Clasificación de ataques mediante técnicas de aprendizaje automático en Redes Definidas por Software

Castillo Camacho, Miguel Angel

Departamento de Ciencias de la Computación

Carrera de Tecnologías de la Información

Trabajo de Integración Curricular, previo a la obtención de título de Ingeniero/a en Tecnologías de la Información

Ing. Núñez Agurto, Alberto Daniel, Mgtr

24 de febrero de 2023

Reporte de verificación de contenido

CERTIFICADO DE ANÁLISIS
magister

SDN_Attack_TC_Tesis

1% Similitudes

< 1% Texto entre comillas
0% similitudes entre comillas
0% Idioma no reconocido

Nombre del documento: SDN_Attack_TC_Tesis.pdf
ID del documento: 1d3f391fc5ec9f04e0391f662a46a9c3c3ec4034
Tamaño del documento original: 4,21 Mo

Depositante: ALBERTO DANIEL NÚÑEZ AGURTO
Fecha de depósito: 26/2/2023
Tipo de carga: interface
fecha de fin de análisis: 27/2/2023

Número de palabras: 20.972
Número de caracteres: 127.315

Ubicación de las similitudes en el documento:

Fuentes principales detectadas

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	**doi.org** | SD-Honeypot Integration for Mitigating DDoS Attack Using Machine Learnin... https://doi.org/10.30630/joiv.6.1.853	< 1%		Palabras idénticas : < 1% (46 palabras)
2	**repositorio.espe.edu.ec** | Mapeo sistemático de literatura asequibles y reproducible... http://repositorio.espe.edu.ec:8080/bitstream/21000/20972/5/T-ESPE-038877.pdf.txt	< 1%		Palabras idénticas : < 1% (21 palabras)

Fuentes con similitudes fortuitas

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	**doi.org** | An intelligent flow-based and signature-based IDS for SDNs using ensembl... https://doi.org/10.3233/JIFS-200850	< 1%		Palabras idénticas : < 1% (28 palabras)
2	**www.semanticscholar.org** | [PDF] Bandwidth Control Mechanism and Extreme Gradi... https://www.semanticscholar.org/paper/Bandwidth-Control-Mechanism-and-Extreme-Gradient-Alamri...	< 1%		Palabras idénticas : < 1% (25 palabras)
3	**doi.org** | Distributed denial of service attacks detection for software defined network... https://doi.org/10.11591/eei.v11i4.3835	< 1%		Palabras idénticas : < 1% (16 palabras)
4	**repositorio.espe.edu.ec** https://repositorio.espe.edu.ec/bitstream/21000/32783/1/T-ESPE-052531.pdf	< 1%		Palabras idénticas : < 1% (15 palabras)
5	**diagramasuml.com** | ¿Qué es el KDD o Proceso de descubrimiento de conocimiento? https://diagramasuml.com/que-es-el-kdd-o-proceso-de-descubrimiento-de-conocimiento/	< 1%		Palabras idénticas : < 1% (15 palabras)

Fuentes mencionadas (sin similitudes detectadas)

Estas fuentes han sido citadas en el documento sin encontrar similitudes.

1 https://www.unb.ca/cic/datasets/nsl.html
2 https://www.stratosphereips.org/datasets-ctu13
3 https://onlineacademiccommunity.uvic.ca/isot/datasets/
4 https://data.mendeley.com/datasets/jxpfjc64kr/1
5 https://www.unb.ca/cic/datasets/ids.html

Firmado electrónicamente por:
ALBERTO DANIEL NUNEZ AGURTO

- - - - - - - - - - - - - - - -

Núñez Agurto, Alberto Daniel

Director

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

Certificación

Certifico que el trabajo de integración curricular: "**Clasificación de ataques mediante técnicas de aprendizaje automático en Redes Definidas por Software**" fue realizado por el señor **Castillo Camacho, Miguel Angel**, el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Santo Domingo de los Tsáchilas, 24 de febrero de 2023

Núñez Agurto, Alberto Daniel

C.C.: 1716572548

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

RESPONSABILIDAD DE AUTORÍA

Yo, **Castillo Camacho, Miguel Angel** con cédula de ciudadanía n° 230020355 declaro que el contenido, ideas y criterios del trabajo de integración curricular: **Clasificación de ataques mediante técnicas de aprendizaje automático en Redes Definidas por Software**, es de mi autoría y responsabilidad, cumpliendocon los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidospor la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Santo Domingo de los Tsáchilas, 24 de febrero de 2023

Firma:

.....

Castillo Camacho, Miguel Angel

C.C.: 2300203557

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **Castillo Camacho, Miguel Angel** con cédula de ciudadanía n° 2300203557 autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Clasificación de ataques mediante técnicas de aprendizaje automático en Redes Definidas por Software**, en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Santo Domingo de los Tsáchilas, 24 de febrero de 2023

Firma:

…… ……

Castillo Camacho, Miguel Angel

C.C: 2300203557

Dedicatoria

Dedico este trabajo a mis padres porque desde el primer día de este camino universitario me apoyaron, corrigieron y me motivaron a continuar con mis estudios. A todos mis hermanos porque con cada uno de ellos crecí y junto a ellos aprendí muchas cosas.

Agradecimiento

Agradezco a mis padres por su apoyo incondicional durante el proceso de mi formación académica y que me han permitido cumplir con un objetivo más de vida. También a familiares que supieron apoyarme. Agradezco a la universidad y carrera por brindarme la oportunidad de aprender muchas cosas. Agradezco a docentes y amigos por su colaboración en determinados tiempos.

Índice

Índice de fíguras

Índice de tablas

Resumen

Las Redes Definidas por Software (SDN) representan un nuevo modelo de red que separa la funcionalidad de control de la gestión de datos, lo que puede mejorar significativamente la eficiencia y flexibilidad de esta última. Sin embargo, se enfrenta a importantes amenazas, que ponen en peligro la seguridad y disponibilidad de los datos y servicios. Este trabajo tiene como objetivo definir un modelo de clasificación de ataques mediante técnicas de aprendizaje automático, para mejorar la capacidad de defensa y aumentar la seguridad de la gestión de datos en SDN. Para esto, se realizó una revisión sistemática de la literatura (SLR) siguiendo la metodología de Bárbara Kitchenham, cuyos resultados muestran información de los principales conjuntos de datos y las técnicas de aprendizaje automático más utilizadas en SDN. Además, se aplicó las metodologías de investigación en ciencia del diseño (DSR), el descubrimiento de conocimiento en bases de dato (KDD) y el análisis exploratorio de datos (EDA) para desarrollar los modelos de aprendizaje automático. Se utilizó dos conjuntos de datos públicos con tráfico SDN y se entrenaron tres modelos de aprendizaje automático: árboles de decisión (DT), bosques aleatorios (RF) y máquinas de soporte vectorial (SVM), con la aplicación de diferentes grupos de características. Los resultados obtenidos en la fase de entrenamiento fueron 99,76%, 99,31% y 99,50% de precisión, para DT, RF y SVM respectivamente. Sin embargo, en la fase de implementación, con la captura de tráfico de datos de red en directo y la clasificación de ataques en la plataforma SDN emulada en Mininet, se obtuvieron los siguientes resultados: 43,28%, 24,14% y 23,97% de precisión, para SVM, DT y RF respectivamente. Por lo tanto, los resultados obtenidos en este trabajo son mejores que el estado del arte y muestra el despliegue de un modelo de aprendizaje automático en una SDN.

Palabras clave: Redes Definidas por Software, Ataques, Aprendizaje Automático.

Abstract

Software-Defined Networking (SDN) represents a new network model that separates control functionality from data management, which can significantly improve the efficiency and flexibility of the latter. However, it faces significant security threats that endanger the security and availability of data and services. This work aims to define a model for classifying attacks using machine learning techniques to improve defense capabilities and increase data management security in SDN. To achieve this, a systematic literature review (SLR) was conducted following the methodology of Barbara Kitchenham, whose results provide information on the main data sets and machine learning techniques most used in SDN. In addition, research methodologies in design science (DSR), knowledge discovery in databases (KDD), and exploratory data analysis (EDA) were applied to develop the machine learning models. Two public datasets with SDN traffic were used, and three machine learning models were trained: decision trees (DT), random forests (RF), and support vector machines (SVM), with the application of different feature sets. (SVM), with the application of different feature sets. The results obtained in the training phase were 99.76%, 99.31%, and 99.50% accuracy for DT, RF, and SVM, respectively. However, in the implementation phase, with the capture of live network data traffic and attack classification on the SDN platform emulated in Mininet, the following results were obtained: 43.28%, 24.14%, and 23.97% accuracy for SVM, DT, and RF, respectively. Therefore, the results obtained in this work are better than state-of-the-art and demonstrate the deployment of a machine learning model in an SDN.

Keywords: Software Defined-Networking, Attacks, Machine Learning.

Capítulo I

Antecedentes

Las Redes Definidas por Software (SDN) han alcanzado un alto nivel de popularidad debido a su flexibilidad y escalabilidad, permitiendo monitorear globalmente el tráfico de red y realizar configuraciones rápidas, a diferencia de las redes tradicionales que son rígidas y estáticas (Bannour et al., 2018). A pesar de su imponente funcionalidad, son atractivas para ser objeto de ataques como denegación de servicio (DoS), denegación de servicio distribuido (DDoS) y botnets o redes de bots. Estos ataques son los más comunes en las infraestructuras de red y cada vez tienen mayor volumen y sofisticación (Yungaicela-Naula et al., 2021).

Un informe trimestral de Kaspersky analiza que los ataques DDoS se incrementan masivamente en una escala superior al 100% en el tercer trimestre de 2022 (Kaspersky, 2022). Esto ha sido considerado como un dato crucial para este tipo de ataques. En las SDN, estos ataques tienen enfoques en tecnologías como Internet de las Cosas (IoT), redes de quinta generación (5G) y computación basada en la nube. Este tipo de ataques es conocido por su amplia capacidad de consumir recursos de la red y apagar los servicios que se ofrecen en ella.

Existen ataques dirigidos a las infraestructuras de redes de organizaciones, que son llevados a cabo por botnets. Según el informe trimestral de Kaspersky, también se ha detectado un aumento en la aparición de nuevas botnets con capacidades ampliadas en cuanto a ataques DDoS. Los ataques de botnet comienzan con la actividad de escaneo de la red, se filtran mediante tráfico similar al de un usuario y luego crean ataques en múltiples etapas, culminando en la denegación de servicio (Hussain et al., 2021). Las redes de bots son difíciles de identificar y medir. Además, cuantificar los miembros de una red botnet, los tipos o variantes de bots y su localización es una tarea muy compleja

debido a su comportamiento y capacidad de expansión y reproducción. En la actualidad, estos ataques se dirigen a las últimas tecnologías.

Los métodos de clasificación de tráfico han experimentado una evolución importante, comenzando con la categorización basada en el puerto de la aplicación (Archanaa et al., 2017). Sin embargo, debido a los cambios repentinos en el uso de puertos por parte de las aplicaciones, este método se ha vuelto inválido. Más tarde, surgió el método de inspección de paquetes profundos (DPI), que se presentaba como la opción más precisa. Aunque esta técnica daba resultados prometedores, con el tiempo se convirtió en un proceso laborioso y de bajo rendimiento desde un punto de vista computacional intensivo (Hubballi & Swarnkar, 2018).

De manera imprevista, el aprendizaje automático (Machine Learning, ML) ganó importancia en el campo de las redes. Actualmente, se ha convertido en un método de clasificación flexible. Una de sus principales aplicaciones es la detección y clasificación del tráfico de red, así como también en métodos de extracción de características de flujo y monitoreo (Kumar et al., 2022). Este modo de clasificación se está haciendo cada vez más popular debido a la amplia variedad de algoritmos disponibles para su implementación.

Problemática

En redes tradicionales, los ataques se llevan a cabo en al menos un enlace de red con el objetivo de desactivar los servicios de los servidores disponibles. En cambio, en una red SDN, para desestabilizar los servicios, los ataques deben tener lugar en el controlador. Un atacante, una vez que ha filtrado la red, puede intentar generar un ataque DDoS saturando la red con tráfico innecesario. Al generar paquetes con direcciones IP desconocidas, los nodos de la red no pueden reflejar esta información en sus tablas de flujos, lo que hace que estos paquetes sean reenviados al controlador, que debe responder a las grandes cantidades de flujo generadas por el tráfico normal y el malicioso. Como resultado, la red

se ralentiza o incluso el controlador se apaga. Si el ataque proviene de una botnet, este intenta mezclar malware entre el tráfico normal para ingresar en el entorno del controlador y sobrecargar la CPU o la memoria, causando un colapso de la infraestructura de red. Aunque las redes SDN sonconsideradas una solución prometedora debido a su configuración sencilla y eficiente, su seguridad aún no está suficientemente comprendida y es poco explorada (Tan et al., 2020).

SDN presenta amenazas más graves debido a las vulnerabilidades en el componente centralizado de control. Aunque los controladores pueden estar centralizados lógicamente y geográficamente distribuidos, siempre dependen de la infraestructura. Por lo tanto, es importante que el controlador detecte y identifique rápidamente los flujos maliciosos para descomponer o mitigar los ataques y garantizar un funcionamiento eficiente de la red (Rahouti et al., 2022).

Justificación

Actualmente, las SDN son una tecnología altamente demandada en centros de datos y redes de operadores debido a su capacidad para contribuir a la innovación en el desarrollo de redes futuras (Tan et al., 2020). Sin embargo, SDN también presenta desafíos de seguridad, ya que los atacantes están enfocados en el componente central de inteligencia de la red, el controlador. Este dispositivo controla la red en su totalidad, incluyendo la redirección del tráfico en la capa de infraestructura. Por lo tanto, es necesario implementar mecanismos de seguridad efectivos para proteger el controlador y la red en su conjunto.

Existen varios estudios sobre seguridad que han adoptado diversos métodos para detectar ataques de origen desconocido. El aprendizaje automático es un enfoque efectivo que cuenta con una amplia variedad de algoritmos y ha demostrado ser ampliamente utilizado en estas tareas con resultados notables. ML es apto para trabajar con el tráfico de datos de la red, lo que le permite observar, aprender y tomar decisiones. Por lo tanto, es

viable crear un clasificador de ataques basado en anomalías del flujo de tráfico, utilizando técnicas de ML para identificar ataques que pongan en riesgo el funcionamiento y la seguridad en SDN.

Objetivos

Objetivo General

Definir un modelo de clasificación de ataques mediante el uso de técnicas de aprendizaje automático en Redes Definidas por Software.

Objetivos Específicos

- Determinar los enfoques de clasificación de ataques y los trabajos relacionados en el ámbito de seguridad en SDN.
- Realizar la selección de algoritmos de aprendizaje automático y el conjunto de datos apropiado para entrenar el modelo de clasificación de ataques en SDN.
- Desarrollar la implementación del modelo de clasificación de ataques en un escenario emulado.
- Evaluar el modelo de clasificación de ataques.

Alcance

El aprendizaje automático es una herramienta poderosa en la detección de ataques, ya que puede analizar grandes cantidades de datos y aprender a identificar patrones y anormalidades en el tráfico de la red. El objetivo de este proyecto es diseñar un modelo de clasificación de ataques basado en flujos de tráfico, utilizando técnicas de aprendizaje automático, con el fin de mejorar la capacidad de detección y prevención de ataques en SDN. Por lo tanto, el proceso de construcción del modelo incluye la selección de un conjunto de datos apropiado y la evaluación de diferentes algoritmos de aprendizaje

automático para determinar el más adecuado para nuestros propósitos. Una vez que se haya construido el modelo, se evaluará su efectividad en la detección de ataques.

Es importante destacar que el éxito de este proyecto depende en gran medida de la precisión del modelo en la identificación de ataques. Por lo tanto, se espera que el modelo tenga una tasa de precisión de al menos el 95% durante su proceso de identificación de ataques. Este resultado sería una contribución significativa a la seguridad de las redes SDN y una herramienta valiosa para organizaciones que utilizan este tipo de tecnología.

Hipótesis

El uso de técnicas de aprendizaje automático permitirá mejorar la precisión en la identificación de ataques en SDN en comparación con los métodos actuales.

Capítulo II

Marco Conceptual

Redes Definidas por Software

Las SDN son una arquitectura emergente que comprende un conjunto de tecnologías computacionales y se presenta como una innovación en el desarrollo de la infraestructura de red, permitiendo aumentar su dinámica, flexibilidad y revolucionar el control y la gestión de la red (Rahouti et al., 2022). En SDN, se puede gestionar una arquitectura centralizada, lo que permite tener una visión general de toda la red. Además, mediante el uso de aplicaciones de software, es posible programar el comportamiento de la red, analizar flujos, enlaces, obtener estadísticas, detectar anomalías en la red, entre otras funcionalidades. Todo esto se logra gracias a la separación lógica entre el plano de control y el plano de datos. Estos dos planos intercambian información a través de un protocolo de comunicación abierto llamado OpenFlow (Kreutz et al., 2015).

La principal característica de SDN es la gestión de la inteligencia de la red en un único dispositivo llamado controlador, que contiene los elementos del plano de control. Esto permite que SDN aborde diferentes problemas y ofrezca soluciones variadas a los desafíos (Bera et al., 2017). De esta forma, los dispositivos de hardware son independientes de los protocolos o especificaciones de implementación para un funcionamiento correcto de la red.

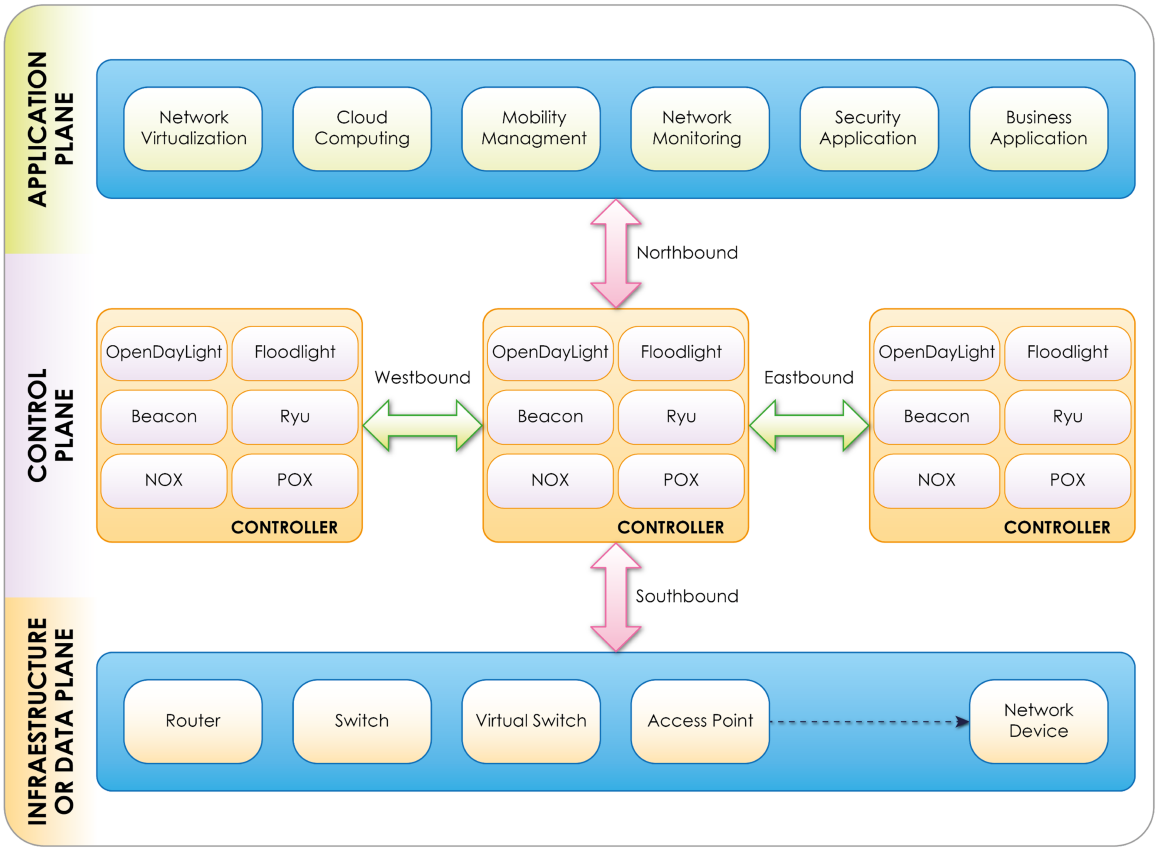
Arquitectura de SDN

La arquitectura funcional de SDN cuenta con tres capas: infraestructura, control y aplicación, como se muestra en la Figura 1. En la capa de infraestructura se encuentran todos los routers y switches que están conectados entre sí, siendo estos dispositivos los encargados de orquestar todo el tráfico generado en la red (Benzekki et al., 2017; Nunez-Agurto et al., 2022). En la capa de control se realizan las configuraciones de los

elementos de la capa inferior y se proporciona acceso a servicios disponibles en la red. En la capa de aplicación, se encuentran aplicaciones programables que definen el comportamiento y uso de la red. Su función es automatizar y satisfacer las necesidades de la organización en cuanto a la red. La comunicación entre la capa de aplicación y la capa de control se realiza a través de interfaces de programación de aplicaciones (API) (Scott-Hayward et al., 2016).

Figura 1

Arquitectura funcional de SDN.



Nota. Recuperado de Nunez-Agurto et al., 2022.

Controladores

El controlador es el encargado de centralizar toda la lógica relacionada con los procesos que se ejecutan en el plano de control y de gestionar la red y agregar funcionalidad (Sufiev et al., 2019). Los controladores son esenciales en el desarrollo de redes SDN, ya que proporcionan inteligencia para aprender de los hosts y los intercambios de información a través de las API. Por lo tanto, todo paquete que se desee intercambiar entre los hosts de la red debe ser procesado por el controlador y este envía la información al nodo más cercano del destino. Este nodo contendrá en su tabla de flujo la información sobre los paquetes y las direcciones origen y destino, lo que facilita y optimiza los procesos de enrutamiento en la red. La Tabla 1 muestra algunos controladores que pueden ser implementados en SDN

Tabla 1

Controladores SDN con soporte de Openflow (Ahmad & Mir, 2021).

Controlador	Lenguaje de programación	Plataforma
NOX	C++	Linux
POX	Python	Linux, MacOS y Windows
Floodlight	Java	Linux, MacOS y Windows
Opendaylight	Java	Linux, MacOS y Windows
Ryu	Python	Linux
Trema	Ruby y C	Linux
ONOS	Java	Linux, MacOS y Windows
Beacon	Java	Linux, MacOS y Windows

Seguridad de la información

La seguridad de la información se enfoca en proteger las fuentes de información, y se basa en garantizar la confidencialidad, la integridad y la disponibilidad. Estos tres aspectos

actúan como barreras para evitar el acceso no autorizado. Aunque existen diversas medidas tecnológicas para mantener la seguridad de la información, el ser humano es el factor clave en la aplicación de medidas preventivas y reactivas, ya que es conocido por ser el eslabón más débil en la cadena de seguridad (Evans et al., 2019). La confidencialidad se refiere a que la información no se revele a personas no autorizadas. La integridad se asegura de que la información permanezca sin alterarse desde su origen. La disponibilidad se refiere a que una entidad autorizada pueda acceder a la información en todo momento.

