'] = le.fit_transform(df_encoded['Destination'])

    # 3.- Seleccionar las características numéricas para el clustering
    # Por ejemplo: Source, Destination, Protocol, Length, Payload, Port Source, Port Destination
    X = df_encoded[['Source', 'Destination', 'Protocol', 'Length', 'Payload', 'Port Source', 'Port Destination']]

    # 4.- Escalar los datos
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # 5.- Aplicar DBSCAN
    dbscan = DBSCAN(eps=0.5, min_samples=5)  # Ajusta los parámetros
    labels = dbscan.fit_predict(X_scaled)

    # 6.- Agregar las etiquetas al DataFrame copia
    df_encoded['Anomalia'] = labels

    # Ver los resultados
    #print(df.head())

    # 7.- Asignar todas las etiquetas que no son -1 a 0
    df_encoded['Anomalia'] = df_encoded['Anomalia'].apply(lambda x: 0 if x != -1 else -1)
```

Anexo 12: Código fuente de los algoritmos

```python
# ---------------------------------- DBSCAN y SVM -------------------------------------------
def ejecutar_dbscan_svm(df):
    df_encoded = df
    # 2.- Codificar las columnas categóricas (Source, Destination, Protocol)
    # Utilizamos LabelEncoder para convertir IPs y Protocolos a números
    le = LabelEncoder()

    df_encoded['Source'] = le.fit_transform(df_encoded['Source'])
    df_encoded['Destination'] = le.fit_transform(df_encoded['Destination'])

    # 3.- Seleccionar las características numéricas para el clustering
    X = df_encoded[['Source', 'Destination', 'Protocol', 'Length', 'Payload', 'Port Source', 'Port Destination']]

    # 4.- Escalar los datos
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # 5.- Aplicar DBSCAN
    dbscan = DBSCAN(eps=0.5, min_samples=5)  # Ajusta los parámetros
    labels = dbscan.fit_predict(X_scaled)

    # 6.- Agregar las etiquetas al DataFrame copia
    df_encoded['Anomalia'] = labels

    # 7.- Asignar todas las etiquetas que no son -1 a 0
    df_encoded['Anomalia'] = df_encoded['Anomalia'].apply(lambda x: 0 if x != -1 else -1)

    # 8.- Filtrar los puntos que se consideran anomalías (DBSCAN los etiqueta como -1)
    anomalias = df_encoded[df_encoded['Anomalia'] == -1]
    print(f'Número de anomalías detectadas: {len(anomalias)}')
```

Anexo 13: Código fuente de los algoritmos

```python
# ---------------------------------- K-MEANS y RANDOM FOREST-------------------------------------
def ejecutar_kmeans_rf(df):
    # 2.- Codificar las columnas categóricas (Source, Destination, Protocol)
    # Utilizamos LabelEncoder para convertir IPs y Protocolos a números
    le = LabelEncoder()

    df_encoded = df.copy()

    df_encoded['Source'] = le.fit_transform(df_encoded['Source'])
    df_encoded['Destination'] = le.fit_transform(df_encoded['Destination'])
    #df_encoded['Protocol'] = le.fit_transform(df_encoded['Protocol'])

    # 3.- Seleccionar las características numéricas para el clustering
    # Por ejemplo: Source, Destination, Protocol, Length, Payload, Port Source, Port Destination
    X = df_encoded[['Source', 'Destination', 'Protocol', 'Length', 'Payload', 'Port Source', 'Port Destination']]

    # 4.- Escalar los datos
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # 5.- Definir el modelo de K-Means con el número de clusters (k)
    kmeans = KMeans(n_clusters=8, random_state=42)

    # 6.- Entranar el modelo
    kmeans.fit(X_scaled)

    # 7.- Obtener las etiquetas de los clusters
    labels = kmeans.labels_

    # 8.- Calcula las distancias de cada punto a su centroide
    centroids = kmeans.cluster_centers_
```

Anexo 14: Código fuente de los algoritmos

```
# ------------------------------------ K-MEANS y SVM-------------------------------------
def ejecutar_kmeans_svm(df):
    # 2.- Codificar las columnas categóricas (Source, Destination)
    # Utilizamos LabelEncoder para convertir IPs a números
    le = LabelEncoder()

    df_encoded = df.copy()

    df_encoded['Source'] = le.fit_transform(df_encoded['Source'])
    df_encoded['Destination'] = le.fit_transform(df_encoded['Destination'])

    # 3.- Seleccionar las características numéricas para el clustering
    X = df_encoded[['Source', 'Destination', 'Protocol', 'Length', 'Payload', 'Port Source', 'Port Destination']]

    # 4.- Escalar los datos
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # 5.- Definir el modelo de K-Means con el número de clusters (k)
    kmeans = KMeans(n_clusters=8, random_state=42)

    # 6.- Entranar el modelo
    kmeans.fit(X_scaled)

    # 7.- Obtener las etiquetas de los clusters
    labels = kmeans.labels_

    # 8.- Calcula las distancias de cada punto a su centroide
    centroids = kmeans.cluster_centers_

    # 9.- Crea un array con las distancias de cada punto al centroide de su cluster
    distances = np.linalg.norm(X_scaled - centroids[labels], axis=1)
```

Anexo 15: Código fuente de los algoritmos – Función IPs anómalas

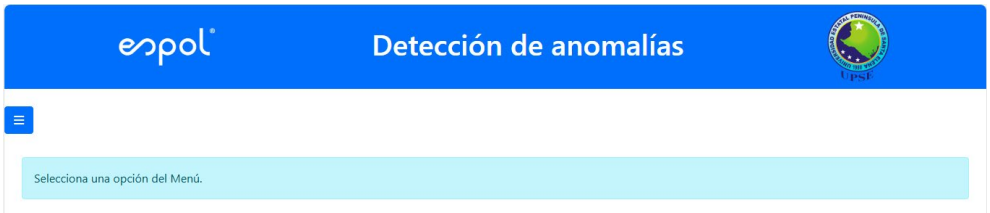
```
# --------------------------------- Identificar IPs con anomalias ----------------------------------------
def ips_anomalias(df,alg):
    ## -------------------- Ver IP's anomalas ------------------------- ##

    # 15.- Seleccionar las IP's que el modelo predijo que eran anomalías
    # DataFrame original es 'df' con la columna 'Anomalia'
    #print(df.head())

    # Copiar el dataframe
    df_encoded = df.copy()

    if alg == "dbscan-rf":
        X_test, y_pred = ejecutar_dbscan_rf(df)
    if alg == "kmeans-rf":
        X_test, y_pred = ejecutar_kmeans_rf(df)
    if alg == "dbscan-svm":
        X_test, y_pred = ejecutar_dbscan_svm(df)
    if alg == "kmeans-svm":
        X_test, y_pred = ejecutar_kmeans_svm(df)

    # 16.- Filtrar las filas del conjunto de prueba que fueron predichas como anomalías
    # (recordando que estas son las que tienen '-1')
    anomalias_indices = X_test.index[y_pred == -1]

    # Filtrar las filas que fueron predichas como anomalías (-
    anomalias_ips = df_encoded.loc[anomalias_indices, ['Source', 'Destination']]

    # Si hay mas de 0 anomalias procesar, caso contrario mandar el mensaje con 0 anomalias
    if (anomalias_ips.shape[0] == 0 ):
        print(f"El modelo ha detectado {len(anomalias_ips)} IPs como anomalías.")
        print("Saliendo del programa por que no hay anomalías que analizar")
        sys.exit()
```

Anexo 16: Código fuente de los algoritmos – Función Virus Total

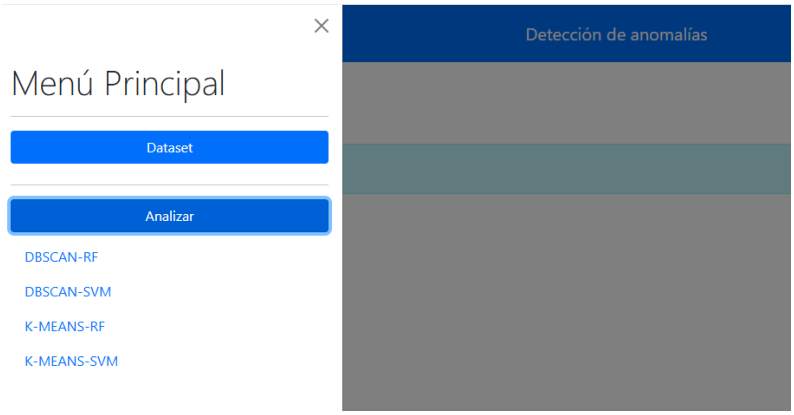
```python
# ---------------------------------- Analizar con Virus Totala las IPs con anomalias -----------------
def vt(df,alg):
    # ---------------- Analizar IP's con API Virus Total ---------------------------------------
    anomalias_ips, df_encoded = ips_anomalias(df,alg)
    df = df_encoded