Ataques

Los ataques son amenazas a la seguridad que pueden tener lugar tanto en un entorno físico como en el ciberespacio. En la actualidad, los ataques en Internet a menudo se realizan de manera sutil utilizando tráfico similar al de los usuarios comunes (Rios et al., 2022). El objetivo de estos ataques es acceder de forma ilegal a un sistema de información o destruir un sistema informático o una infraestructura de red. Estos ataques surgen debido a la identificación de vulnerabilidades en los activos.

Tipos de ataques

Los ataques de denegación de servicio (DoS) tienen como objetivo acumular tráfico excesivo en la red para deshabilitar servicios. Este mismo tipo de ataque presenta una mejora conocida como DDoS donde varias fuentes distribuidas inundan de tráfico innecesario a distintos segmentos de la red. Actualmente, es común que estos ataques se repitan constantemente en los servidores web y su detección es compleja, debido a que en la red los patrones de tráfico son semejantes a la de los usuarios (Hong et al., 2018).

Otro tipo de actividad maliciosa y que es muy común en las redes, son los ataques de sondeo. Esta técnica tiene como objetivo descubrir equipos, escanear sus puertos y descubrir servicios que se ofrecen en la red. En base a ese orden, el resultado que se obtiene es un reconocimiento previo de la red, para encontrar posibles vulnerabilidades y

aprovecharlas posteriormente empleando otras técnicas de ataque (Gu et al., 2020).

Por último, los botnets son un conjunto de ordenadores sincronizados y comprometidos para ejecutar malware pernicioso (Koroniotis et al., 2019). Son utilizados para realizar actividades como el robo de información, toma de control de servicios en línea o la destrucción de redes informáticas. Las redes de bots están organizadas por varios bots (máquina infectada), un servidor de mando y control, y un botmaster (Vormayr et al., 2017). Este último, es el encargado de emitir las órdenes para llevar a cabo diversos ataques, así como los que se han mencionado anteriormente.

Inteligencia Artificial

La Inteligencia Artificial (IA) es un enfoque cuya intención es construir máquinas inteligentes que tengan la capacidad de imitar o incluso superar la inteligencia humana (Macas et al., 2022). La disciplina se está extendiendo cada vez más en diferentes campos. Cualquier software que replique las actividades humanas, ya sea para resolver problemas específicos o complejos, es considerado IA. De esta manera, una máquina puede aprender, razonar y predecir las entradas en un entorno establecido. El aprendizaje automático es una subárea de la IA que permite el desarrollo de tareas automatizadas a partir de datos.

Aprendizaje Automático

El aprendizaje automático es una técnica de Inteligencia Artificial que permite ejecutar un conjunto de instrucciones para absorber conocimiento a partir de los datos y, en base a ello, tomar decisiones (Macas et al., 2022). Los algoritmos de aprendizaje automático se clasifican en tres categorías: supervisado, no supervisado y de refuerzo.

El aprendizaje supervisado. Utiliza datos etiquetados y, en función del volumen de la entrada de nuevos datos o entidades, se asigna una etiqueta que es la responsable de dar un valor a la entrada. Algunos algoritmos de aprendizaje supervisado incluyen Regresión

Lineal (LR), Árboles de Decisión (DT), Random Forests (RF), Máquinas de Vectores de Soporte (SVM), k-Nearest Neighbors(K-NN) y Naive Bayes (NB).

El aprendizaje no supervisado. El aprendizaje no supervisado es el que trabaja en base a observaciones, es decir, con datos no etiquetados, tratando de descubrir patrones a partir del entorno que analiza. Existen algoritmos de clusterización y asociación. El primero se encarga de encontrar características similares según las entradas, como ejemplo se encuentra K-Means. El segundo descubre relaciones entre las características de cada muestra, como ejemplo está el algoritmo Apriori.

El aprendizaje por refuerzo. Es el que se basa en agentes, los mismos que extraen conocimiento en base al entorno que exploran. El conocimiento adquirido lo obtienen mediante reiteradas acciones que a su vez son recompensadas o sancionadas, dependiendo de los aciertos o errores que cometan durante su aprendizaje. Este aprendizaje es útil para la toma de decisiones. Aquí, el agente aprende de su experiencia y es capaz de deducir de acuerdo a la situación en la que se encuentra.

Métodos de clasificación de tráfico

El criterio de clasificación de tráfico de red consiste en categorizar y regular los flujos para equilibrar los recursos y gestionar eficazmente la red (Archanaa et al., 2017). Actualmente, los métodos de clasificación son: basados en puertos, inspección profunda de paquetes y técnicas de aprendizaje automático. El primer método consiste básicamente en analizar los paquetes activos de la red y clasificarlos según el puerto de las aplicaciones. Sin embargo, existen aplicaciones que utilizan puertos alternativos o protocolos no comunes, lo que dificulta la identificación del tráfico. El segundo método de clasificación consiste en verificar la carga útil de los paquetes, es decir, su contenido, para identificar la aplicación. Es considerado una técnica efectiva pero costosa en términos operacionales (Hubballi & Swarnkar, 2018). Además, tiene como desventaja la falta de

detección en tráfico cifrado. Finalmente, la clasificación de tráfico basada en aprendizaje automático usa datos estadísticos y se ha convertido en uno de los métodos preferidos debido a su rapidez de detección. Las técnicas de aprendizaje máquina pueden utilizarse para detectar tráfico normal o maligno, e incluso se pueden ampliar sus funciones para mitigar ataques o seleccionar características basadas en el número de paquetes, bytes o duración del flujo, entre otros (Pérez-Díaz et al., 2020).

Estado del arte

En este apartado se presentan los prototipos desarrollados para la detección y clasificación de ataques en redes SDN. Se llevó a cabo una revisión sistemática de la literatura (SLR) siguiendo las pautas de Barbara Kitchenham (Kitchenham & Brereton, 2013), mediante la planificación, realización y documentación de la revisión para elaborar el estado del arte. Los prototipos deben estar enfocados en problemas relacionados con ataques que afecten el funcionamiento y la infraestructura de la red, y las soluciones deben utilizar técnicas de aprendizaje automático para clasificar los ataques en SDN.

Planificación de la revisión

Identificación de la necesidad de una revisión

Para comenzar la revisión literaria, se identifica primero el problema central tratado en la tesis en cuestión. Después, se investigan los estudios relevantes al tema con el fin de adquirir conocimiento y fortalecer la idea que se desea desarrollar. Además, esto permite verificar la existencia de estudios previos relacionados con el tema y abordar cualquier aspecto no resuelto.

Especificación de las preguntas de investigación

Se han formulado tres preguntas de investigación que serán respondidas a medida que se avance en la revisión de la literatura.:

- RQ1. ¿Cuáles son los modelos de Machine Learning utilizados en la clasificación

de ataques en SDN?

- RQ2. ¿Cuáles son las características utilizadas por las soluciones actuales en la clasificación de ataques en SDN?
- RQ3. ¿Cuáles son los conjuntos de datos disponibles para entrenar modelos de clasificación de ataques en SDN?

Criterios de inclusión y exclusión

La finalidad de aplicar criterios de inclusión y exclusión es identificar estudios relacionados con cuestiones metodológicas de clasificación de ataques, usando técnicas de aprendizaje automático en SDN. Por lo tanto, los criterios de exclusión son:

- Artículos que no enmarquen SDN.
- Trabajos sobre modelos de clasificación que no apliquen aprendizaje automático.
- Artículos cuyo contenido no se encuentre en inglés.
- Artículos publicados entre los años 2016 y 2022.

En cambio, los criterios de inclusión aplicados son:

- Artículos cuya propuesta metodológica se enfoque en la detección y/o clasificación de ataques en SDN, utilizando únicamente técnicas de aprendizaje automático.
- Artículos que presenten en su contenido una justificación, discusión o evaluación de las técnicas de aprendizaje automático utilizadas, herramientas y resultados encontrados.
- Artículos que se encuentren publicados en revistas cuyo cuartil sea Q4 en adelante.

Construcción de la cadena de busqueda

Proceso de busqueda

Para identificar los artículos relacionados con el tema de investigación, se utilizaron las bases de datos científicas IEEE Explore, Web of Science y Scopus, que son especializadas en el área de ciencias de la computación. Se realizaron previamente investigaciones para establecer los términos clave, y se utilizaron las siguientes palabras clave para la búsqueda: Software Defined Network, classification, detection, attack, anomaly, machine learning.

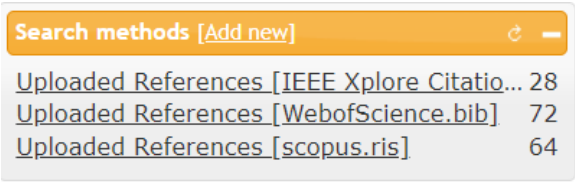
Por lo tanto, la cadena de búsqueda queda de la siguiente manera:

- ((SDN OR “Software Defined Network” OR “Software-Defined Networking”) AND (Classification* OR Detection) AND (Attack OR Anomaly) AND (“Machine Learning”)).

Esta cadena búsqueda es ingresada en las tres bases de datos científicas, aplicando el periodo de tiempo establecido anteriormente. Luego, se extraen estos artículos y se cargan en Rayyan, el cual es una herramienta adecuada para proceder con la revisión sistemática de la literatura. Se obtienen un total de 164 artículos. En la Figura 2 se muestra el número de trabajos obtenidos por cada base de datos científica.

Figura 2

Número de artículos subidos en Rayyan.



Realizar la revisión

En esta sección, se identifican los estudios con mayor relevancia. Con el uso de la herramienta Rayyan, se empieza por eliminar los artículos duplicados. Luego de una revisión de literatura, los artículos se seleccionan de acuerdo a los criterios de inclusión y exclusión. En este caso, 20 artículos han sido incluídos. Finalmente, con la información analizada se procede a responder las preguntas de investigación planteadas anteriormente:

En respuesta a la RQ1, los algoritmos de aprendizaje automático se presentan como soluciones prometedoras para la identificación y clasificación de tráfico en redes definidas por software. Por un lado, SVM es un algoritmo configurable que se extiende para identificar varias clases de tráfico y puede ser utilizado para discernir flujos que presenten un comportamiento atípico, ya que contiene un núcleo que realiza operaciones no lineales en la división de datos (Shahzadi et al., 2021). Asimismo, indican que el algoritmo KNN, a pesar de contar con un enfoque no paramétrico, es útil para asociar diferentes tipos de tráfico debido a que sus decisiones se basan en los vecinos más cercanos según las características que presenta cada entrada."

DT es un algoritmo que abarca dos criterios (Gini y Entropy) hábiles para tomar decisiones en cuanto a las características de referencia para la toma de decisiones, donde cada criterio brinda cambios significativos (Tonkal et al., 2021). Asimismo, XGBoost emplea una búsqueda en profundidad para aprender características complejas y sus decisiones se basan de manera similar al árbol de decisión. Solo interviene un marco de refuerzo de gradiente para optimizar el rendimiento de los modelos en la clasificación de tráfico (Alzahrani & Alenazi, 2021).

Por otro lado, NB es un algoritmo cuyas decisiones se basan en valores probabilísticos para la clasificación de tráfico, de acuerdo a la cantidad de datos empleados para el entrenamiento. Este clasificador se destaca porque no necesariamente requiere de un

conjunto de datos significativo y, además, es escalable en cuanto al porcentaje para establecer un límite de decisión (Sangodoyin et al., 2021). Finalmente, RF es un algoritmo que produce varios árboles de decisión para obtener resultados finales de la clasificación. Este modelo tiende en algunos casos a obtener resultados más precisos debido a la aleatoriedad de selección de características y permite obtener mayor certeza en cuanto a la identificación de una clase de tráfico (Ahuja et al., 2021). Esto se debe a que cada árbol que se genera arroja un resultado. Al final, en base al grupo de resultados, se toma una decisión sobre la entrada procesada.

Respondiendo a RQ2., en cuanto a características para ataques de DDoS dentro del trabajo de Alzahrani y Alenazi (2021), considera cinco características: duration (tiempo de duración de la comunicación), protocol-type, src-bytes (bytes de datos enviados de origen a destino), srv-count (número de conexiones al mismo servicio) y dst-host-same-src-port-rate (porcentaje de conexiones desde los servicios de puerto al host de destino). En cambio, Alamri y Thayananthan (2020) selecciona 24 características y las que más destacan son: flag, dst-bytes (bytes de datos de destino a origen), serror-rate (tasa de conexiones erróneas), logged-in (estado de la conexión). Sudar y Deepalakshmi (2021) optan por 11 características, entre las cuales están: same-srv-rate (promedio de conexiones al mismo servicio), diff-srv-rate (promedio de conexiones a diferentes servicios). En el primer estudio la selección la realiza el autor de forma manual. Mientras que, en el segundo, el autor aplica el método de la ganancia de información (IG). Para el tercer autor, utiliza varios de métodos de selección para escoger las mejores características. Además del método mencionado previamente, aplica el método de selección secuencial, de correlación, Chi-cuadrado e información mutua.

Tariq y Baig (2017) presenta un sistema de detección para ataques de botnets, donde aplica un módulo de extracción de características en base a una revisión previa del comportamiento que genera este tipo de ataque. Toma en cuenta la duración del flujo, total

de bytes recibidos, total de paquetes recibidos, bytes por segundo, paquetes por segundo, byte por paquete y tiempo de llegada del flujo. Los valores de estas características los considera tanto en su valor mínimo, promedio y máximo.

En cuanto a la RQ3, se muestran en la Tabla 2 los conjuntos de datos identificados en las propuestas de solución a problemas de ataques en SDN.

Tabla 2

Conjuntos de datos identificados para propuestas de solución de ataques en SDN.

Dataset	Descripción
NSL-KDD	Parte de KDD-Cup. Contiene 41 características entre tráfico normal y de ataque. Los ataques son: DoS, Probe, R2L y U2R.
CTU-13	Comprende sólo tráfico de botnet con siete tipos de bots.
ISOT	Abarca subconjuntos de datos con diferentes escenarios de tráfico normal y ataque (DoS, Botnet, Ransomware).
DDOS attack SDN Dataset	Contiene 23 características de tráfico DDoS en protocolos TCP, UDP e ICMP.
UNB-ISCX	Incluye tráfico normal, DoS, DDoS, fuerza bruta.
CSE-CIC-IDS2018	Contiene tráfico normal y siete escenarios de ataque: DoS, DDoS, Botnet, Heartbleed, Fuerza bruta y ataques web.

Elaboración del estado del arte

EP1. An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks (Sahoo et al., 2020). En el presente estudio se diseñó un framework para la detección de ataques DDoS, utilizando SVM como modelo principal. El módulo desarrollado recibe las estadísticas de flujo de los switches OpenFlow, luego se extraen las características, las cuales entran en el modelo de ML. El conjunto de datos utilizado para

entrenar el modelo fue creado por el autor (Sahoo et al., 2020), el cual contiene 27 características y 224709 muestras de cuatro tipos de ataque. Como técnica de selección de características, se aplicó el Análisis de Componentes Principales del Núcleo (K-PCA). Para optimizar el modelo, se usó el Algoritmo Genético (AG) y durante el entrenamiento se utilizó la validación cruzada K-Fold con 5 iteraciones. Las herramientas que se utilizaron fueron: switches OVS, controlador POX y Mininet. Como resultado, el modelo SVM+K-PCA+AG clasifica entre tráfico normal y tráfico malicioso, obteniendo una precisión (accuracy) de 98.90%, con la partición del dataset 90:10 para entrenamiento y prueba. Con esta misma división de datos, empleando los modelos SVM, KNN y RF, se alcanzaron precisiones (accuracies) de 95.54%, 92.02% y 94.33%, respectivamente.

EP2. Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-Defined Networks Against DDoS Attacks (Alamri & Thayananthan, 2020). En la presente investigación desarrollan un mecanismo de control de ancho de banda y un clasificador de flujos en la red, utilizando XGBoost. El objetivo es detectar ataques DDoS. Los autores emplean un algoritmo de control de ancho de banda para obtener un valor umbral adaptativo (basada en tiempo y tasa de bytes) y penalizar los flujos que superen ese límite. De tal modo, que desarrollan módulos para fijar valores umbrales, supervisar y controlar el ancho de banda, detectar flujos anormales y obtener estadísticas en tiempo real. En el caso de detectar tráfico anormal, se establece un valor que, si es superado por el contador de infracciones de umbral máximo, este entra a la clasificación de flujos para determinar si es normal o ataque DDoS. Los datos son coleccionados por los autores usando Hping3 y iPerf3, usando Mininet, Ryu, 3 OVS y 9 hosts. El modelo XGBoost se entrena con 20 características. Utilizan la técnica SMOTE para equilibrar las clases minoritarias del dataset. Para evaluar el rendimiento del modelo, utilizan los conjuntos de datos CICDDoS2019, NSL-KDD y CAIDA DDoS, con el cual obtuvieron 99.70%, 99.90% y 99.92% de accuracy respectivamente. En cambio, en el

entorno configurado anteriormente obtienen 99.90% de accuracy y recall, 99.98% de precisión y F1-score. Para determinar el tipo de ataque de DDoS, utilizan CICDDoS2019 y NSL-KDD, consiguiendo un accuracy de 91.26% y 99.91% respectivamente.

EP3. An intelligent flow-based and signature-based IDS for SDNs using ensemble feature selection and a multi-layer machine learning-based classifier (Sudar & Deepalakshmi, 2021). En la investigación realizada, se desarrolla un clasificador multicapa para clasificar el tráfico en un Sistema de Detección de Intrusos (IDS). El objetivo es aplicar mecanismos de seguridad en el plano de control y plano de datos. Para el plano de control, se plantea un módulo de IDS basado en flujos, utilizando técnicas de ML para la selección de características y clasificación de entradas. El modelo se entrena con 11 características. Se considera el conjunto de datos NSL-KDD, que se combina con el dataset de los autores, generado con la herramienta Scapy. Por otro lado, se emplea la selección de características basada en conjunto (EFS), combinando con la Ganancia de Información (IG), selección de características secuenciales y correlacionales, la prueba de Chi-cuadrado y la información mutua (IM). En cambio, en el plano de datos, se crea un IDS basado en firmas, utilizando Snort para inspeccionar distintas instancias de tráfico en SDN. El modelo de clasificación multicapa sitúa a SVM en la primera capa, mientras que en la segunda capa se encuentran NB y C4.5. De acuerdo a los resultados de la capa 1, NB inspecciona el tráfico normal, de lo contrario, C4.5 analiza el tráfico de ataque en sus cuatro tipos. De forma individual, durante el entrenamiento y la prueba, la precisión (accuracy) de SVM es de 92.12% y 93.50%, NB obtiene 93.12% y 94.41%, mientras que C4.5 alcanza 93.31% y 95.16% en el orden dado. En cambio, de manera conjunta, el modelo multicapa consigue un 97.70% de precisión (accuracy).

EP4. Automated DDOS attack detection in software defined networking (Ahuja et al., 2021). En este estudio, se implementa un modelo de clasificación de tráfico benigno y malicioso con técnicas de ML. Se desarrolla un modelo híbrido basado en el

clasificador de vectores de soporte (SVC) y RF. SVC actúa como el primer clasificador y RF procesa los resultados del primero. El modelo se entrena con datos generados por los autores, los cuales contienen 23 características. Para la consecución del dataset, se utilizan Mininet, Ryu, un OVS, dos hosts y las herramientas mgen y hping3. Para la división de datos, se usa una proporción de 80:20 para entrenamiento y prueba. Se emplea PCA para reducir a 20 dimensiones el dataset y la incrustación estocástica de vecinos distribuida (t-SNE) para analizar la distribución del tráfico obtenido. Como resultado final, el modelo híbrido SVC+RF obtiene un accuracy del 98.80%.

EP5. Comprehensive DDoS Attack Classification Using Machine Learning Algorithms (Ussatova et al., 2022). En esta investigación se aplican algoritmos de ML supervisados para clasificar ataques DDoS en un entorno SDN. Se hace uso de un conjunto de datos publicado por los autores (Ahuja et al., 2020) el cual contiene 23 características con dos clases: tráfico benigno y malicioso. Se utiliza SMOTE para el equilibrio de clases y el escalador Min-Max como método de normalización de datos. Las técnicas aplicadas para la selección de características para reducir la dimensionalidad del dataset son IG, Chi-Square y F-test. Se utiliza la proporción 70:30 para entrenamiento y prueba, implementando los modelos NB, LR, SVM, K-NN, DT, RF, XGBoost y CatBoost. En cuanto al rendimiento de los modelos, DT, RF, K-NN, XGBoost y CatBoost alcanzan 99.00% de precisión. En cambio, NB logra 64.00%, mientras que SVM y LR obtienen 74.00%.

EP6. Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks (Alzahrani & Alenazi, 2021). Los autores diseñaron un sistema de detección de intrusos basado en la red (NIDS) con el objetivo de detectar comportamientos maliciosos en SDN. Para ello, crearon un programa basado en algoritmos de ML y lo desplegaron en la capa de aplicación de SDN. Los modelos utilizados fueron XGBoost, Random Forest (RF) y Decision Tree (DT). Utilizaron el

conjunto de datos NSL-KDD, que contiene 41 características, pero solo consideraron cinco. Emplearon la técnica de normalización basada en el escalador Min-Max. El NIDS se compone de tres elementos configurables: un gestor de reglas para la red, un detector de ataques y el clasificador del tipo de ataque. Este a su vez se integra al controlador y compara activamente el tráfico. Finalmente, obtuvieron un accuracy de 95.55%, 94.60% y 94.50% respectivamente.

EP7. DDoS Detection in SDN using Machine Learning Techniques (Nadeem et al., 2022). En este estudio, utilizan diferentes técnicas de selección de características para encontrar las más óptimas y desarrollar modelos de ML. Como objetivo se plantean detectar ataques DDoS. Para ello, a partir del conjunto de datos NSL-KDD, crean una capa de selección de características, con tres métodos: filtrado, envoltura e integrado. El primer método utiliza IG, coficiente de correlación y Chi-Square. En el segundo método utiliza la selección de características hacia adelante (FFS), eliminación de características hacia atrás (BFE) y recursiva (RFE). Para el último método, aplican el operador de selección y contracción mínima absoluta (Lasso). Las características que se seleccionan para los modelos SVM, KNN, NB, RF y DT, varían en cuanto a cada método de selección de características. Con 28 características y aplicando la técnica de eliminación recursiva, los modelos SVM, KNN, NB, RF y DT alcanzan un accuracy de 89.18%, 98.57%, 92.12%, 99.97% y 99.80%, respectivamente.

EP8. Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning (Sangodoyin et al., 2021). Se presenta un sistema de detección y clasificación de ataques DDoS de bajo coste, basado en algoritmos de ML. Los autores realizan un análisis comparativo para escoger los mejores modelos de aprendizaje automático en cuanto a rendimiento de predicción y clasificación. Posteriormente, emplearon 3600 datos entre ataques de inundación HTTP, UDP, TCP y tráfico normal, donde cada una de estas clases son

clasificadas. Las muestras de ataque fueron generadas con la herramienta Cañón de Iones de Órbita Baja (LOIC) sobre una red SDN emulada en Mininet. Los datos son normalizados con el método Z-Score. Posteriormente, cada modelo predice en cuanto a las métricas de fluctuación, tiempo de respuesta y rendimiento. Los modelos implementados son: Árbol de Clasificación y Regresión (CART), Gaussian Naive Bayes (GNB), K-NN y Análisis Discriminante Cuadrático (QDA). En ese mismo orden, el accuracy obtenido es de 98%, 96.1% 95.6% y 95.9%. Los resultados que se muestran se obtienen a partir del conjunto de datos establecido para la preparación de los modelos.

EP9. Machine learning approach equipped with neighbourhood component analysis for DDoS attack detection in software-defined networking (Tonkal et al., 2021). El presente documento, plantea un sistema de detección de ataques de DDoS en SDN, aplicando algoritmos de ML equipados con Análisis de Componentes de Vencindad (NCA). En la entrada de datos de cada modelo, presenta un módulo de selección de características. Este módulo es implementado con el algoritmo NCA, cuyo objetivo es extraer las características con mayor relevancia y reducir el coste computacional durante el entrenamiento de cada modelo de ML. Los modelos implementados son KNN, DT, ANN y SVM, mismos que son entrenados con 14 características del conjunto de datos "DDOS attack SDN Dataset". Como resultado, el sistema detecta entre tráfico normal y de ataque. El modelo DT con criterio Gini, alcanza un accuracy de 99.17% durante su entrenamiento, superando al resto de algoritmos implementados como SVM y KNN alcanzando 81.48% y 97.75%. Con estos resultados se determina si el tráfico es normal o de ataque.

EP10. Machine Learning Based Botnet Detection in Software Defined Networks (Tariq & Baig, 2017). En este trabajo se propone un método de detección de botnets basado en flujos mediante técnicas de ML en SDN. El enfoque principal es implementar un algoritmo de ML para comparar activamente los flujos de red actuales con los datos utilizados para entrenar dicho modelo. Por lo tanto, se desarrollan cuatro

módulos: recolección de trazas, señalización de traza, extracción de características y detección de trazas. Específicamente, se aplica el algoritmo C4.5. Los conjuntos de datos tratados son CTU-13 e ISOT. Para el primer módulo, se utiliza el repositorio de datos de series temporales (TSDR) del controlador OpendayLight, lo que permite recoger las trazas actuales y formar pequeños lotes de flujos. En el segundo módulo, se agrupan los flujos con la misma clave en un lote. Para el tercer módulo, se aplica la extracción de características de los flujos conglomerados anteriormente entre 10 a 60 flujos. Por último, en el modelo se procesan los datos estadísticos extraídos y se determina si es tráfico normal o botnet. A pesar de que sólo utilizan un modelo, obtienen un accuracy de 94.80%. Sin embargo, no se describen las técnicas de preprocesamiento de datos de los flujos recolectados en tiempo real, ya que los datos estadísticos obtenidos de cada característica pueden ser muy diferentes. Aun así, la propuesta final utiliza 8 características.

EP11. Overhead Reduction Technique for Software-Defined Network Based Intrusion Detection Systems (Janabi et al., 2022). Aquí se expone un IDS basado en ML para identificar flujos normales o de ataques en un entorno SDN y que, a su vez, permita reducir la carga computacional en el controlador. El IDS desarrollado contiene 3 módulos: extracción de características, analizador y decisión. El sistema extrae los datos de la tabla de flujo del switch y, a su vez, son enviados por un canal IDS para ser analizados. Posteriormente, en el analizador se agrupa cada una de las muestras para formalizar en fila el flujo completo. Finalmente, estos datos son enviados al módulo de decisión que implementa el algoritmo NB, el cual determina si pertenece a un flujo normal o de ataque. El modelo se encuentra entrenado con 14 características con las muestras del conjunto de datos CSE-CIC-IDS2018. Como resultado, obtienen un 98.46% de precisión. La reducción de la carga computacional del controlador se centra en la agregación de un canal de comunicación independiente de OpenFlow entre el plano de datos y control. Se destaca el uso del método de T-Test para reducir el número de características innecesarias

y optimizar el tiempo de detección del modelo.

EP12. The DDoS attacks detection through machine learning and statistical methods in SDN (Dehkordi et al., 2021). Los autores proponen un sistema de detección de ataques DDoS aplicando métodos estadísticos y de ML en SDN. Los módulos que presentan son: colección de datos, selección de datos basada en entropía y clasificación. Los datos son recogidos de las tablas de flujo y enviados al controlador. En el controlador se ejecuta el selector de flujos, mediante el método de entropía que se encarga de calcular, mediante valores umbrales, un promedio de estimación de que determinado flujo se considere como ataque. El resultado del módulo anterior es la entrada del clasificador, donde se determina el tipo de flujo, es decir, normal o de ataque. Previamente, el modelo es entrenado con 14 características utilizando los conjuntos de datos UNB-ISCX, CTU-13 e ISOT. El modelo de clasificación se determina entre los algoritmos JR8, BayesNet, Regresión Logística, NB, RandomTree y REPTree. En ese mismo orden, el accuracy alcanzado por cada modelo es de 99.93%, 99.71%, 99.79%, 97.75%, 99.95% y 99.96%. A pesar de que la mayoría de los modelos implementados no son tradicionales, los resultados obtenidos únicamente con los datos de entrenamiento son aceptables.

EP13. Distributed denial of service attacks detection for software defined networks based on evolutionary decision tree model (Kamel & Abdullah, 2022). Los investigadores emplean el algoritmo DT como modelo principal para detectar ataques de DDoS en SDN. Partiendo del conjunto de datos “DDOS-attack SDN dataset” y seleccionando 18 características se entrena el modelo DT mejorado. Este modelo se refuerza a partir del algoritmo genético (AG), puesto que permite encontrar los valores de los hiperparámetros más adecuados para su aprendizaje. Al final este árbol se construye con 49 niveles, es decir, la profundidad máxima que alcanza entre los nodos de decisión. El modelo logra un accuracy de 99.46%, clasificando tráfico normal y ataque.

EP14. SD-Honeypot Integration for Mitigating DDoS Attack Using Machine Learning Approaches (Sumadi et al., 2022). Esta investigación utiliza algoritmos de ML para detectar ataques de DDoS e integra un sensor honeypot en el plano de control para aplicar reglas de mitigación. El trabajo presenta dos módulos: de detección y mitigación. Para detectar los ataques se utiliza un algoritmo de ML. En cambio, para mitigarlos se emplean reglas de restricción que son emitidas desde un servidor honeypot hacia el controlador. El conjunto de datos es generado bajo un escenario real con las herramientas TCPReplay y Scapy. Los modelos implementados son SVM, GNB, K-NN, RF y Árbol de Clasificación y Regresión (CART), los cuales clasifican entre tráfico normal y ataque. El plano de control emplea el sensor Suricata, que permite identificar anomalías que fluyen en la red, y luego las dirige al modelo para determinar un promedio de estimación sobre una amenaza. Por último, se aplican las reglas de mitigación bajo el honeypot integrado. En cuanto al accuracy obtenido, SVM alcanza el 93%, GNB el 70%, mientras que K-NN, CART y RF logran el 69%. En cuanto al servidor, su función es almacenar los ataques en una base de datos, fijando la cabecera de la información del paquete extraído. En base a la carga computacional generada con esos datos, se aplican reglas de prevención, como por ejemplo la suspensión del enlace

A continuación, la Tabla 3 muestra el análisis de los trabajos seleccionados y sobre todo se examina los resultados conseguidos por los autores:

Tabla 3

Soluciones de ataques basadas en ML.

Autor	Técnica ML	Conjunto de datos	Resultado de modelo
(Sahoo et al., 2020)	SVM	(Alkasassbeh et al., 2016)	Detectar ataques DDoS, clases: Normal y ataque.

Continúa en la siguiente página...

Tabla 3 – continuación de la página anterior

Autor	Técnica ML	Conjunto de datos	Resultado de modelo
(Alamri & Thayananthan, 2020)	XGBoost	Generado por los autores	Detectar ataques DDoS, clases: Normal y ataque.
(Sudar & Deepalakshmi, 2021)	SVM, NB y C4.5	NSL-KDD	Detectar y clasificar ataques DDoS, clases: Normal, Probe, DoS, R2L y U2R.
(Ahuja et al., 2021)	SVC+RF	Generado por los autores	Detectar ataques DDoS, clases: Normal y ataque.
(Ussatova et al., 2022)	NB, LR, SVM, KNN, DT, RF, XGBoost y CatBoost	(Ahuja et al., 2020)	Detectar ataques DDoS, clases: Normal y ataque.
(Alzahrani & Alenazi, 2021)	XGBoost, RF, SVM y DT	NSL-KDD	Detectar y clasificar ataques DDoS, clases: Normal, DoS, Probe, R2L y U2R.
(Nadeem et al., 2022)	SVM, KNN, NB, RF y DT	NSL-KDD	Detectar ataques DDoS, clases: Normal y ataque.
(Sangodoyin et al., 2021)	CART, GNB, KNN y QDA	Generado por los autores	Detectar y clasificar ataques DDoS por clases: Normal, HTTP, TCP y UDP.
(Tonkal et al., 2021)	KNN, DT y SVM	DDOS attack SDN Dataset	Detectar ataques DDoS, clases: Normal y ataque.

Continúa en la siguiente página...

Tabla 3 – continuación de la página anterior

Autor	Técnica ML	Conjunto de datos	Resultado de modelo
(Tariq & Baig, 2017)	C4.5	CTU-13, ISOT	Detectar ataques de botnet, clases: Normal y botnet.
(Janabi et al., 2022)	NB	CSE-CIC-IDS2018	Detectar ataques DDoS, clases: Normal y ataque.
(Dehkordi et al., 2021)	JR8, BayesNet, Regresión Logística, NB, RandomTree y REPTree	UNB-ISCX, CTU-13, ISOT	Detectar ataques DDoS, clases: Normal y ataque.
(Kamel & Abdullah, 2022)	DT	DDOS-attack SDN dataset	Detectar ataques DDoS, clases: Normal y ataque.
(Sumadi et al., 2022)	SVM, GNB, KNN, RF y CART	Generado por los autores	Detectar ataques DDoS, clases: Normal y ataque.

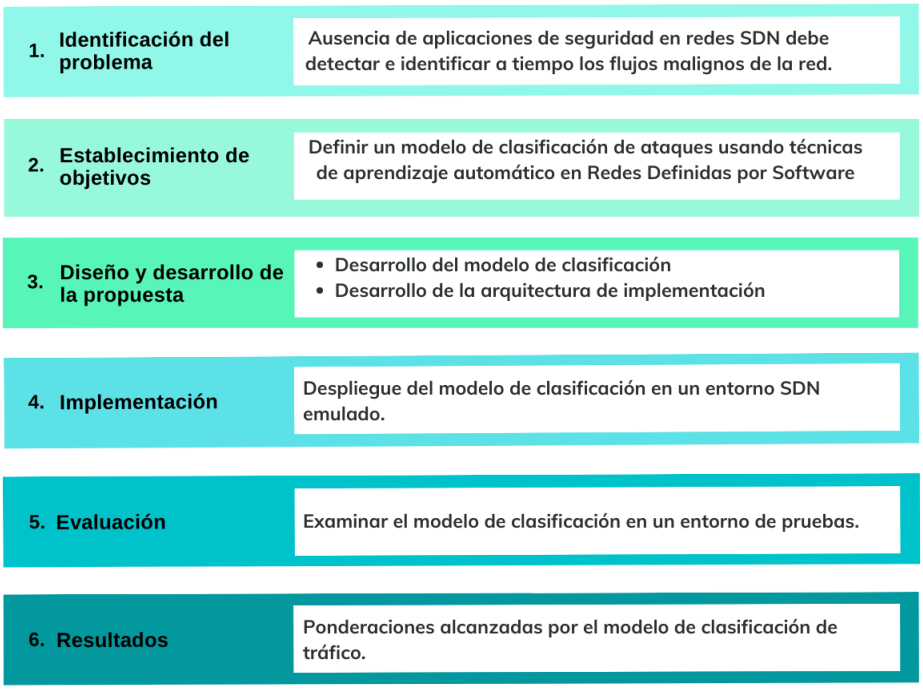
Capítulo III

Metodología de trabajo para la propuesta de solución

La metodología adoptada en el presente trabajo es la Investigación Científica del Diseño (Design Science Research, DSR). Este enfoque se centra en desarrollar diversos mecanismos o instrumentos tecnológicos a partir de un diseño y posteriormente evaluarlos para su aplicación final. Se caracteriza por soportar un conjunto indefinido de métodos, aplicaciones, herramientas, entre otros, que contribuyen a la investigación y desarrollo de soluciones en problemas identificados, incrementando el valor de la literatura y el conocimiento en la elaboración de diseños (An & Kim, 2018). La Figura 3 presenta etapas para llevar a cabo las actividades que conducen al cumplimiento de la propuesta.

Figura 3

Fases de la metodología DSR adaptadas.



Nota. Recuperado de Montenegro et al., 2016.

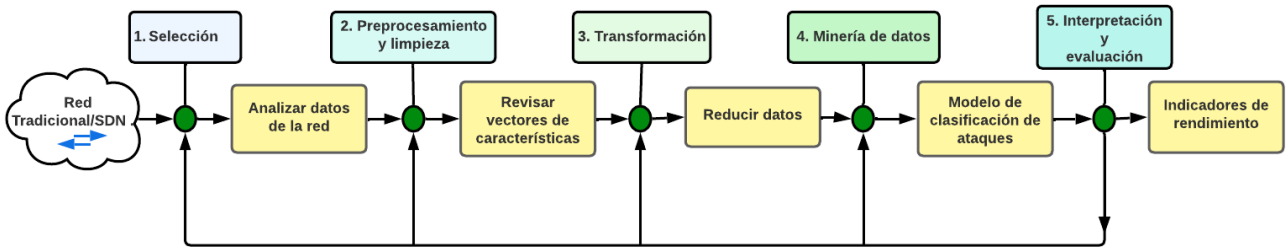
Metodología para el desarrollo del modelo de clasificación

Descubrimiento de conocimiento en bases de datos

El descubrimiento de conocimiento en bases de datos (Knowledge Discovery in Databases, KDD) se divide en fases iterativas e interactivas para desarrollar un flujo de trabajo basado en ML. KDD comprende un conjunto de procedimientos que se agrupan en cinco fases: selección, preprocesamiento, transformación, minería de datos e interpretación y evaluación. Su principal objetivo es extraer conocimiento de los datos que previamente se desconocían (Bosc et al., 2017). La metodología avanza de manera cronológica e iterativa para encontrar patrones en los datos que permitan resolver el problema planteado. La Figura 4 muestra las fases que comprende KDD.

Figura 4

Flujo de trabajo en base a la metodología KDD.



A continuación, se describe cada una de las fases establecidas:

- **Selección de datos**: consiste en seleccionar un conjunto de datos de origen primario, es decir, que no tiene ningún proceso de tratamiento de datos. A partir de ellos se analizan las estadísticas que aportan los datos y se determina si son o no viables para resolver el problema planteado.

- **Preprocesamiento y limpieza de datos**: esta fase involucra reconocer las variables

independientes y dependientes, esto significa que se debe establecer la matriz de características y la etiqueta a predecir. Al mismo tiempo se debe analizar datos faltantes, valores atípicos, tratándolos con métodos estadísticos.

- **Transformación de datos**: define un rango de valores de los datos para evitar que ciertas características predominen sobre las demás. También, incluye procesos de transformación de datos en valores numéricos. Por último, se aplica métodos para la reducción de la matriz de características con el fin de optimizar el rendimiento del modelo en cuanto al aprendizaje e identificación.

- **Minería de datos**: comprende desarrollar y ejecutar algoritmos de aprendizaje automático previamente analizados. En este caso, los algoritmos se seleccionan en base a los trabajos anteriormente investigados. Así mismo, incluye la implementación de técnicas para idealizar un modelo de aprendizaje, de regularización, sobreajuste, entre otros. Los resultados obtenidos se analizan en la siguiente fase y depende de las técnicas implementadas en las fases anteriores para tratar los datos.

- **Interpretación y evaluación**: se analizan los resultados y se sustentan en base a las métricas para evaluar el rendimiento del modelo de aprendizaje automático. En base a la eficacia que disponga un algoritmo, se procede a su implementación donde se somete a pruebas en un entorno realista.

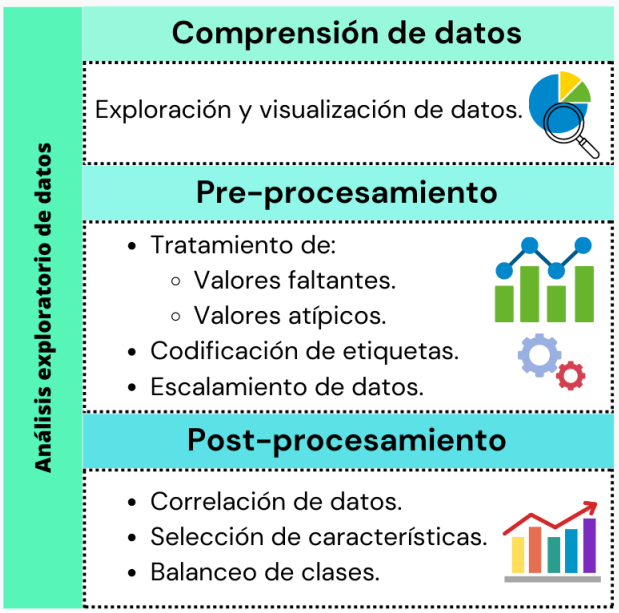
Análisis exploratorio de datos

El análisis exploratorio de datos (Exploratory Data Analysis, EDA) consiste en comprender los datos desde su origen hasta prepararlos para el descubrimiento de patrones, mediante la implementación de modelos de aprendizaje automático. EDA abarca tres etapas: comprensión de los datos, pre-procesamiento y

post-procesamiento (da Silva Junior et al., 2022). La Figura 5 muestra cada una de las etapas y las estrategias que plantea para su consideración.

Figura 5

Etapas de EDA.



- **Comprensión de los datos**: conlleva realizar un descubrimiento de los datos, mediante el análisis y visualización de datos. Es opcional realizar una visualización de los datos en procesos iniciales del análisis exploratorio del conjunto de datos, puesto que esto se puede llevar en procesos posteriores. Sin embargo, esto deja un vacío en la visualización y sobre todo en la interpretación de los datos, debido a que se encuentra firmemente enlazada en la utilización de métodos visuales (Batch & Elmqvist, 2018).

- **Pre-procesamiento**: se aplican diversas técnicas para operar con datos faltantes, valores atípicos, codificación de etiquetas y escalamiento de datos. En base a los datos que se presenten y el contexto, se debe planificar procesos para emprender

cada una de las técnicas para el tratamiento de datos. Como resultado, se obtiene un conjunto de datos completamente limpio. A pesar de ello, es necesario dominar el contexto de los datos antes de tomar decisiones que impliquen el tratamiento de los datos.

- **Post-procesamiento**: esta etapa se centra en analizar la correlación y balanceo de datos. Aquí se aplican diferentes estrategias para seleccionar características con el fin de reducir la matriz o dimensión del conjunto de datos. Uno de los objetivos de utilizar métodos de selección de características, es facilitar a la discriminación de los datos.

Capítulo IV

Origen de los datos

Los datos provienen del dataset público de InSDN, el cual se desarrolla sobre una red SDN específica para generar tráfico normal y de ataques, lo que resulta importante para evaluar el rendimiento o desarrollar nuevos sistemas de detección de intrusos. El tráfico de ataque cubre diferentes elementos que forman parte de la red SDN, y sus tipos son: DoS (52,741), DDoS (48,413), Probe (36,372), ataque de fuerza bruta (1,110), ataques web (192) y Botnet (164). Por otro lado, el tráfico normal (68,424) se genera con protocolos como HTTPS, HTTP, SSH, correo, DNS, FTP, y se ejecutan diversas aplicaciones populares como Facebook y YouTube. Los registros de cada uno de estos tipos de tráfico se encuentran en archivos CSV, cuyo tamaño es de aproximadamente 1,21 GB y 3,58 GB, respectivamente. Los autores utilizaron Wireshark para capturar el tráfico y CICFlowMeter para extraer los flujos. Además, en este conjunto de datos se pueden encontrar archivos PCAP (Elsayed et al., 2020)."

Sin embargo, al contar con flujos ya extraídos, se determinó por utilizar Argus para aumentar la credibilidad de los datos. Esta herramienta permite extraer flujos unidireccionales y bidireccionales. Además, Argus se ejecuta sobre entornos Linux y ofrece mayor flexibilidad en las opciones de configuración para la extracción de características y la transformación a diversos archivos para el almacenamiento de los datos (Argus, 2022). Otra herramienta opcional es CICFlowMeter que funciona para la extracción de flujos bidireccionales, permitiendo obtener 84 características y se presenta como una librería de Python, aunque también presenta una versión en Java (Habibi Lashkari, 2018). Empleando Argus se configuró para extraer 120 características. Por lo tanto, todos los archivos PCAP que ofrece el dataset InSDN fueron procesados para obtener sus flujos.

Adicionalmente, se agregó tráfico de botnet obtenido del conjunto de datos público de CTU-13 el cual contiene 13 escenarios de tráfico botnet. No obstante, se consideran los archivos PCAP del primer escenario correspondiente al tipo de bot Neris (García et al., 2014). La Tabla 4 muestra una comparación del número de flujos extraídos en ambas herramientas.

Tabla 4

Comparación de flujos extraídos en CICFlowMeter y Argus.

Tráfico	Cantidad	CICFlowMeter	Argus
	InSDN		
Normal	68424	53784	111719
DoS	52471	33842	35030
DDoS	48413	398643	3352394
Probe	36372	41980	49452
Fuerza bruta	1110	582	886
Ataque web	192	120	564
Botnet	164	92	193
	CTU13		
Botnet	2753290	132168	192168

Selección de los datos

En base a los flujos extraídos anteriormente, se descartan las clases que contienen muestras reducidas, es decir, se consideran los tipos de tráfico Normal, DDoS, DoS y Probe. Se optó por estas clases para que exista un mayor equilibrio en los datos, aunque para el tráfico normal y DDoS el número de flujos extraídos supera los 100000 registros. Por otra parte, es necesario comprender las 120 características que han sido extraídas,

donde se puede distinguir que están basadas en identificadores de red, paquetes, bytes, flags, tiempo y flujos. En la Tabla 5 se presentan las descripciones de las características.