    # Dataframe ip anomalas
    df_ips = anomalias_ips
    df_ips_source = df_ips.loc[(df_ips['Source'] != ""), ['Source']]
    df_ips_destination = df_ips.loc[(df_ips['Destination'] != ""), ['Destination']]

    def consultar_virustotal(ip_address):
        API_KEY =
        url = f'https://www.virustotal.com/api/v3/ip_addresses/{ip_address}'
        headers = {
            'x-apikey': API_KEY
        }

        try:
            # Hacemos la solicitud GET a la API
            response = requests.get(url, headers=headers)
            time.sleep(15) # esperar 15 segundos
            if response.status_code == 200:
                data = response.json()
                malicious = "NO"
                reasons = []

                if 'data' in data and 'attributes' in data['data']:
                    attributes = data['data']['attributes']
                    last_analysis = attributes.get('last_analysis_results', {})
```

Anexo 17: Ventana de Login

Anexo 18: Ventana principal del Dashboard



Anexo 19: Menú del Dashboard



Anexo 20: Resultados de los algoritmos con IP maliciosas

IP	Maliciosa	Razones
192.XXX.XXX.XXX	SI	No hay razones definidas
0.XXX.XXX.XXX	SI	No hay razones definidas
192.XXX.XXX.XXX	SI	No hay razones definidas
192.XXX.XXX.XXX	SI	No hay razones definidas
162.XXX.XXX.XXX	SI	Malware
192.XXX.XXX.XXX	SI	No hay razones definidas

IP	Maliciosa	Razones
181.XXX.XXX.XXX	SI	No hay razones definidas
162.XXX.XXX.XXX	SI	Malware
192.XXX.XXX.XXX	SI	No hay razones definidas
13.XXX.XXX.XXX	SI	Malware
17.XXX.XXX.XXX	SI	Malware
13.XXX.XXX.XXX	SI	Malware
162.XXX.XXX.XXX	SI	Malware
192.XXX.XXX.XXX	SI	No hay razones definidas
13.XXX.XXX.XXX	SI	Malware
13.XXX.XXX.XXX	SI	Malware
239.XXX.XXX.XXX	SI	No hay razones definidas
162.XXX.XXX.XXX	SI	Malware
185.XXX.XXX.XXX	SI	No hay razones definidas
104.XXX.XXX.XXX	SI	Malware

Anexo 21: Acuerdo de Confidencialidad

ACUERDO DE CONFIDENCIALIDAD

ACUERDO DE CONFIDENCIALIDAD Y NO DIVULGACIÓN DE INFORMACIÓN, QUE CELEBRAN POR UNA PARTE **CHOEZ BAQUE FRANCISO ALEXIS,** CON NÚMERO DE CÉDULA 0923317440, A QUIEN EN LO SUCESIVO SE DENOMINARÁ **"EL RECEPTOR"** Y POR LA OTRA, LA INSTITUCIÓN ***ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL (ESPOL)***, A QUIEN EN LO SUCESIVO SE LE DENOMINARÁ **"EL DIVULGANTE"** REPRESENTADA EN ESTE ACTO POR *ING. JUAN PABLO HIDALGO VILLACRES.*, AL TENOR DE LAS DECLARACIONES Y CLÁUSULAS SIGUIENTES:

PRIMERA.- Objeto. El presente Acuerdo se refiere a la información que EL DIVULGANTE proporcione al RECEPTOR, ya sea de forma oral, digital, gráfica o escrita y, en estos dos últimos casos, ya esté contenida en cualquier tipo de documento, para la ejecución de un proyecto de tesis o investigación científica que servirá como requisito para obtener el título de Máster en Ciberseguridad.

SEGUNDA.- 1. EL RECEPTOR únicamente utilizará la información facilitada por EL DIVULGANTE para el fin mencionado en la Estipulación anterior, comprometiéndose EL RECEPTOR a mantener la más estricta confidencialidad respecto de dicha información, advirtiendo de dicho deber de confidencialidad y a cualquier persona que, por su relación con EL RECEPTOR, deba tener acceso a dicha información para el correcto cumplimiento de las obligaciones del RECEPTOR para con EL DIVULGANTE.