Tabla 5

Lista de características.

Característica	Descripción
SrcId	Identificador de argus.
Rank	Número de secuencia del registro de flujo.
StartTime (Src, Dst)	Tiempo de inicio del flujo.
LastTime (Src, Dst)	Tiempo en que finaliza el flujo.
Trans	Contador de registros de flujos agregados.
Seq	Número de secuencia argus.
Flgs	Indicadores de estado de flujo.
RunTime	Tiempo total de ejecución de flujo activo.
Dur (Src, Dst)	Duración del flujo.
Dur Mean(StdDev, Sum, Min, Max)	Estadísticas de duración del flujo.
Dir	Dirección de la transacción o comunicación.
Addr (Src, Dst)	Dirección IP del host origen y destino.
Proto	protocolo de transacción.
Sport	Puerto de origen.
Dport	Puerto de destino.
Tos (Src, Dst)	Valor de byte para indicar el tipo de servicio requerido por el host origen y destino.
DSb (Src, Dst)	Valor de byte de servicio diferenciado en host origen y destino.
Ttl (Src, Dst)	Tiempo de vida del paquete de origen a destino y viceversa.

Continúa en la siguiente página...

Tabla 5 – continuación de la página anterior

Característica	Descripción
IpId (Src, Dst)	Identificador de IP de origen y destino.
Cause	Estado del flujo.
TotPkts	Total de paquetes en la transacción.
Pkts (Src, Dst)	Paquetes emitidos por host origen y destino.
TotBytes	Total de bytes en la transacción.
Bytes (Src, Dst)	Bytes emitidos por host origen y destino.
TotAppByte	Total de bytes de aplicación.
AppBytes (Src, Dst)	Bytes de aplicación de origen a destino y viceversa.
PCRatio	Grado de relación o interacción entre productor-consumidor.
Load	Número de bits por segundo en el flujo.
Load (Src, Dst)	Número de bits por segundo emitidos por el host origen y destino.
Loss	Total de paquetes perdidos en el flujo.
Loss (Src, Dst)	Total de paquetes perdidos en el flujo por host origen y destino.
pLoss	Porcentaje de paquetes perdidos.
pLoss (Src, Dst)	Porcentaje de paquetes perdidos por host origen y destino.
Retrans	Total de paquetes retransmitidos.
Retran (Src, Dst)	Total de paquetes retransmitidos por host y destino.
SrcGap	Bytes faltantes en el flujo de datos por host origen.
DstGap	Bytes faltantes en el flujo de datos por host destino.
Rate	Paquetes por segundo.
Rate (Src, Dst)	Paquetes por segundo emitidos por host origen y destino.

Continúa en la siguiente página...

Tabla 5 – continuación de la página anterior

Característica	Descripción
IntPkt (Src, Dst)	Tiempo de llegada entre paquetes enviados desde el host y destino.
IntPktAct (Src, Dst)	Tiempo de llegada que permanece activo los intervalos de paquetes.
IntPktIdl (Src, Dst)	Tiempo que permanece inactivo los intervalos de paquetes.
IntPktMax (Src, Dst)	Tempo de llegada máximo entre paquetes.
IntPktMin (Src, Dst)	Tempo de llegada mínimo entre paquetes.
Jitter (Src, Dst)	Fluctuación de origen y destino.
JitAct (Src, Dst)	Tiempo que permanece activo la fluctuación de paquetes en origen y destino.
JitIdl (Src, Dst)	Tiempo sin fluctuación de paquetes.
State	Estado de la transacción o comunicación.
Win (Src, Dst)	Tamaño ventana TCP de host origen y destino.
TCPBase (Src, Dst)	Número de secuencia base de TCP de host origen y destino.
TcpRtt	Métrica de tiempo en ida y vuelta de la configuración en la conexión TCP.
SynAck	Tiempo de sincronización TCP.
AckDat	Tiempo de confirmación TCP.
TcpOpt	Indicadores de estado de las operaciones que se ejecutan en TCP.
Inode	Nodo intermedio entre origen y destino.
Offset	Bytes de desplazamiento sobre el flujo de datos.
sPktSz (Mean, Max, Min)	Tamaño de paquete, promedio, máximo y mínimo en host origen.

Continúa en la siguiente página...

Tabla 5 – continuación de la página anterior

Característica	Descripción
dPktSz (Mean, Max, Min)	Tamaño de paquete, promedio, máximo y mínimo en host destino.

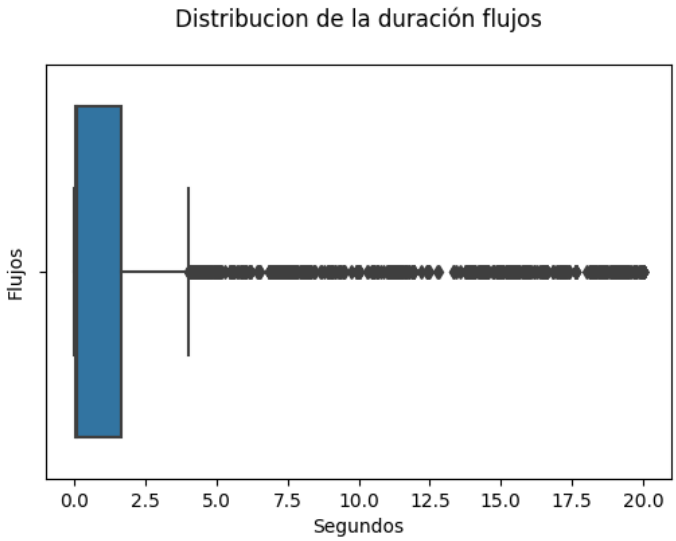
Análisis de los datos

En este apartado, se ha llevado a cabo un conocimiento preliminar de los tipos de datos y valores que almacena cada característica. Este análisis es fundamental para tener una mayor comprensión de los datos, y en base a ello, se deduce aquellas características que son útiles para las fases posteriores. Es decir, se plantea una posible matriz de características con su variable objetivo.

De acuerdo al seguimiento del análisis de los datos, en la Figura 6 se observa cómo los flujos se encuentran en diferentes escalas entre 0 y 20 segundos. La mayoría de las muestras se encuentran en el segundo cuartil, es decir, con un valor superior a 0.0 segundos y menor a 2.5 segundos. Sin embargo, existe un alto número de flujos que se encuentran en el último cuartil, superando los 5 segundos y alcanzando el límite de los 20 segundos.

Figura 6

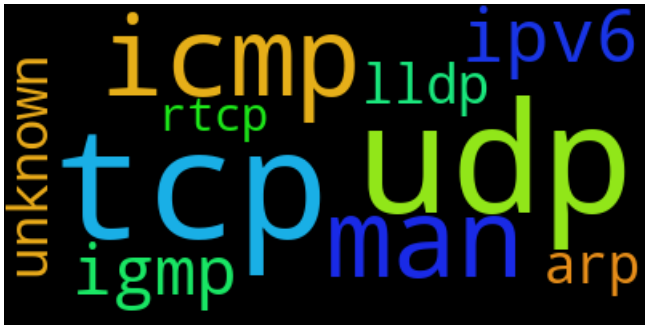
Duración de flujos extraídos.



Existen protocolos en las diferentes capas del modelo OSI, entre los cuales destacan los protocolos de la capa de transporte como TCP y UDP. Asimismo, existen protocolos de la capa de red como ICMP, IGMP, IPv6, ARP y MAN. Por lo tanto, los protocolos que más se utilizan son UDP y TCP, sobre los cuales se trabajará. En la Figura 7 se muestran los protocolos generadores de tráfico.

Figura 7

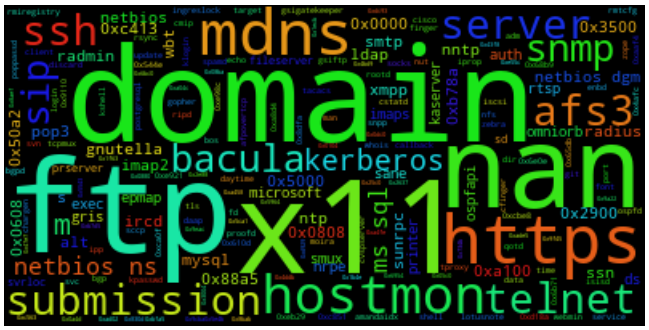
Protocolos de Internet soportados



Para finalizar, en la Figura 8 se muestran los puertos que han sido más buscados por los hosts origen, es decir, los puertos destino. Se deduce que DNS, FTP, HTTPS y el servicio X11 son los más concurridos. También hay tráfico de telnet, ssh, kerberos, entre otros, que son variados. Aun así, existen valores faltantes (NaN) que se tratan en la siguiente fase.

Figura 8

Puertos destino con mayor concurrencia.



Limpieza y preprocesamiento de datos

Valores faltantes

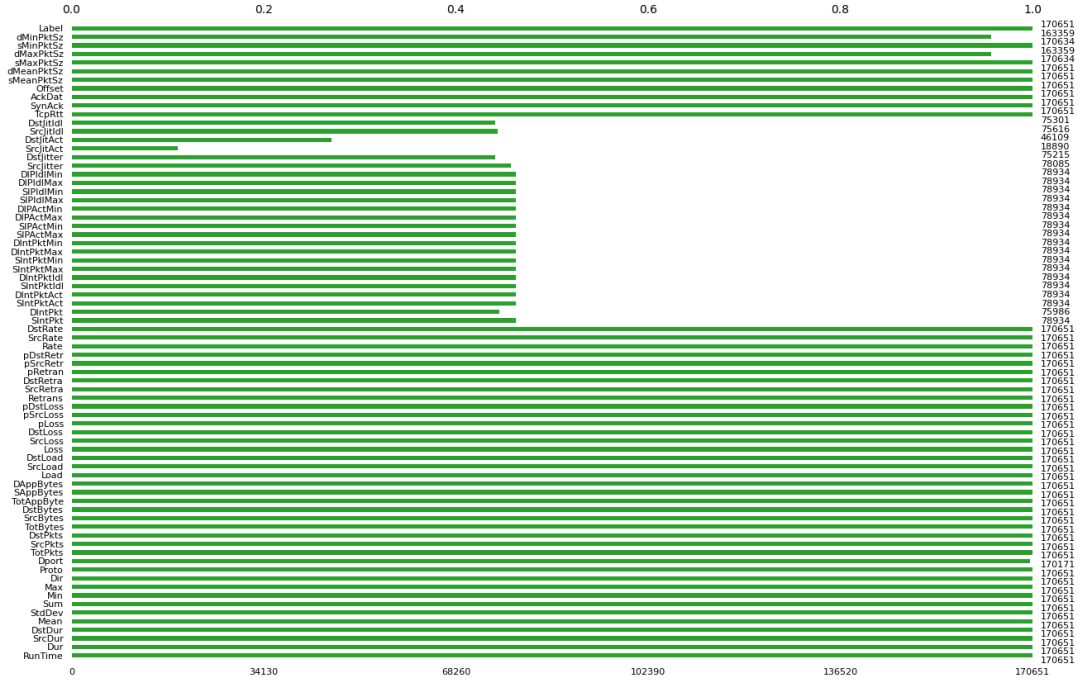
La existencia de valores faltantes es una de las grandes anomalías que pueden existir en los conjuntos de datos, de los cuales se desconoce el motivo de su ausencia, y es necesario comprender a qué tipo de valores faltantes pertenece. Existen tres tipos de estos valores: faltantes completamente al azar (MCAR), faltantes al azar (MAR) y no perdidos al azar (NMAR) (Garcia et al., 2020). Para el primer tipo existe una alta probabilidad de que en el conjunto de datos falte cualquier tipo de valor en diferentes instancias y atributos. Para el segundo, los valores faltantes se registran en cualquiera de los atributos. Mientras que, para el último tipo los valores se ausentan porque depende de otro valor el cual también es ausente.

Durante la inspección y visualización de los datos se detectaron valores faltantes del tipo NMAR. La Figura 9 muestra los valores ausentes en las características de tiempo de

llegada en los intervalos de paquetes, tiempos que permanecen activos los paquetes, fluctuación y tamaño del paquete entre el host origen y destino. Esto afecta los valores estadísticos de promedio, mínimo y máximo. Ante esta situación, se verificó el número de atributos que son dependientes de las características principales. Así mismo, se identificaron los atributos de la misma clase, pero diferente operación. Es decir, si una instancia no cuenta con el tamaño máximo o mínimo de los paquetes enviados por el host origen o destino, siendo cero el tamaño promedio de paquetes, entonces estas características también deben poseer ese mismo valor. Por lo tanto, se aplicaron estrategias de imputación por media y valores constantes mediante la clase SimpleImputer(), mientras que para las características que tienen un número elevado de datos faltantes, se procedió a eliminarlas.

Figura 9

Análisis de características con valores faltantes.



Valores atípicos

Los valores atípicos, conocidos como outliers, son aquellos que difieren de otros, ya sea por el punto de ubicación del valor de un dato, comportamientos distintos o valores no comunes bajo la misma característica. Por otro lado, todos los datos que se encuentran en un mismo nivel o rango de valores visualmente conocidos se denominan inliers. Para la detección de estos valores, existen diferentes métodos como los basados en técnicas estadísticas, la distancia entre observaciones, la densidad, el agrupamiento y el aprendizaje (Wang et al., 2019).

Durante la inspección de los datos, se identificó que algunos registros tenían nomenclaturas de puertos diferentes que no pertenecían al rango de puertos conocidos, registrados y dinámicos. Por lo tanto, se procedió a analizar el protocolo que usa ese flujo y, en base a ello, se eliminaron esos registros. De esta manera, se manejan flujos que corresponden a un grupo de puertos anteriormente mencionados. Por otro lado, se analizaron características como el total de paquetes, bytes y número de paquetes emitidos por segundo. En algunas instancias de estas características contienen valores que podrían tratarse como atípicos. Sin embargo, estos valores reflejan el tráfico real de cada una de las clases procesadas.

Codificación de etiquetas

En este apartado se realiza el proceso de codificación de las características que poseen datos no numéricos. Específicamente, tanto los protocolos como los puertos, son analizados en base a la numeración que maneja la Autoridad de Números Asignados de Internet (IANA). Por lo tanto, se maneja una codificación de datos discretos. En cambio, para la variable objetivo se manejan por categorías.

Transformación de datos

Escalamiento de datos

Desde esta etapa, se cuentan con 48 características restantes que se han mantenido durante el proceso de limpieza y procesamiento de datos. Considerando este número de características reducidas, se analizan minuciosamente cuáles de ellas pueden formar parte de la matriz final de las variables independientes. De modo que, todas estas características se someten al escalamiento de datos mediante la estandarización. Este enfoque se aplica porque no es sensible a grandes valores y es ideal para mantener la distribución de los valores atípicos del tráfico de red en una escala diferente. Anteriormente se analizaron los valores atípicos y se indicó el motivo de su conservación en algunas características. La fórmula de la estandarización es la siguiente:

$$Z = \frac{(xi - mean(x))}{stdev(x)} \tag{1}$$

Estandarizar datos implica tener el promedio o media en cero y la desviación estándar en uno. Como resultado se obtiene cuánto varía cada observación con respecto a la media de ese conjunto de características. De esta forma, se puede comprobar la igualdad en la distribución de los datos originales con los que se encuentran estandarizados. La Figura 10 y Figura 11 muestra los datos sin estandarizar y el resultado de la estandarización aplicada al conjunto de datos con las características restantes.

Figura 10

Datos no estandarizados.

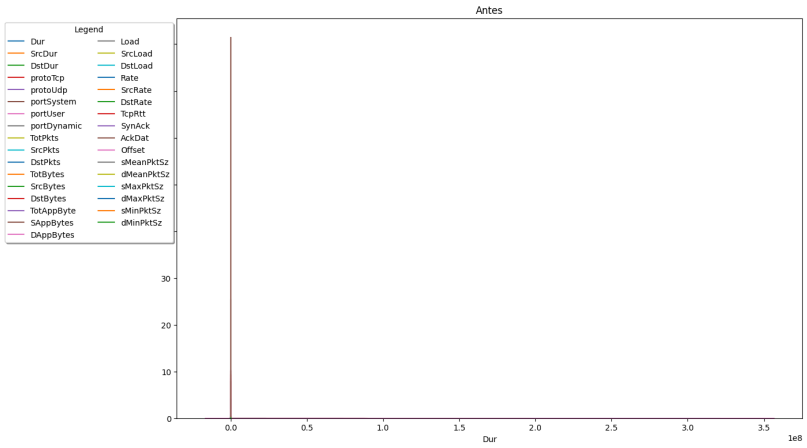
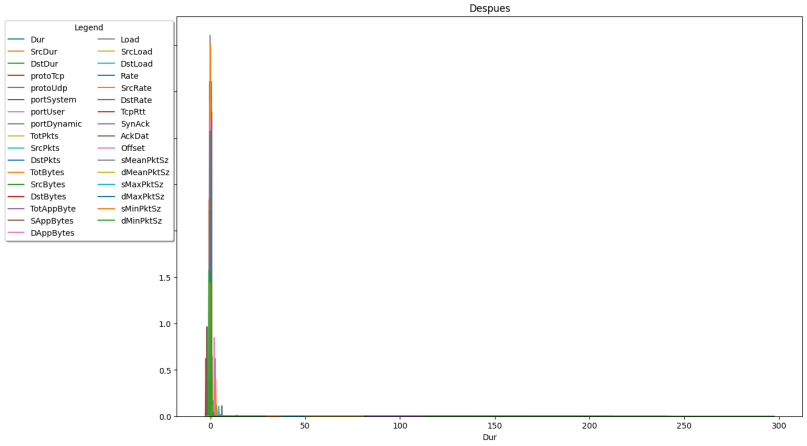


Figura 11

Estandarización de datos.



Selección de características

El objetivo de este apartado es encontrar un subconjunto predefinidos de características que optimice la puntuación de los modelos de clasificación. Inicialmente, es necesario comprender los datos y en base a ello, inferir el tipo de información que se puede obtener con esos datos. Es así, que para conocer la importancia de las características, en un

principio se manejó una escala personal entre alta, media y baja. Sin embargo, también se utilizó SelectKBest, el cual es proporcionado por Scikit-learn y es un método que permite seleccionar las mejores características del conjunto de datos. Este método arroja dos resultados; primero, los atributos que se consideren los más relevantes y luego, los menos importantes. A pesar de ello, es necesario realizar un análisis cuidadoso de las características antes aplicar otras técnicas para procesos posteriores.

La Tabla 6 muestra las características consideradas como esenciales para el método SelectKBest. En la segunda columna se muestra la ponderación personal asignada a cada una de las características, lo cual requiere un amplio conocimiento para tomar una decisión. En la última columna, se establecen las características que deben ser reemplazadas por aquellas que contienen valores repetitivos entre los atributos de algunas instancias o que tienen una valoración baja. No hay un orden específico para estas características.

Tabla 6

Selección de características.

SelectKBest	**Ponderación**	**Intercambio**
RunTime	Alta	Proto
Dur	Alta	
SrcDur	Media	
DstDur	Media	
Mean	Alta	TcpRtt
Sum	Media	SynAck
Min	Alta	AckData
Max	Alta	SrcPkts

Continúa en la siguiente página...

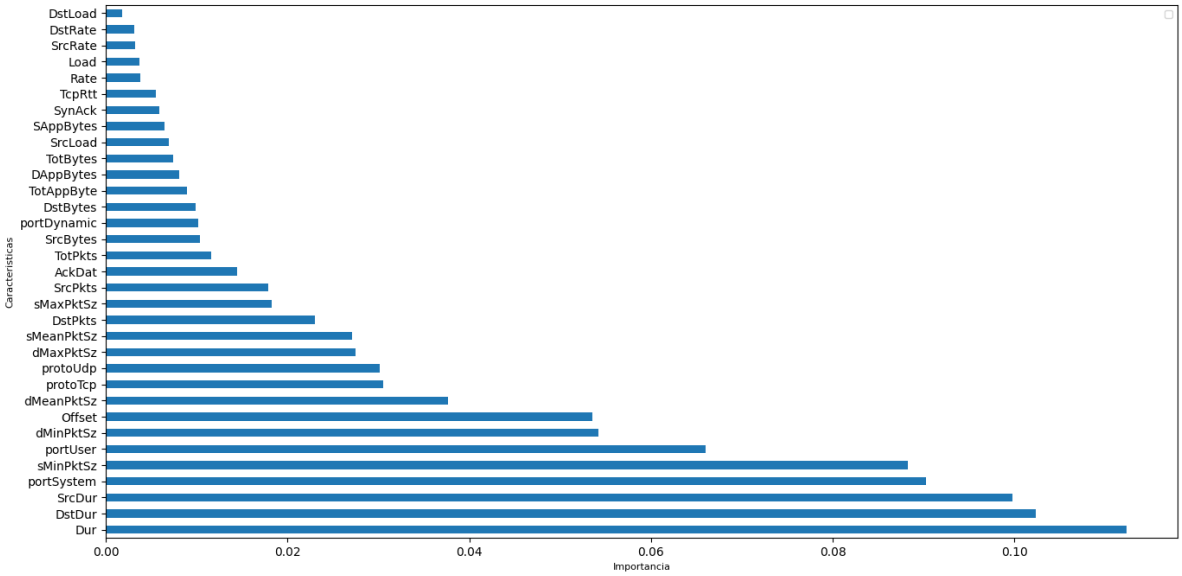
Tabla 6 – continuación de la página anterior

SelectKBest	Ponderación	Intercambio
Dport	Alta	
TotPkts	Alta	
TotBytes	Alta	
SrcBytes	Alta	
DstBytes	Alta	
TotAppByte	Media	
SAppBytes	Media	
DAppBytes	Media	
Load	Media	
SrcLoad	Media	
DstLoad	Media	
pLoss	Baja	DstPkts
pSrcLoss	Baja	sMinPktSz
Rate	Alta	
SrcRate	Alta	
DstRate	Alta	
Offset	Media	
sMeanPktSz	Alta	
dMeanPktSz	Alta	
sMaxPktSz	Alta	
dMaxPktSz	Alta	
dMinPktSz	Alta	

Como resultado, se seleccionaron 30 características. En consecuencia, la Figura 12 muestra el resultado de aplicar el clasificador de Árboles Extremadamente Aleatorios (ExtraTrees) para comprobar el nivel de importancia considerado en cada una de los atributos. Esta es otra técnica de selección de características que se ha utilizado en este caso, para reafirmar que las variables escogidas sean correctas. A partir de estos resultados, se aplican dos técnicas para extraer características, las cuales son: matriz de correlación y PCA.

Figura 12

Distribución de puntuación para la selección de características.



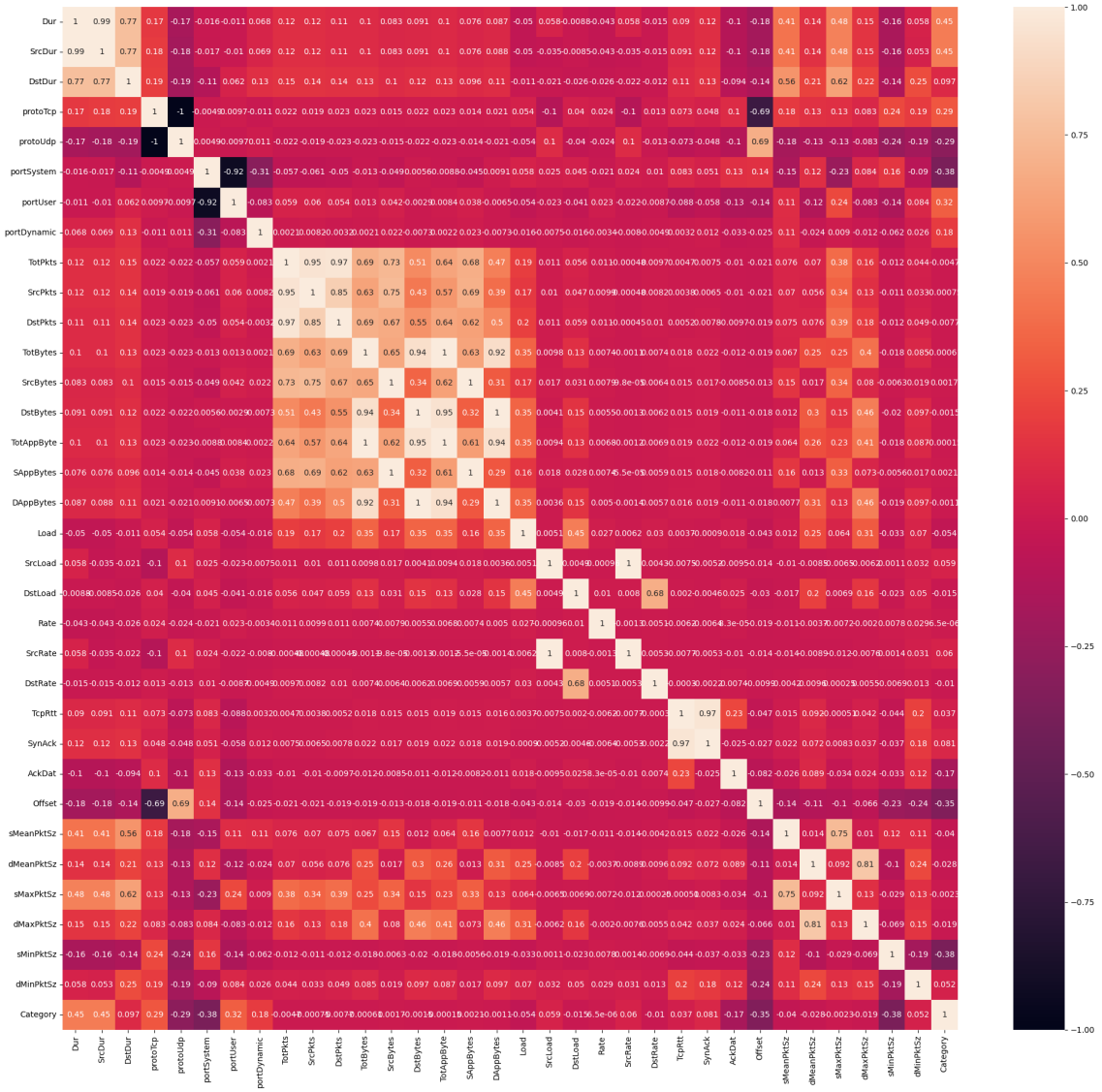
Matriz de correlación

La Figura 13 muestra la matriz de correlación, que permite analizar la relación lineal entre cada una de las variables. Todas las entradas diagonales son iguales a uno, mientras que las entradas lineales se obtienen a partir del coeficiente de correlación de Pearson, por lo que sus valores varían. El valor más cercano a uno (1) determina la magnitud de coherencia entre las variables, mientras que si el valor se acerca a menos uno (-1),

entonces esa característica no tiene una relevancia para las demás características (Mestre & Vallet, 2017).

Figura 13

Matriz de correlación.



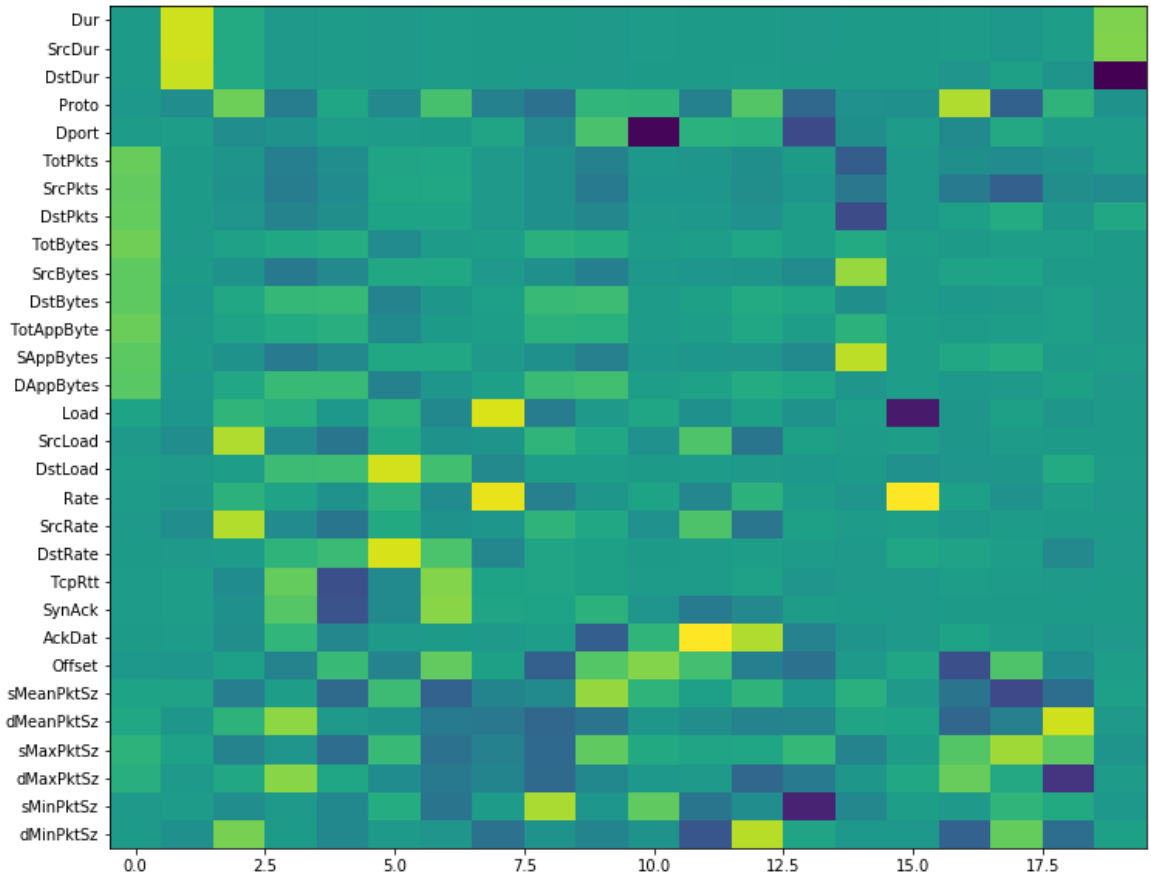
Análisis de componentes principales

PCA proporciona un modo exploratorio de características, el cual se aplica a un grupo de datos para descomponer linealmente las variables y maximar la varianza (Camacho

et al., 2019). Si la varianza no se optimiza, mayor será la dispersión de los datos. Por lo tanto, para aplicar este método, es necesario aproximar la media a cero y la desviación estándar a uno. La Figura 14 muestra la matriz de covarianza que se compone de la dimensión (número de características) y la proyección de componentes. El número de componentes a analizar depende de la cantidad de características y de las que se desee generar como entrada al modelo de aprendizaje.

Figura 14

Matriz de covarianza

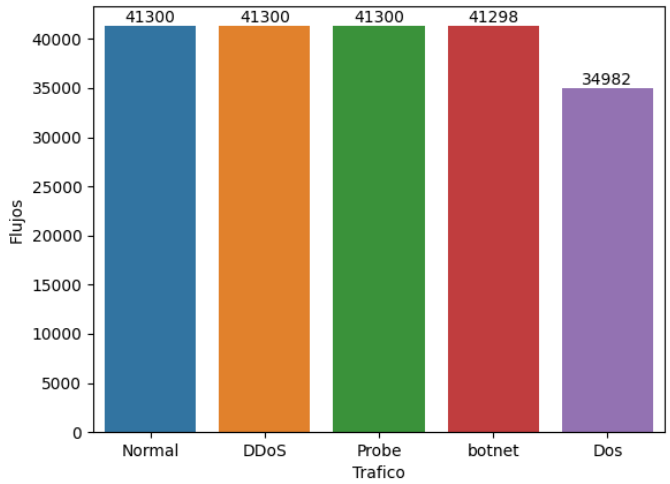


Balanceo de clases

En función de la cantidad de muestras por cada tipo de tráfico que se indicó anteriormente (ver Tabla 4) y de los resultados obtenidos en el proceso de limpieza y preprocesamiento de datos, se debe seleccionar una cantidad adecuada de datos para el entrenamiento de los modelos. La Figura 15 refleja la cantidad de tráfico que se logró limpiar, donde las clases Botnet y DoS no cuentan con la misma cantidad de tráfico que las clases Normal, DDoS y Probe. Si bien se podrían igualar todas las clases a la misma cantidad de la clase DoS, se decidió trabajar con 34200 muestras para cada una. De esta forma, se evita que el modelo aprenda de forma sesgada para otras clases.

Figura 15

Selección de cantidad de tráfico por clase



Conjunto de características

En base a los apartados anteriores de la transformación de datos, uno de los objetivos es evaluar el impacto que tiene cada una de las variables en los resultados de clasificación entre cada uno de los tipos de tráfico. Por lo tanto, se crean cuatro conjuntos de características, siendo este paso sumamente importante, teniendo en cuenta que esto ayuda

a identificar qué variables son importantes para captar el comportamiento del tráfico. Al seleccionar las características adecuadas, se puede aumentar la capacidad de los modelos para captar el comportamiento del tráfico y asegurar que sean lo más precisos posible al evaluar nuevas entradas de datos. A continuación, la Tabla 7 indica entre 9, 12, 14 y 20 características agrupadas.

Tabla 7

Conjunto de características

Grupo	Características
FG1	Dur, Proto, Dport, TotPkts, SrcPkts, DstPkts, TotBytes, SrcBytes, DstBytes
FG2	Dur, Proto, Dport, TotPkts, SrcPkts, DstPkts, TotBytes, SrcBytes, DstBytes, Rate, SrcRate, DstRate
FG3	Dur, SrcDur, DstDur, Proto, Dport, TotPkts, SrcPkts, DstPkts, TotBytes, SrcBytes, DstBytes, Rate, SrcRate, DstRate, sMeanPktSz, dMeanPktSz, sMaxPktSz, dMaxPktSz, sMinPktSz, dMinPktSz
FG4	Dur, SrcDur, DstDur, Proto, Dport, TotPkts, SrcPkts, DstPkts, TotBytes, SrcBytes, DstBytes, Rate, SrcRate, DstRate

Entrenamiento del modelo

Elección del modelo

Los modelos se seleccionan en base a dos tipos de algoritmos: explicativos y por conjuntos. En el primero se determina el impacto que tienen las variables independientes sobre el resultado, es decir, la variable dependiente. En el segundo, los resultados obtenidos son el producto de la combinación de predicciones entre varios modelos. También, se debe identificar si estos modelos son apropiados o no para una clasificación multiclase. Por lo tanto, como algoritmo explicativo se escoge DT. Mientras, que en los algoritmos basados en conjuntos, se opta por usar SVM, y RF (Alojail & Bhatia, 2020). La Tabla 8 indica las técnicas analizadas para la clasificación de ataques.

Tabla 8

Selección y evaluación de modelos de clasificación

Modelo ML	Ventaja	Desventaja	Ejecución
Naive Bayes	Implementación fácil	Pocos parámetros	Media
Random Forest	Alto rendimiento	Sobreajuste	Media
Regresión Logística	Salida variada	Supone	Rápido
SVM	Eficiente	Resultados bajos	Lento
Decision Tree	Alto rendimiento	Sobreajuste	Rápido
KNN	Deducción rápida	Encontrar N vecinos	Medio

Configuración de hiperparámetros

Una vez definidos los modelos, se dividen los datos en 80% para entrenamiento y 20% para prueba. Posteriormente, se utiliza GridSearchCV como técnica para elegir mejores hiperparámetros del modelo de aprendizaje automático. Esta técnica se aplica en todos los modelos y consta de dos componentes: GridSearch y CV. El primer componente permite

encontrar los parámetros más apropiados para el modelo. El segundo componente aprueba el modelo durante su entrenamiento, mediante la validación cruzada. En la Tabla 9 se muestran los hiperparámetros a buscar en los modelos establecidos.

Tabla 9

Búsqueda de hiperparámetros

Modelo	Hiperparámetro	Descripción
Decision Tree	criterion	Función para dividir los datos
	max_depth	Niveles de profundidad del árbol
	min_samples_split	Número mínimo de datos requerido para dividir un nodo
	min_samples_leaf	Número mínimo de muestras para dejar un nodo
	ccp_alpha	Valor para regular la complejidad de poda.
	splitter	Método para crear división en los nodos
Random Forest	n_estimators	Número de árboles
	criterion	Función para dividir los datos
	max_depth	Niveles de profundidad del árbol
	min_samples_split	Número mínimo de datos requerido para dividir un nodo
	min_samples_leaf	Número mínimo de muestras para dejar un nodo
	ccp_alpha	Valor para regular la complejidad de poda
SVM	C	Parámetro de penalización para la clasificación
	kernel	Núcleo para el modelo

Tabla 9 – Continúa

Modelo	Hiperparámetro	Descripción
	gamma	Coeficiente para el núcleo
	decision_function_shape	Función de comparación de muestras

Hiperparámetros definidos

Los hiperparámetros encontrados en cada uno de los modelos se aplican a cada conjunto de características establecido en la transformación de datos (ver tabla anterior). La Tabla 10 muestra los hiperparámetros encontrados para cada modelo.

Tabla 10

Hiperparámetros para los modelos clasificadores

Modelo	Hiperparámetro	Valor
	criterion	entropy
	max_depth	12
Decision	min_samples_split	2
Tree	min_samples_leaf	1
	ccp_alpha	0.0001
	splitter	best
	n_estimators	10
	criterion	entropy
Random	max_depth	12
Forest	min_samples_split	5
	min_samples_leaf	2
	ccp_alpha	0.0001
	Siguiente...	

Tabla 10 – Continúa

Modelo	Hiperparámetro	Valor
SVM	C	1000
	kernel	rbf
	gamma	scale
	decision_function_shape	ovo

Capítulo V

Resultados

Una vez entrenados los modelos de aprendizaje automático, en este capítulo se procede a interpretar los resultados obtenidos al ejecutar la clasificación multiclase con los algoritmos y conjunto de características previamente establecidos. Los resultados obtenidos reflejan que los modelos basados en árboles de decisión son una buena elección para alcanzar los resultados esperados. Sin embargo, se debe tener en cuenta el riesgo de sobreajuste o desajuste que pueden contraer debido a que los modelos pueden aprender de datos con ruido o que no son relevantes.

Evaluación del rendimiento del modelo

Para evaluar el desempeño de cada modelo, se consideran dos conceptos: métricas de evaluación o clasificación y la matriz de confusión o de predicción.

Métricas de evaluación

Uno de los pasos más importantes en los modelos de aprendizaje automático es su evaluación, es decir, determinar su rendimiento. Según el tipo de técnica aplicada, existen diferentes métricas de evaluación. En este caso, las métricas de clasificación permiten conocer el rendimiento alcanzado por un modelo durante su entrenamiento, las cuales son utilizadas para tomar decisiones objetivas y determinar su mejor aplicabilidad. Las métricas a analizar son:

- **Accuracy**: representa el total de las predicciones realizadas correctamente sobre el total de datos a predecir.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

- **Precision**: es el número de muestras que han sido detectadas correctamente como positivas.

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

- **Recall**: porción de muestras positivas que el modelo ha identificado correctamente.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

- **F1-Score**: promedio de ponderación entre precision y recall que buscan la perfercción del modelo.

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

A continuación, se muestra la Tabla 11 con los resultados obtenidos en cada conjunto de características (ver Tabla 7). En base al accuracy se puede determinar que los modelos DT y RF han sido los mejores puntuados.

Tabla 11

Resultados de las métricas de evaluación

Modelo	Accuracy	Precision	Recall	F1-Score
		FG1		
DT	99.34%	99.34%	99.34%	99.34%
RF	99.31%	99.33%	99.31%	99.32%
SVM	96.83%	96.89%	96.83%	96.83%
		FG2		
DT	99.28%	99.29%	99.28%	99.28%
	Siguiente...			

Tabla 11 – Continúa

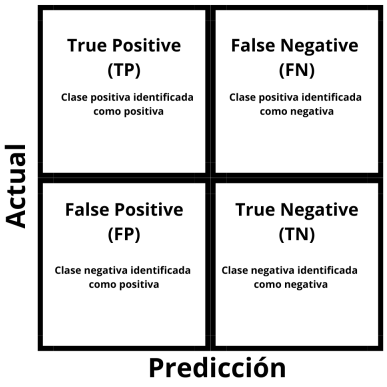
Modelo	Accuracy	Precision	Recall	F1-Score
RF	99.36%	99.38%	99.36%	99.37%
SVM	96.91%	96.97%	96.91%	96.91%
		FG3		
DT	99.80%	99.80%	99.80%	99.80%
RF	99.76%	99.76%	99.76%	99.76%
SVM	99.49%	99.49%	99.49%	99.49%
		FG4		
DT	99.25%	99.26%	99.25%	99.25%
RF	99.32%	99.34%	99.32%	99.32%
SVM	97.06%	97.11%	97.06%	97.06%

Matriz de confusión

La matriz de confusión muestra las predicciones del modelo frente a las etiquetas de verdad. Cada fila pertenece a una clase predecida, mientras que cada columna es una clase real. La Figura 16 transparenta algunas terminologías para su interpretación.

Figura 16

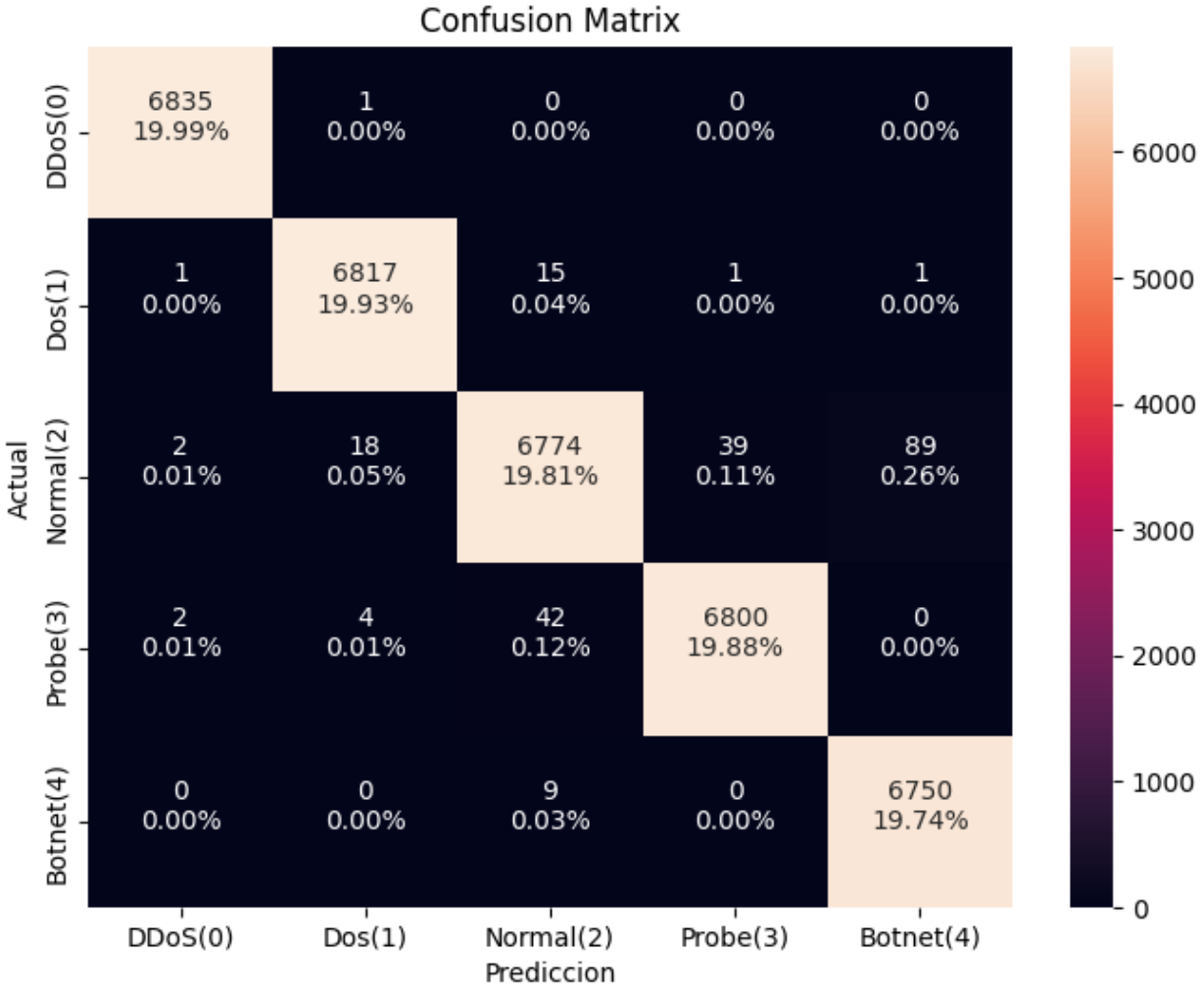
Estructura de la matriz de confusión



En este apartado se consideran dos matrices de confusión por grupo de características. En la Figura 17 se muestran los resultados del modelo DT para FG1, donde se puede comprobar que la clase Normal y Probe clasifican un tráfico mínimo como otras clases. Por otro lado, la Figura 18 muestra que en el modelo SVM en las clases DDoS, DoS y Normal, se clasifica tráfico en otras clases, mientras que la clase Probe reduce la cantidad de identificación de este mismo tipo de tráfico como Normal.