2. EL RECEPTOR y/o las personas mencionadas en el párrafo anterior se comprometen a no reproducir, modificar, hacer pública, compartir o divulgar a terceros la información objeto del presente Acuerdo sin previa autorización escrita y expresa del DIVULGANTE. Esto incluye la data facilitada que al procesarla se convierte en un dataset para un entorno educativo-investigativo en la que, de haber una presentación del proyecto o investigación, solo se mostrarán los resultados de este de forma pública y la información fuente seguirá siendo confidencial. Los resultados mostrados podrán ser utilizados como apoyo para otras actividades investigativas y científicas.

3. De igual forma, EL RECEPTOR adoptará medidas de seguridad técnicas para proteger la información objeto de este Acuerdo, incluyendo prevención y respuesta ante incidentes de ciberseguridad como pérdida, robo, sustracción o alteración.

TERCERA.- Sin perjuicio de lo estipulado en el presente Acuerdo, ambas partes aceptan que la obligación de confidencialidad no se aplicará en los siguientes casos:

a) Cuando la información se encontrará en el dominio público en el momento de su suministro al RECEPTOR o, una vez suministrada la información, ésta acceda al dominio público sin infracción de ninguna de las Estipulaciones del presente Acuerdo.

b) Cuando la información ya estuviera en el conocimiento del RECEPTOR con anterioridad a la firma del presente Acuerdo y sin obligación de guardar confidencialidad.

c) Cuando la legislación vigente o un mandato judicial exija su divulgación. En ese caso, EL RECEPTOR notificará al DIVULGANTE tal eventualidad y hará todo lo posible por garantizar que se dé un tratamiento confidencial a la información.

d) En caso de que EL RECEPTOR pueda probar que la información fue desarrollada o recibida legítimamente de terceros, de forma totalmente independiente a su relación con EL DIVULGANTE.

CUARTA.- Los derechos de propiedad intelectual de la información objeto de este Acuerdo pertenecen al DIVULGANTE y el hecho de revelarla al RECEPTOR para el fin mencionado en la Estipulación Primera no cambiará tal situación.

En caso de que la información resulte revelada o divulgada o utilizada por EL RECEPTOR de cualquier forma distinta al objeto de este Acuerdo, ya sea de forma dolosa o por mera negligencia, habrá de indemnizar al DIVULGANTE los daños y perjuicios ocasionados, sin perjuicio de las acciones civiles o penales que puedan corresponder a este último.

QUINTA.- Las partes se obligan a devolver cualquier documentación, antecedentes facilitados en cualquier tipo de soporte y, en su caso, las copias obtenidas de los mismos, que constituyan información amparada por el deber de confidencialidad objeto del presente Acuerdo en el momento que se dé por terminado el proyecto por cualquier motivo.

SEXTA.- El presente Acuerdo entrará en vigor en el momento de la firma del mismo por ambas partes, extendiéndose su vigencia hasta un plazo de 1 año después de finalizado el proyecto.

SÉPTIMA.- En caso de cualquier conflicto o discrepancia que pueda surgir en relación con la interpretación y/o cumplimiento del presente Acuerdo, las partes se someten expresamente a los Juzgados y Tribunales, con renuncia a su fuero propio, aplicándose la legislación vigente.

Y en señal de expresa conformidad y aceptación de los términos recogidos en el presente Acuerdo, lo firman las partes en el lugar y fecha al indicados.

Guayaquil, 09 de diciembre de 2024

POR EL RECEPTOR	**POR EL DIVULGANTE**
	
Ing. Francisco Chóez Baque **Analista de Soporte Técnico** **CENAIM-ESPOL**	Ing. Juan Pablo Hidalgo **Director de Seguridad y** **Control de Sistemas**