Figura 17

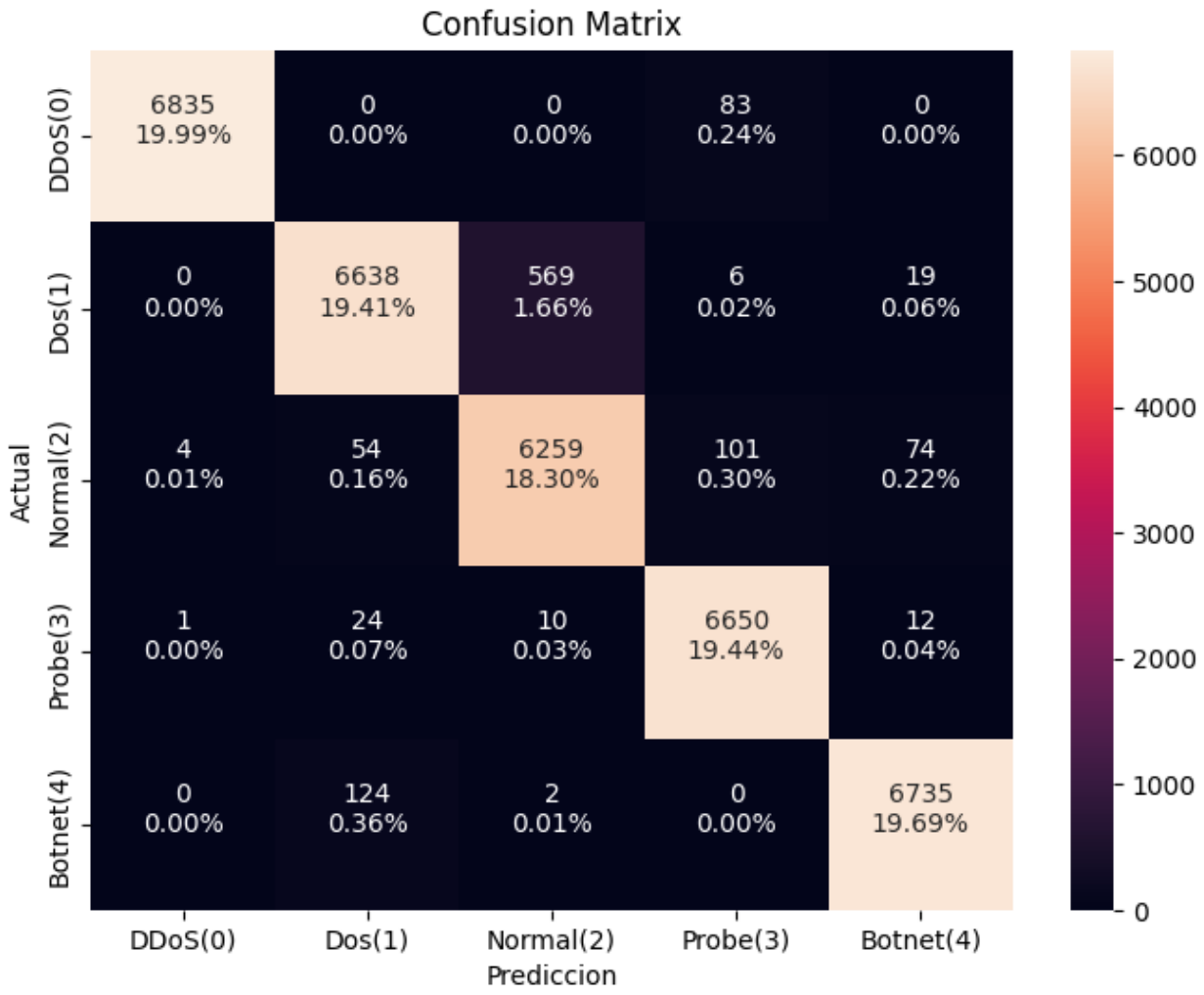
Matriz de confusión DT en FG1



En la Figura 19 se muestra lo correspondiente al modelo DT para FG2, donde la clase Botnet y DDoS se destacan por clasificar correctamente sus entradas. Por su parte, la clase DoS contiene un total de 35 muestras que han sido clasificadas de forma incorrecta. La clase Probe contiene 48 muestras que el modelo ha clasificado en su mayoría como tráfico Normal. Por último, el tráfico Normal tiene un total de 158 muestras que no son identificadas como positivas.

Figura 18

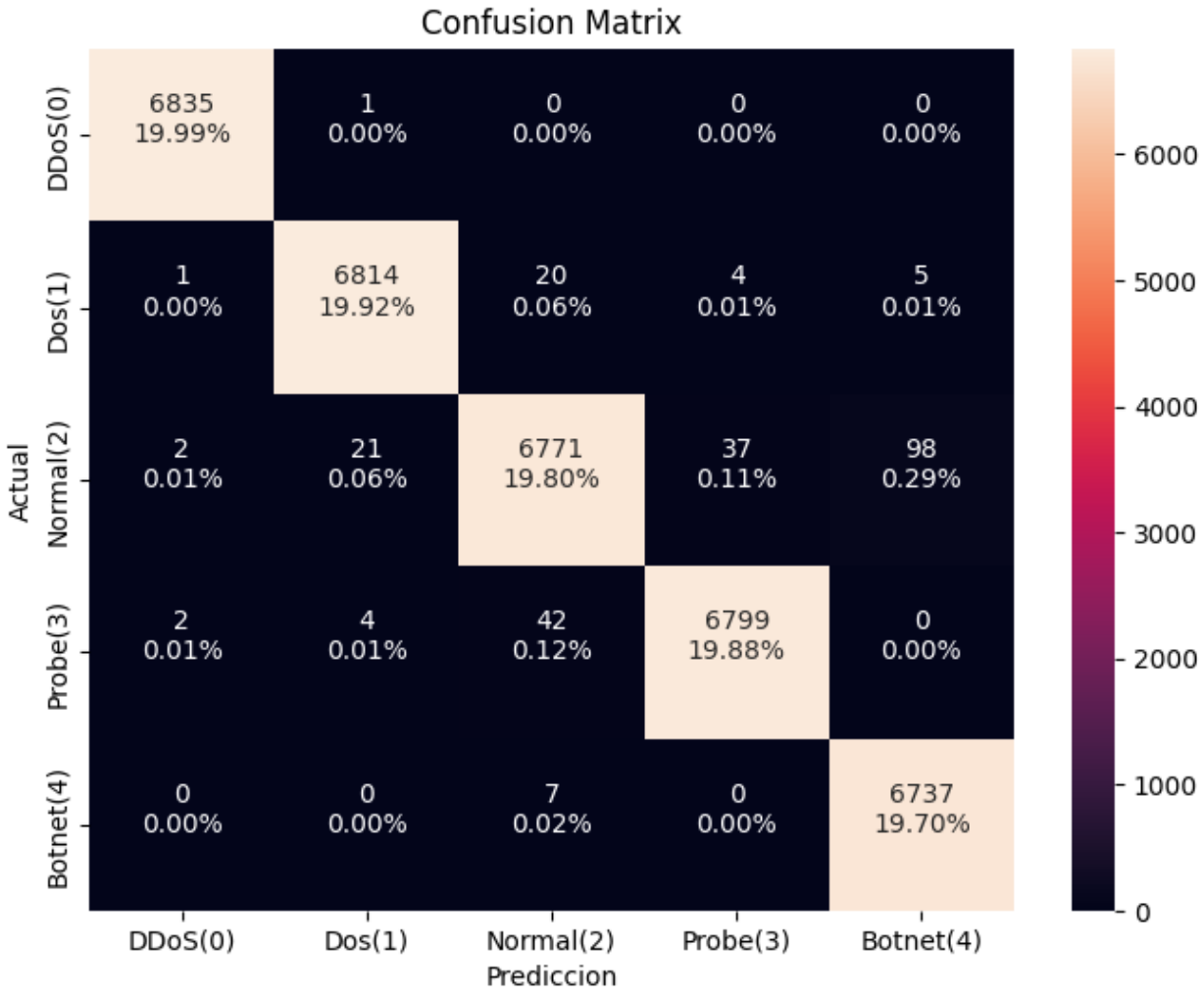
Matriz de confusión SVM en FG1



En el mismo grupo de características, la Figura 20 corresponde al modelo RF, cuyos resultados solo resaltan la clase DDoS por clasificar todas sus entradas correctamente. En la clase Botnet, clasifica solo cinco entradas de forma incorrecta, nueve en DoS y seis en Probe. En cambio, en la clase Normal, la mayoría de las entradas clasificadas incorrectamente pertenecen a las clases Probe y Botnet.

Figura 19

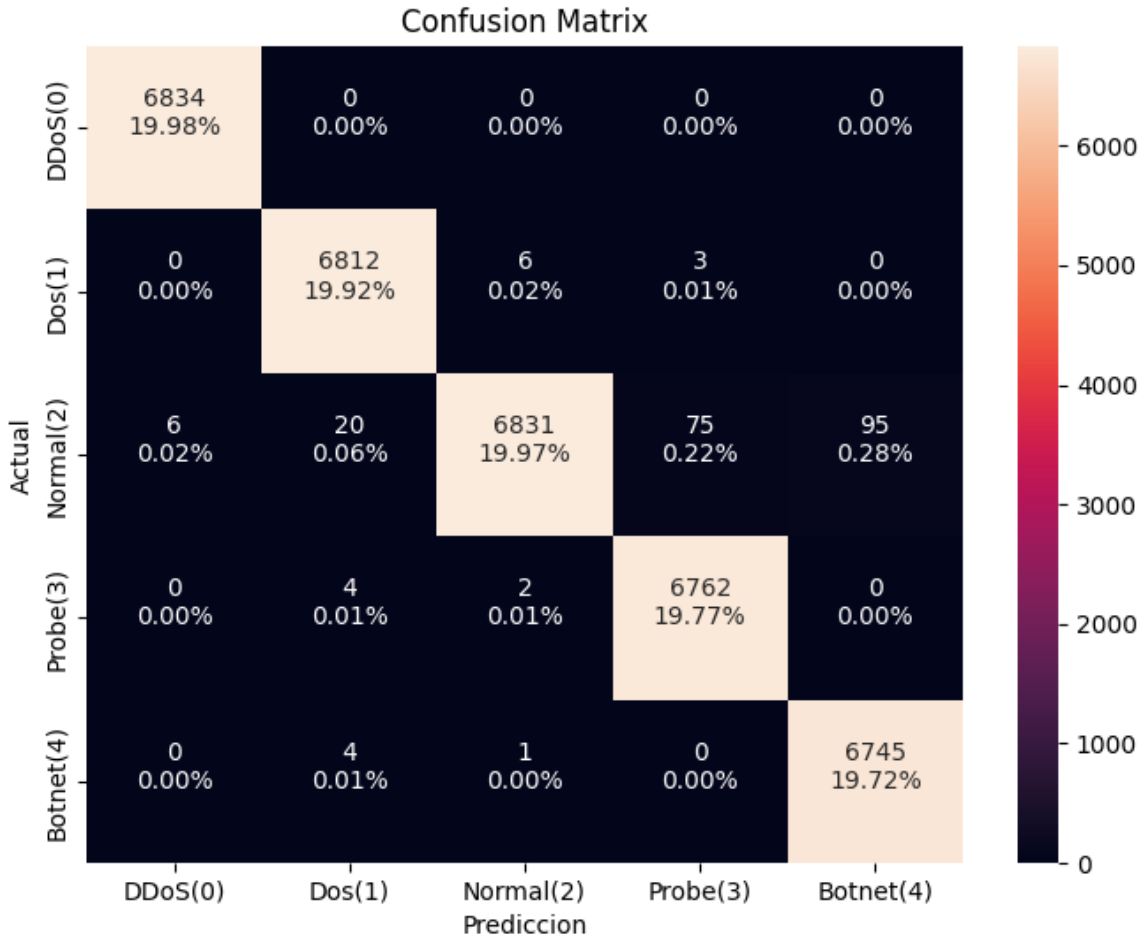
Matriz de confusión DT en FG2



La Figura 21 corresponde al modelo DT para FG3, donde gran cantidad de tráfico de cada clase es clasificado como falso negativo. La clase Botnet, Probe y DoS tienen cinco, seis y nueve muestras clasificadas incorrectamente, respectivamente. La clase Normal contiene 75 muestras clasificadas erróneamente como Probe y 95 como Botnet. Las otras 26 muestras se clasifican incorrectamente como DoS y DDoS.

Figura 20

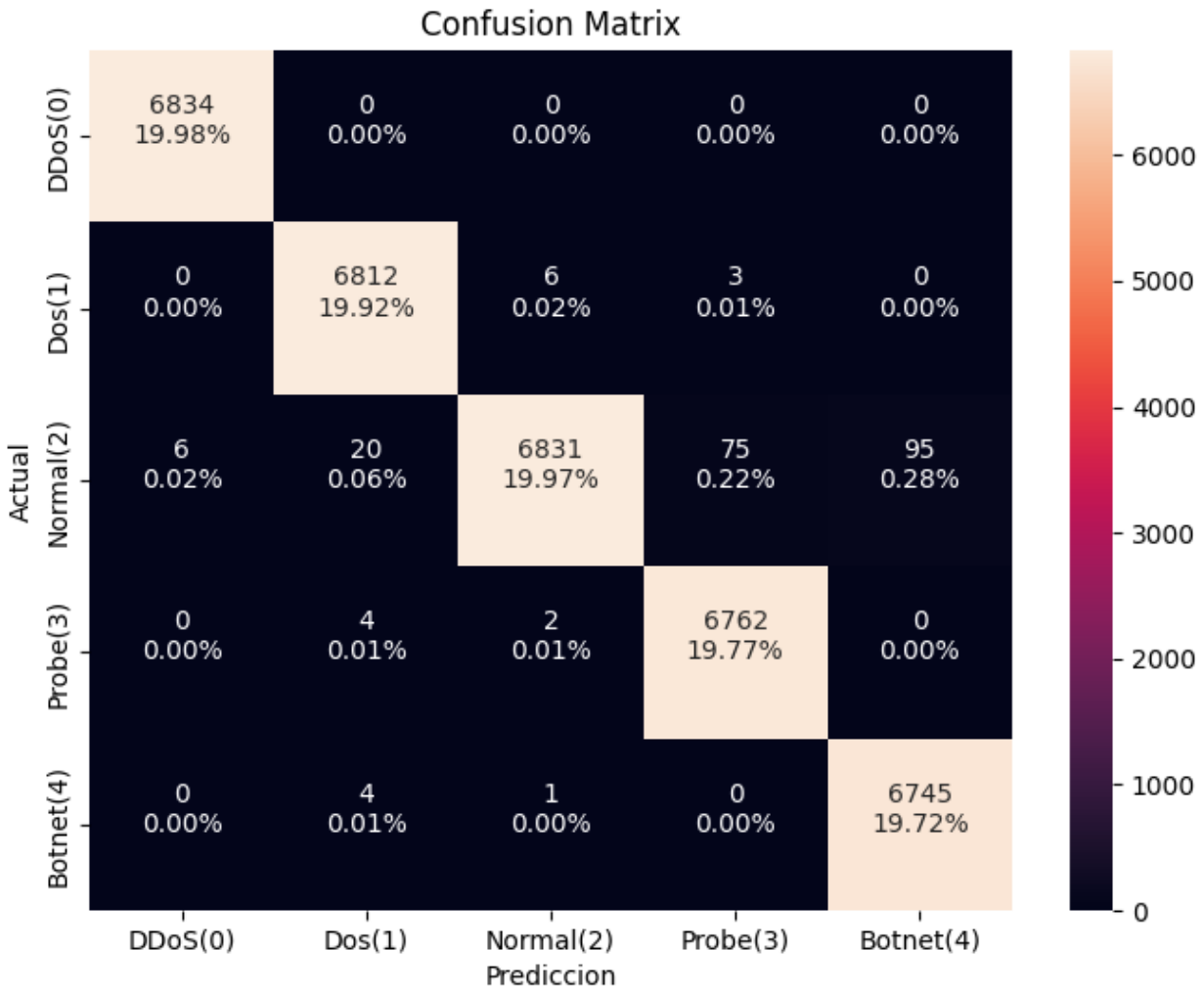
Matriz de confusión RF en FG2



En el mismo grupo FG3, en la Figura 22 del modelo SVM se puede comprobar que su rendimiento ha mejorado en comparación a los dos modelos anteriores del conjunto de características. La novedad que presenta este modelo es que la clase Normal ha sido reducida significativamente en su clasificación de entradas como negativas. En cambio, la clase DDoS aumentó la tasa de clasificaciones negativas para el tipo Probe. En cuanto al resto de clases, Botnet, DoS y Probe, se clasificaron 1, 8 y 12 muestras incorrectamente, respectivamente.

Figura 21

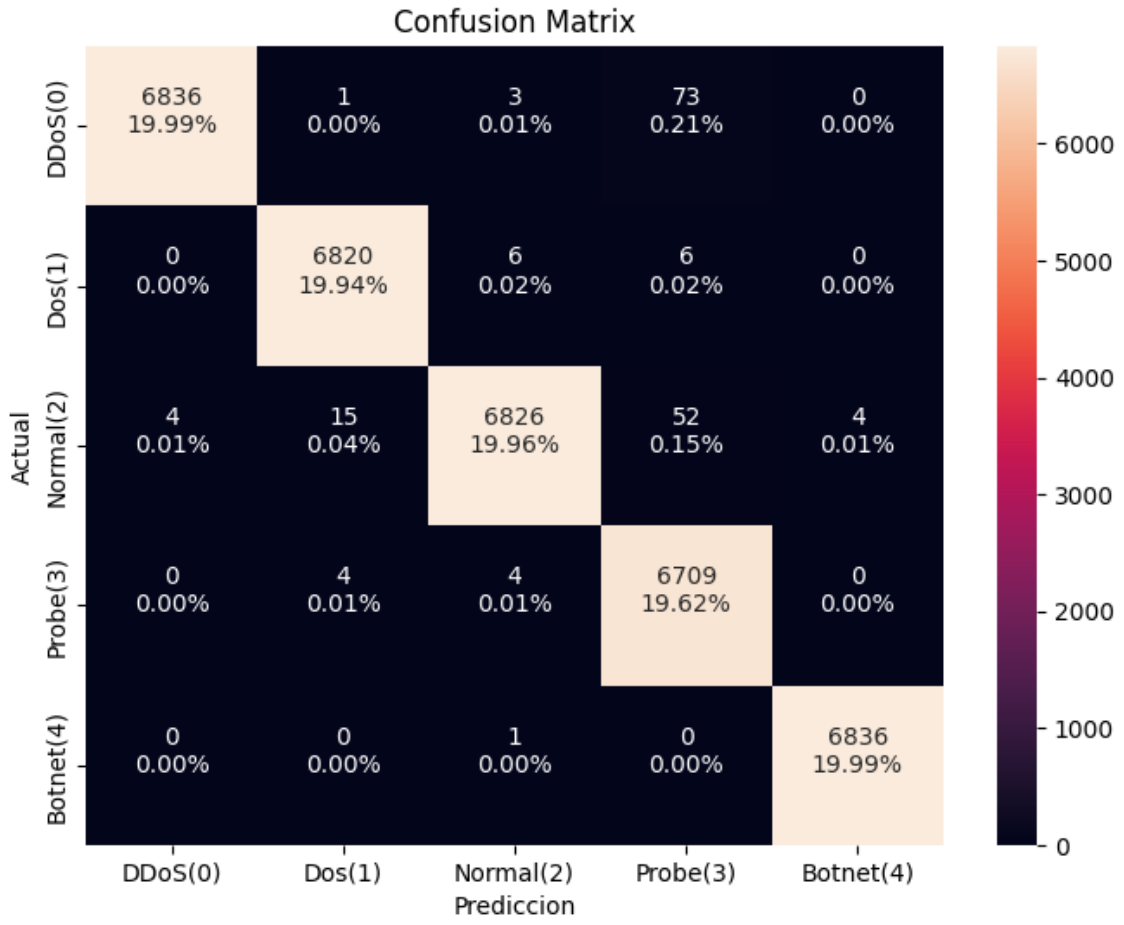
Matriz de confusión DT en FG3



En el grupo FG4, Figura 23 muestra los resultados del modelo RF. Se destaca que la clase Normal presenta un elevado número de falsos negativos, donde 175 entradas son clasificadas erróneamente como tráfico Botnet o Probe. Por otro lado, la clase DDoS es clasificada correctamente. En cuanto a las clases Botnet, Probe y DoS, se identifican pocas muestras que no son clasificadas correctamente como positivas.

Figura 22

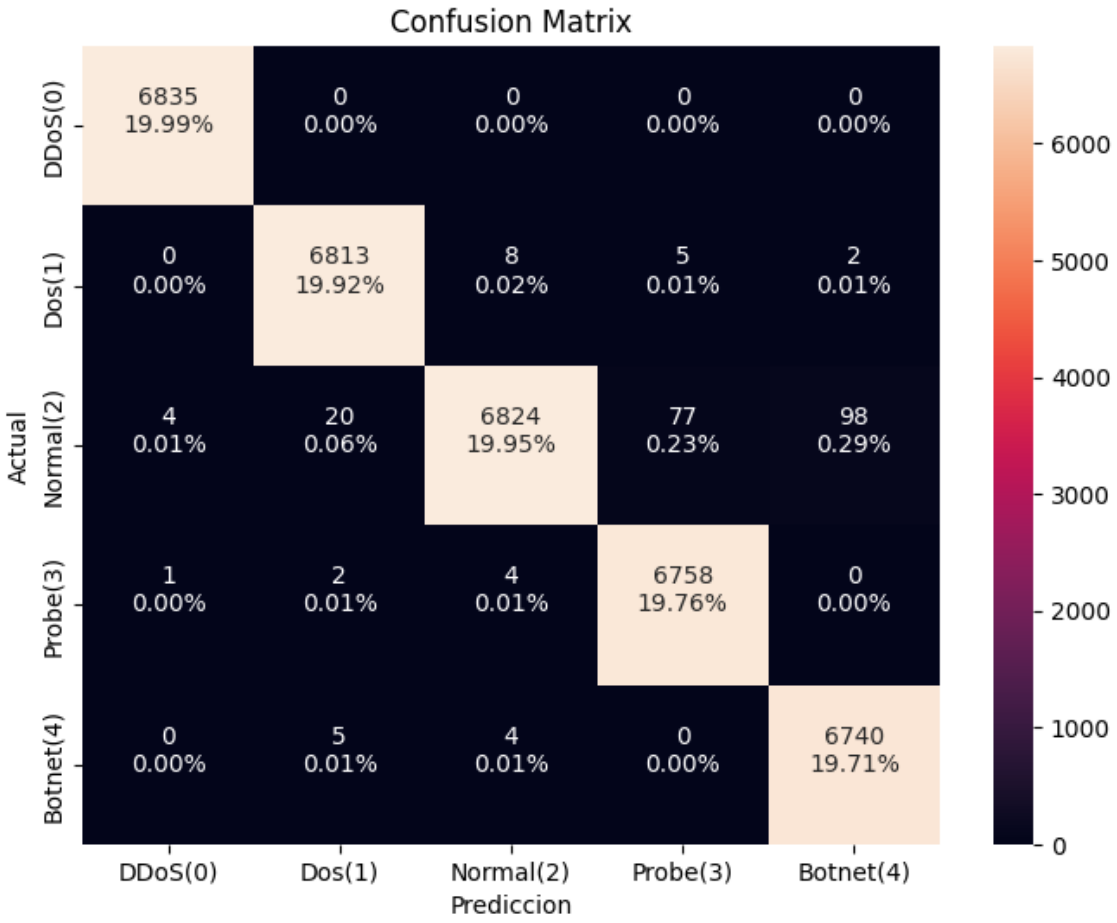
Matriz de confusión SVM en FG3



Finalizando en FG4, la Figura 24 muestra el modelo SVM donde la mayoría de las clases de tráfico clasifican sus muestras como falsos negativos a más de una clase. En el caso de la clase DDoS, sus entradas son identificadas en su mayoría como Probe. Para DDoS, existe un porcentaje del 1.61% donde las clases han sido identificadas como Normal. En cambio, la clase Normal tiene una clasificación de entradas positivas muy baja en comparación con las demás clases. La clase Probe contiene 28 muestras identificadas como DoS, mientras que Botnet clasifica 121 muestras como negativas para el tipo DoS.

Figura 23

Matriz de confusión RF en FG4



Validación del modelo

Para validar los modelos entrenados, se utilizan: validación cruzada, curva de validación y curva ROC.

Validación cruzada

Para evaluar el rendimiento general de los modelos entrenados y reducir el riesgo de sobreajuste durante el entrenamiento, se aplicó la técnica de validación cruzada. Los resultados de cada uno de los modelos por grupo de características se muestran en la

Figura 24

Matriz de confusión SVM en FG4

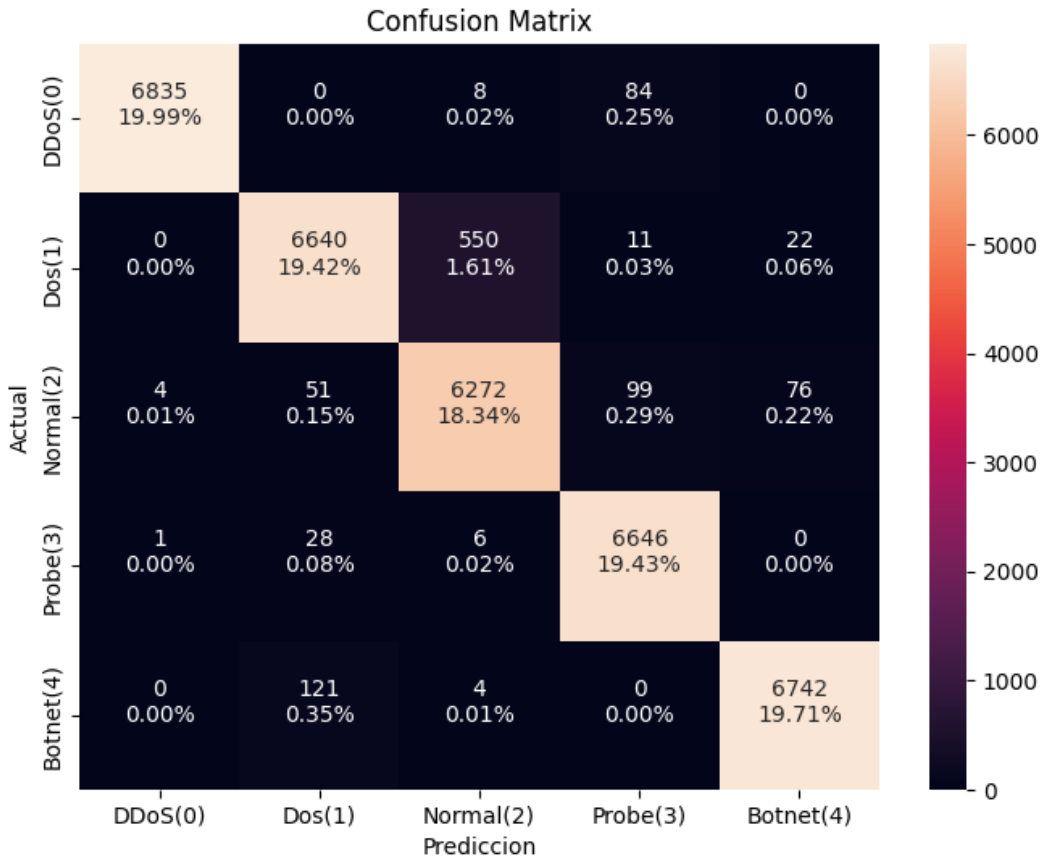


Tabla 12.

Tabla 12

Resultados de la validación cruzada

Modelo	Accuracy	Precision	Recall	F1-Score
		FG1		
DT	99.28%	99.29%	99.28%	99.28%
RF	99.31%	99.33%	99.31%	99.32%
	Siguiente...			

Tabla 12 – Continúa

Modelo	Accuracy	Precision	Recall	F1-Score
SVM	96.83%	96.89%	96.83%	96.83%
		FG2		
DT	99.25%	99.26%	99.25%	99.25%
RF	99.30%	99.32%	99.30%	99.30%
SVM	97.09%	97.09%	97.09%	97.04%
		FG3		
DT	99.76%	99.76%	99.76%	99.76%
RF	99.75%	99.75%	99.75%	99.75%
SVM	99.50%	99.50%	99.50%	99.50%
		FG4		
DT	99.25%	99.26%	99.25%	99.25%
RF	99.25%	99.27%	99.25%	99.26%
SVM	97.06%	97.11%	97.06%	97.06%

Curva de validación

La curva de validación comprueba si el modelo está sobreentrenado y en base a los datos permite encontrar el mejor modelo que se ajuste a las necesidades. Cada gráfica contiene dos curvas: una para la puntuación del conjunto de entrenamiento y el otro para la validación cruzada realizada cinco veces. Así que, por cada grupo de características, se analizará dos modelos, aquellos que tienen la puntuación de accuracy más alta y baja.

En el grupo FG1, la Figura 25 pertenece al modelo RF cuyas líneas de entrenamiento y validación son las mismas. En cambio, la Figura 26 corresponde al modelo SVM donde existen dos puntos sensibles aproximadamente en el 92% y 95%.

Figura 25

Curva de validación de RF en FG1

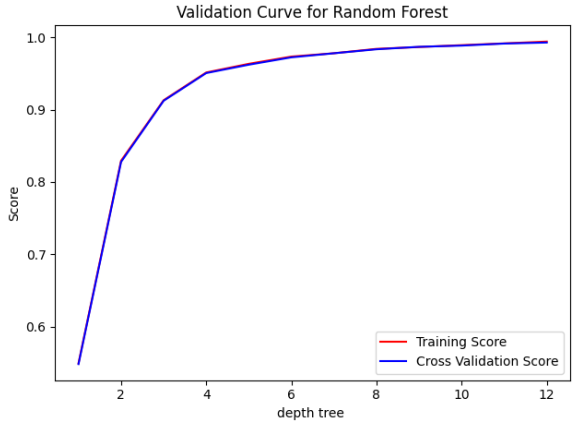
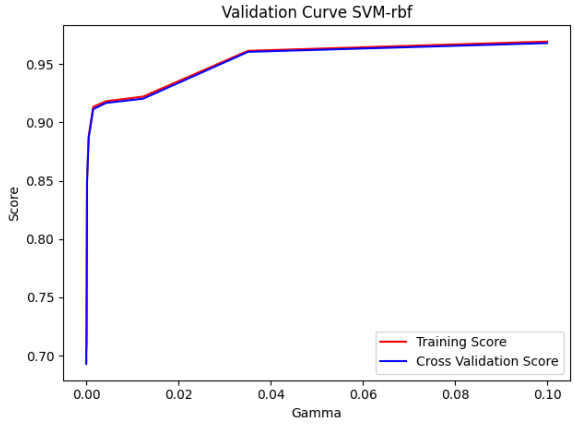


Figura 26

Curva de validación de SVM en FG1



Para el grupo FG2, la Figura 27 del modelo RF las líneas de entrenamiento y validación son idénticas. Además, existen tres puntos donde los parámetros del modelo cambian, es decir, entre el 85%, 92% y 95%. En cambio, la Figura 28 del modelo SVM llega a punto alto de 95% y luego se mantiene ligeramente a un valor superior del porcentaje mencionado.

Figura 27

Curva de validación de RF en FG2

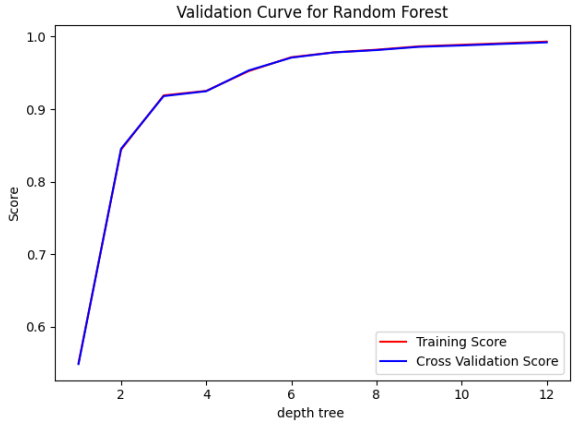
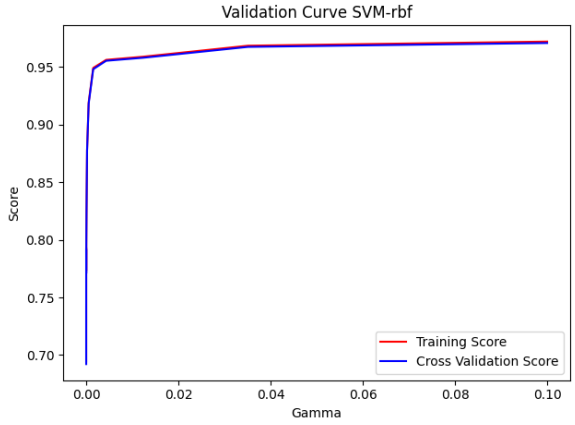


Figura 28

Curva de validación de SVM en FG2



Para el grupo FG3, la Figura 29 le corresponde al modelo DT las líneas de entrenamiento y validación son iguales. Existen tres puntos donde cambia el modelo, aproximadamente en el 80%, 91% y 98%. La Figura 30 del modelo SVM en el punto aproximado del 99% se identifica un cambio donde las líneas de entrenamiento y validación dejan de ser las mismas.

Figura 29

Curva de validación de DT en FG3

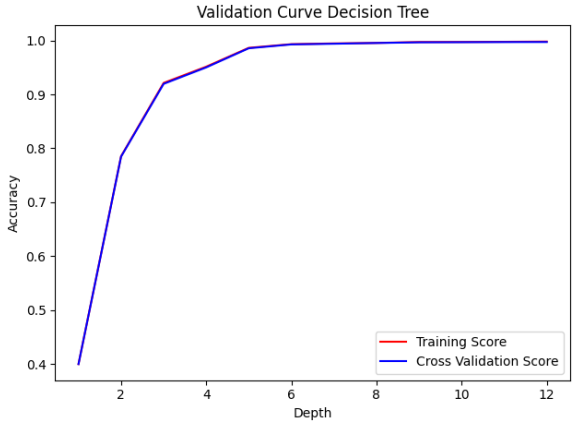
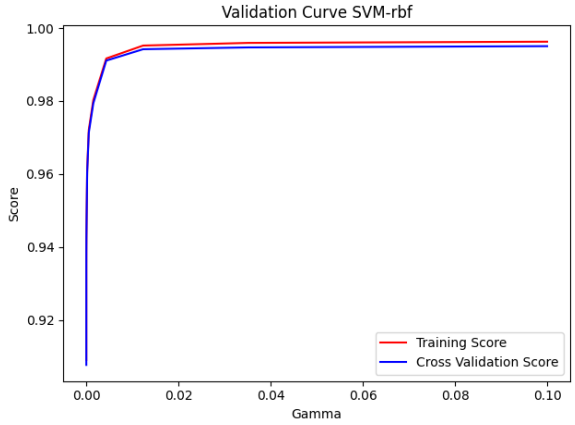


Figura 30

Curva de validación de SVM en FG3



Por último, en el grupo FG4, la Figura 31 del modelo DT mantiene características similares a la gráfica del FG3. En este caso, a partir del 90% de la gráfica del modelo, la curva de validación comienza a estabilizarse. La Figura 32 del modelo SVM en el punto aproximado del 96% las curvas de entrenamiento y validación se sesgan al final de la gráfica.

Figura 31

Curva de validación de DT en FG4

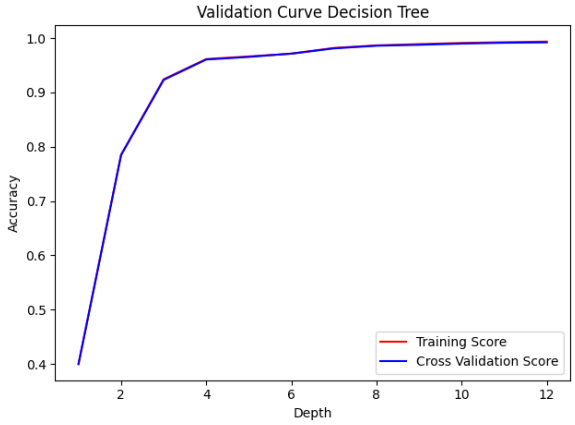
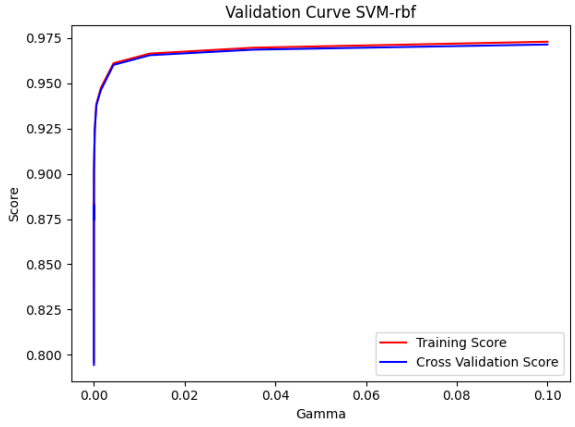


Figura 32

Curva de validación de SVM en FG4



Curva ROC

La curva ROC muestra la relación entre verdaderos positivos (TP) y falsos positivos (FP) en el eje Y y X respectivamente. El área que obtiene bajo la curva ROC permite conocer si un modelo clasificador es relativamente bueno. En base a las curvas de validación anteriores, se mostrará la curva ROC cuyos resultados son obtenidos con el

conjunto de prueba y no con el conjunto entrenamiento.

En el primer grupo FG1, la Figura 33 del modelo RF muestra un excelente rendimiento para nuevas entradas, aquí el área más baja es de la clase Probe. Por otro lado, la Figura 34 representa al modelo SVM con diferentes variaciones en cuanto al área obtenida por cada una de las clases de tráfico. La clase Normal tiene el área más baja con el 95.33%, mientras que la clase DDoS obtiene el área más alta con el 99.81%.

Figura 33

Curva ROC de DT en FG1

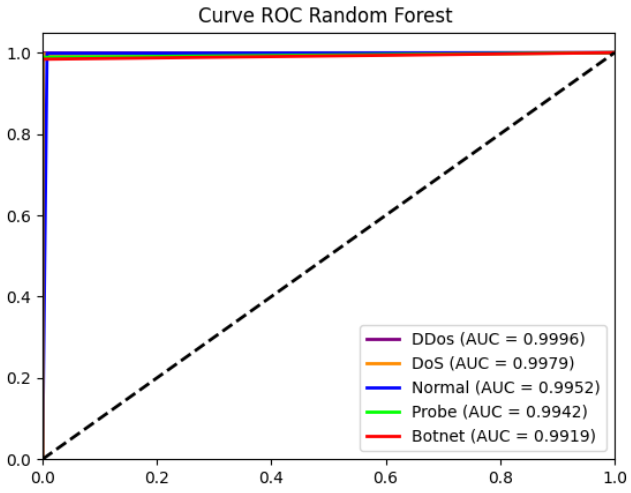
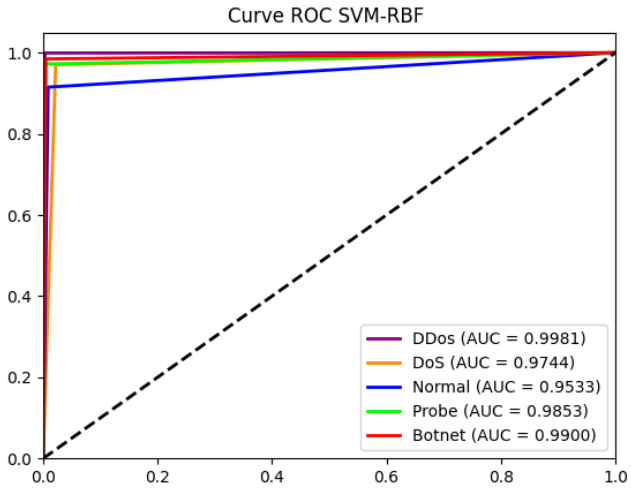


Figura 34

Curva ROC de SVM en FG1



Para el grupo FG2, la Figura 35 del mismo modelo RF muestra mejores resultados en cuanto a las áreas del anterior modelo en FG1. En cambio, la Figura 36 el modelo SVM no muestra cambios significativos con respecto al mismo modelo del grupo FG1.

Figura 35

Curva ROC de DT en FG2

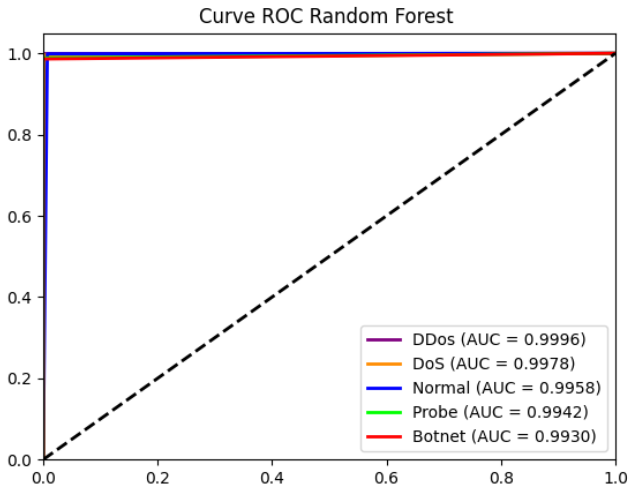
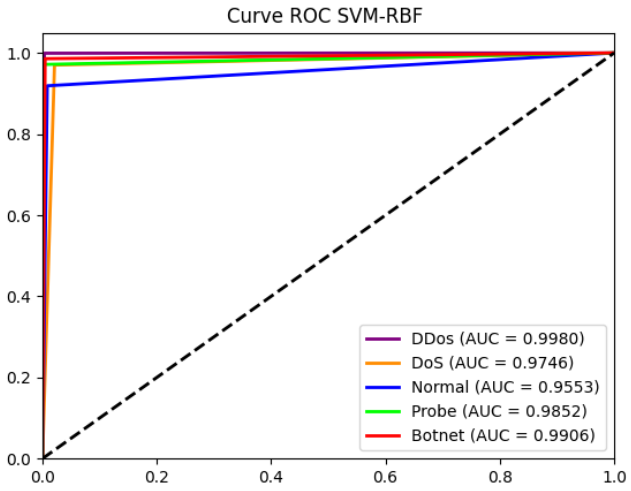


Figura 36

Curva ROC de SVM en FG2



En el grupo FG3, la Figura 37 el modelo DT muestra resultados excelentes sobre todo en la clase Botnet. En cuanto a la Figura 38 el modelo SVM evidencia mejoras con respecto a los dos modelos anteriores de FG1 y FG2.

Figura 37

Curva ROC de DT en FG3

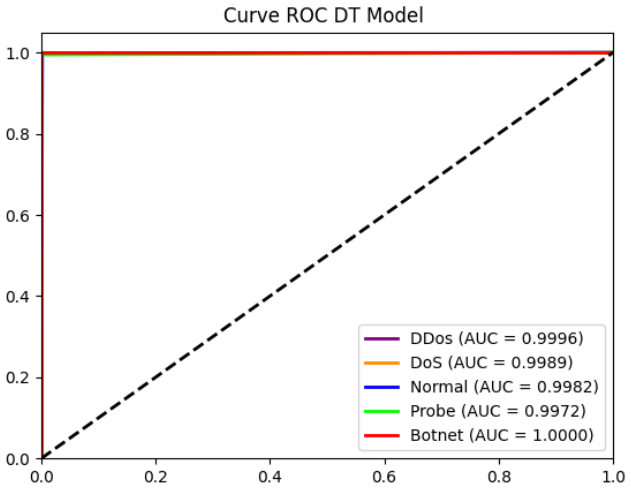
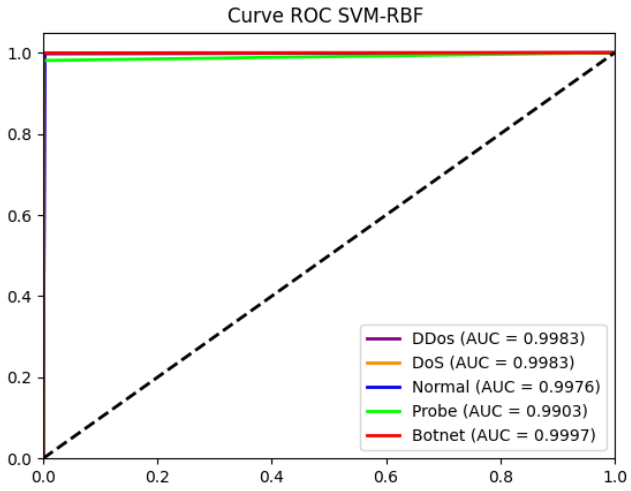


Figura 38

Curva ROC de SVM en FG3



Para finalizar, en el grupo FG4, la Figura 39 del modelo DT las áreas se encuentran por encima del 99.50% en su mayoría a excepción de la clase Normal. En cambio, la Figura 40 del modelo SVM las áreas de la clase DoS y Normal son las más bajas.

Figura 39

Curva ROC de DT en FG4

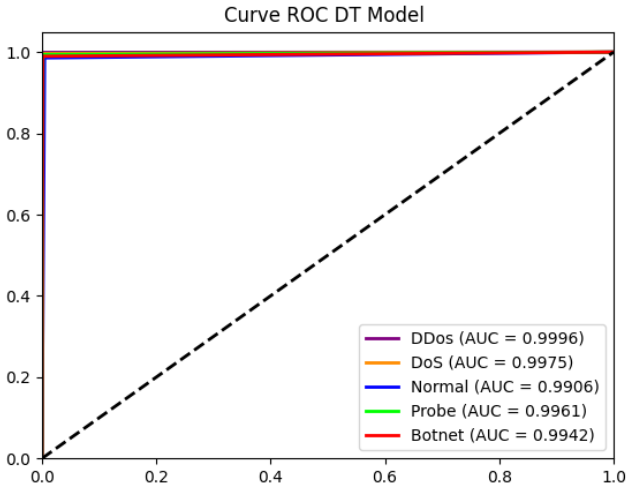
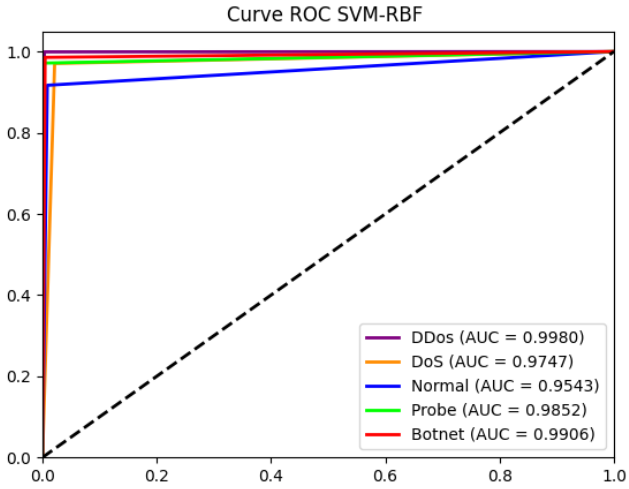


Figura 40

Curva ROC de SVM en FG4



Con la información presentada, se identifica que los modelos DT y RF presentan un alto rendimiento. En la tabla 13 se muestran los mejores modelos en base a las estadísticas de validación cruzada por cada conjunto de características.

Tabla 13

Mejores modelos por conjunto de características

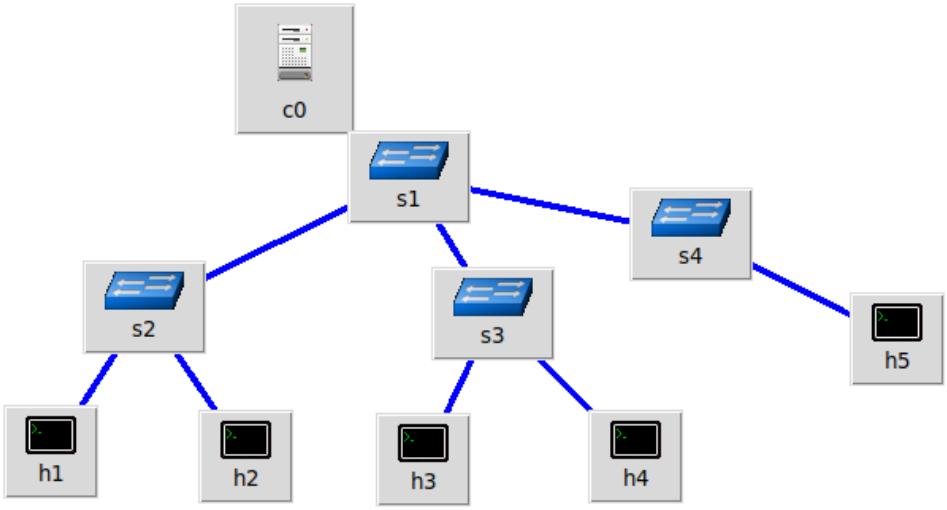
Grupo	Modelo	Accuracy	Precision	Recall	F1-Score
FG1	RF	99.31%	99.33%	99.31%	99.32%
FG2	RF	99.30%	99.32%	99.30%	99.30%
FG3	DT	99.76%	99.76%	99.76%	99.76%
FG4	RF	99.25%	99.27%	99.25%	99.26%

Despliegue del modelo

El modelo desarrollado se implementa utilizando Mininet y Flask. Mininet es un entorno virtual que simula la ejecución real de switches, kernel y aplicaciones bajo línea de comandos disponiendo de tres tipos de dispositivos: controlador, switch y host (Mininet, 2023). Con esta herramienta se crea una topología basada en árbol, utilizando cuatro switches, cinco hosts y un controlador. La Figura 41 muestra la topología implementada para las pruebas del modelo.

Figura 41

Topología en Mininet



Por otro lado, Flask es un framework para desarrollar aplicaciones web, el cual maneja una arquitectura minimalista, permitiendo mejorar la velocidad de las aplicaciones en el lado del servidor (Flask, 2023). Por lo tanto, se lo utiliza para desarrollar una Interfaz de Programación de Aplicaciones (API) para que el controlador y el modelo puedan interactuar. El algoritmo 1 muestra el funcionamiento del controlador y el modelo.

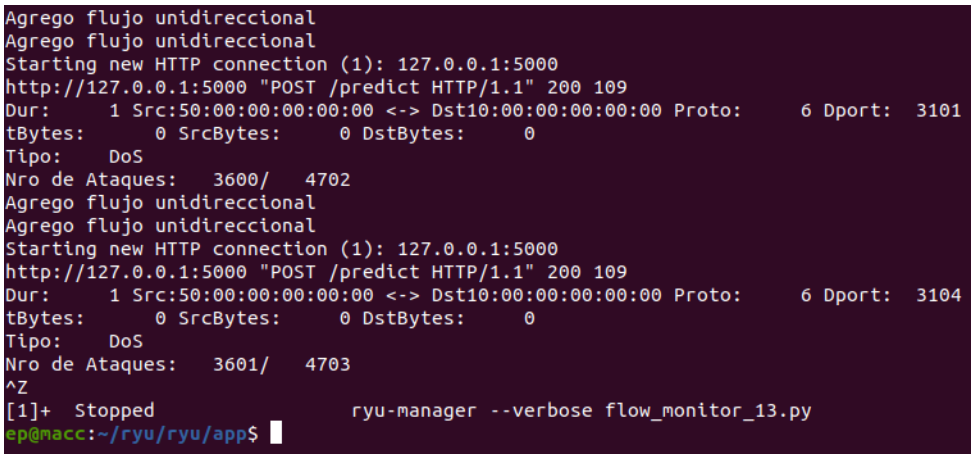
Algoritmo 1 Clasificación de flujos

```
1: procedure CONEXION CONTROLADOR Y APLICACION DE MODELO DE CLASIFICACION
Require: IP del servidor de aplicacion
2:     procedure GESTIÓN DE ESTADÍSTICAS DE FLUJO
Require: Controlador envia solicitud de estadisticas de flujo al Switch
3:         Controlador ← Mensaje
4:         while Mensaje = TCP or UDP do
5:             ListaMensajes ← Mensaje
6:             if len(ListaMensajes)%2 == 0 then
7:                 for Caracteritica in ListaMensajes do
8:                     if Duracion ≤ 20s then
9:                         Flujo1 ← Caracteristica
10:                        Flujo2 ← Caracteristica + 1
11:                        FlujoBidireccional ← Flujo1 + Flujo2
12:                        POST ← FlujoBidireccional
13:                        Modelo ← POST
14:                        Respuesta ← Modelo
15:                        Controlador ← Respuesta
16:                    end if
17:                end for
18:            end if
19:        end while
20:    end procedure
21: end procedure
```

La Figura 42 muestra el funcionamiento del modelo de clasificación de ataques, en el cual del total tráfico de DoS ejecutado ha identificado 3601 muestras como positivas. Para generar este tráfico se utiliza la herramienta hping3, usando el comando hping3 -S –faster -d 100 -p 80 192.168.50.5.

Figura 42

Clasificación de tráfico de tipo DoS



La Figura 43 muestra la clasificación de tráfico Normal en el cual se utilizó el comando wget 192.168.50.5/index.html para hacer una petición al servidor HTTP. En esta demostración, al identificar el tráfico normal, no se clasifica como un ataque.

La Figura 44 presenta el resultado de la clasificación de tráfico Probe en el que de 2402 flujos, son clasificados 2225 muestras positivas. Para este tipo de ataque se utilizó Nmap empleando el comando nmap 192.168.50.5.

Figura 43

Clasificación de tráfico de tipo Normal

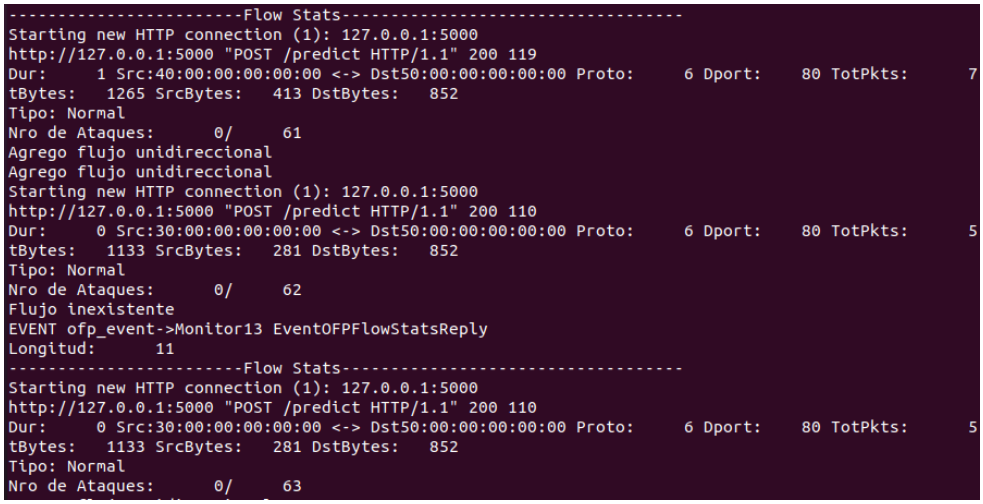
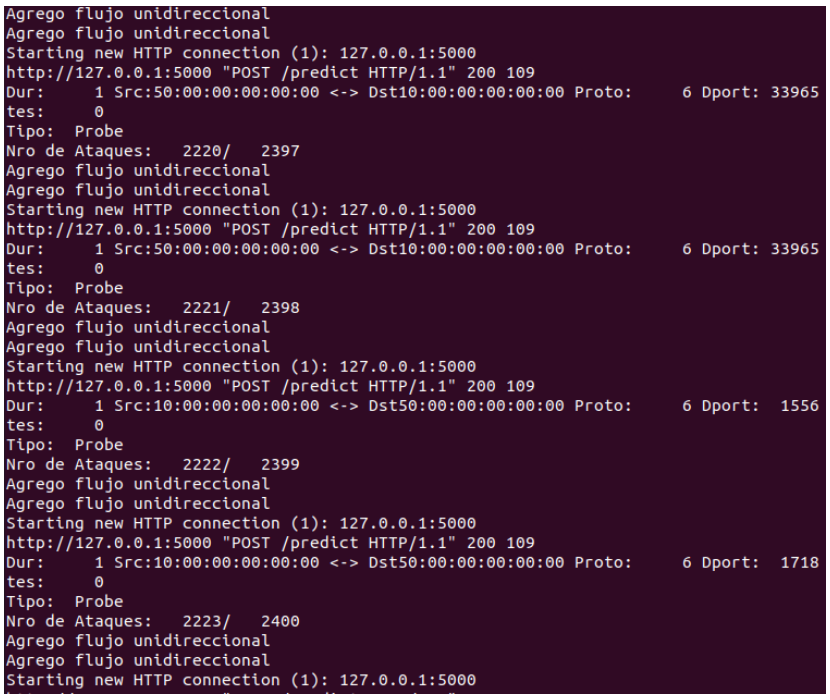


Figura 44

Clasificación de tráfico de tipo Probe



Resultados de la implementación

La Tabla 14 muestra los resultados encontrados en la implementación del modelo de clasificación de ataques, únicamente considerando los conjuntos de características FG1 y FG2. Si bien los resultados reflejados no son los esperados, SVM resulta ser el mejor modelo para clasificar tráfico normal y de ataques con 9 y 12 características. Para FG3 y FG4 no se desarrollaron las pruebas respectivas, debido a la dificultad para extraer las características en el controlador Ryu. Así mismo, para evaluar los resultados no se consideraron las clases Botnet y DDoS, aunque existen herramientas como TCPreplay para generar tráfico, se encontraron limitaciones por la información de la red en la que fueron generados los archivos PCAP.

Tabla 14

Accuracy de la implementación del modelo

Grupo	DT	RF	SVM
FG1	20.02%	21.45%	27.33%
FG2	24.14%	23.97%	43.28%

Capítulo VI

Conclusiones

En este trabajo se desarrollaron tres modelos de aprendizaje automático supervisados para la clasificación de tráfico normal, DoS, DDoS, probe y botnet. Las técnicas utilizadas fueron DT, RF y SVM, las cuales se entrenaron con 136800 muestras de flujo. El conjunto de datos utilizado para entrenar los modelos resultó de la combinación entre InSDN y CTU-13, para el tratamiento de los datos se aplicó las metodologías KDD y EDA. Cada modelo se evaluó utilizando un subconjunto de 34200 muestras. Los tres modelos fueron seleccionados en función de una evaluación de parámetros previa, basada en trabajos relacionados.

Los trabajos relacionados se obtuvieron a partir de una SLR sobre estudios que utilizan técnicas de aprendizaje automático orientadas a la seguridad de las redes SDN. Existen diversas ideas para desarrollar modelos que puedan clasificar ataques de tráfico en SDN; muchos de estos estudios se centran en la detección de tráfico, es decir, en su mayoría manejan dos tipos de tráfico: normal y de ataque. Por lo general, el ataque de tráfico se centra únicamente en DDoS. Además, los conjuntos de datos para sus desarrollos no son de origen SDN y son ambiguos. Sin embargo, existen autores que han generado su propio dataset.

Para la implementación del modelo de clasificación de ataques, se utilizó Mininet, Ryu y Flask. En Mininet se desarrolló la topología, la cual representa el plano de datos. En cambio, Ryu es el controlador que permite gestionar el tráfico de la red mediante el desarrollo de aplicaciones ejecutadas por el mismo controlador. Por último, Flask permite desplegar el modelo de clasificación de ataques y escenifica el plano de aplicación. El controlador tiene una constante interacción entre el plano superior e inferior mediante las APIs.

Los resultados encontrados durante el entrenamiento de los modelos DT, RF y SVM fueron del 99.76%, 99.31% y 99.50% de precisión, respectivamente. Estos hallazgos fueron obtenidos a partir de la validación del modelo y no directamente del resultado del entrenamiento, como se realiza en algunos trabajos relacionados. Además, se implementó el modelo en un entorno de emulación, mediante el cual se pudo evidenciar la clasificación de ataques con una precisión del 43.28%, 24.14% y 23.97% para los modelos SVM, DT y RF, respectivamente.

Recomendaciones

Es fundamental analizar si los datos de origen se extraen o publican correctamente, ya que los resultados finales que se esperan obtener comienzan desde ese punto. Además, es importante considerar diversas técnicas para el preprocesamiento y la limpieza de los datos, así como identificar los distintos tipos de datos que ofrece el conjunto de datos. Durante los procesos de minería de datos, se utilizaron las metodologías KDD y EDA, cuya combinación resultó muy efectiva para el tratamiento de los datos.

Para la selección de una técnica de aprendizaje automático, es necesario conocer propuestas de otros estudios y conocer su funcionamiento. También es necesario conocer que tipos de datos aceptan los algoritmos y si estos son sensibles al ruido de los datos, sobreajuste o desajuste. Además, se deben considerar las características y objetivos específicos del problema a resolver, así como también, la disponibilidad de datos y recursos computacionales.

Para el despliegue de un modelo de clasificación de tráfico, se recomienda conocer las características que ofrece cualquier controlador. De igual manera, es imprescindible conocer el funcionamiento que ofrece cada uno de los módulos, clases, eventos y APIs del controlador, dado que para el modelo con cualquier tipo de aprendizaje, la aplicación en el controlador es un requisito importante, porque desde allí se extraen las características que

son entrada del modelo. Además, una aplicación debe ser programada de manera óptima para aumentar la velocidad del controlador en el procesamiento de tráfico. Se debe evitar desplegar el modelo dentro del entorno del controlador, puesto que el modelo tiende a colapsar.

Trabajos futuros

Para mejorar los resultados del modelo de clasificación y realizar su implementación en una red SDN, se podría aumentar el conjunto de datos. Además, se debería analizar la existencia de nuevas características en distintas versiones de controladores open source, que puedan servir de entrada del modelo.

Una de las posibles mejoras para trabajos futuros es la aplicación de técnicas de aprendizaje profundo para entrenar el modelo de clasificación de ataques, lo que permitiría detectar patrones en los datos de forma más eficiente y precisa. El uso de estas técnicas podría mejorar significativamente el nivel de predicción del modelo. Sin embargo, se debe tener en cuenta la disponibilidad de recursos computacionales para procesar diferentes flujos de tráfico simultáneamente.

Referencias

Ahmad, S., & Mir, A. H. (2021). Scalability, Consistency, Reliability and Security in SDN Controllers: A Survey of Diverse SDN Controllers. *JOURNAL OF NETWORK AND SYSTEMS MANAGEMENT*, *29*(1). https://doi.org/10.1007/s10922-020-09575-4

Ahuja, N., Singal, G., & Mukhopadhyay, D. (2020). Detection of DDoS Attacks in Software Defined Network using Decision Tree. *Mendeley Data*, *1*. https://doi.org/10.17632/jxpfjc64kr.1

Ahuja, N., Singal, G., Mukhopadhyay, D., & Kumar, N. (2021). Automated DDOS attack detection in software defined networking. *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS*, *187*. https://doi.org/10.1016/j.jnca.2021.103108

Alamri, H. A., & Thayananthan, V. (2020). Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-Defined Networks Against DDoS Attacks. *IEEE ACCESS*, *8*, 194269-194288. https://doi.org/10.1109/ACCESS.2020.3033942

Alkasassbeh, M., Al-Naymat, G., Hassanat, A. B., & Almseidin, M. (2016). Detecting Distributed Denial of Service Attacks Using Data Mining Techniques. *International Journal of Advanced Computer Science and Applications*, *7*(1). https://doi.org/10.14569/IJACSA.2016.070159

Alojail, M., & Bhatia, S. (2020). A Novel Technique for Behavioral Analytics Using Ensemble Learning Algorithms in E-Commerce. *IEEE Access*, *8*, 150072-150080. https://doi.org/10.1109/ACCESS.2020.3016419

Alzahrani, A. O., & Alenazi, M. J. E. (2021). Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks. *FUTURE INTERNET*, *13*(5). https://doi.org/10.3390/fi13050111

An, J., & Kim, H.-W. (2018). A Data Analytics Approach to the Cybercrime Underground Economy. *IEEE Access*, *6*, 26636-26652. https://doi.org/10.1109/ACCESS.2018.2831667

Archanaa, R., Athulya, V., Rajasundari, T., & Kiran, M. V. K. (2017). A comparative performance analysis on network traffic classification using supervised learning algorithms. *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 1-5. https://doi.org/10.1109/ICACCS.2017.8014634

Argus. (2022). https://openargus.org/

Bannour, F., Souihi, S., & Mellouk, A. (2018). Distributed SDN Control: Survey, Taxonomy, and Challenges. *IEEE Communications Surveys & Tutorials*, *20*(1), 333-354. https://doi.org/10.1109/COMST.2017.2782482

Batch, A., & Elmqvist, N. (2018). The Interactive Visualization Gap in Initial Exploratory Data Analysis. *IEEE Transactions on Visualization and Computer Graphics*, *24*(1), 278-287. https://doi.org/10.1109/TVCG.2017.2743990

Benzekki, K., El Fergougui, A., & El Belrhiti El Alaoui, A. (2017). Software-defined networking (SDN): A survey. *Security and Communication Networks*, *9*. https://doi.org/10.1002/sec.1737

Bera, S., Misra, S., & Vasilakos, A. V. (2017). Software-Defined Networking for Internet of Things: A Survey. *IEEE Internet of Things Journal*, *4*(6), 1994-2008. https://doi.org/10.1109/JIOT.2017.2746186

Bosc, G., Tan, P., Boulicaut, J.-F., Raïssi, C., & Kaytoue, M. (2017). A Pattern Mining Approach to Study Strategy Balance in RTS Games. *IEEE Transactions on Computational Intelligence and AI in Games*, *9*(2), 123-132. https://doi.org/10.1109/TCIAIG.2015.2511819

Camacho, J., Therón, R., García-Giménez, J. M., Maciá-Fernández, G., & García-Teodoro, P. (2019). Group-Wise Principal Component Analysis for Exploratory Intrusion Detection. *IEEE Access*, *7*, 113081-113093. https://doi.org/10.1109/ACCESS.2019.2935154

da Silva Junior, J. R., Campagna, D. P., Clua, E., Sarma, A., & Murta, L. (2022). Dominoes: An Interactive Exploratory Data Analysis Tool for Software Relationships. *IEEE Transactions on Software Engineering*, *48*(2), 377-396. https://doi.org/10.1109/TSE.2020.2988241

Dehkordi, A. B., Soltanaghaei, M., & Boroujeni, F. Z. (2021). The DDoS attacks detection through machine learning and statistical methods in SDN. *JOURNAL OF SUPERCOMPUTING*, *77*(3), 2383-2415. https://doi.org/10.1007/s11227-020-03323-w

Elsayed, M. S., Le-Khac, N.-A., & Jurcut, A. D. (2020). InSDN: A Novel SDN Intrusion Dataset. *IEEE Access*, *8*, 165263-165284. https://doi.org/10.1109/ACCESS.2020.3022633

Evans, M., He, Y., Luo, C., Yevseyeva, I., Janicke, H., Zamani, E., & Maglaras, L. A. (2019). Real-Time Information Security Incident Management: A Case Study Using the IS-CHEC Technique. *IEEE Access*, *7*, 142147-142175. https://doi.org/10.1109/ACCESS.2019.2944615

Flask. (2023). https://flask.palletsprojects.com/en/2.2.x/

Garcia, C., Leite, D., & Škrjanc, I. (2020). Incremental Missing-Data Imputation for Evolving Fuzzy Granular Prediction. *IEEE Transactions on Fuzzy Systems*, *28*(10), 2348-2362. https://doi.org/10.1109/TFUZZ.2019.2935688

García, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, *45*, 100-123. https://doi.org/https://doi.org/10.1016/j.cose.2014.05.011

Gu, X., Wu, W., Gu, X., Ling, Z., Yang, M., & Song, A. (2020). Probe Request Based Device Identification Attack and Defense. *SENSORS*, *20*(16). https://doi.org/10.3390/s20164620

Habibi Lashkari, A. (2018). CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection. https://doi.org/10.13140/RG.2.2.13827.20003

Hong, K., Kim, Y., Choi, H., & Park, J. (2018). SDN-Assisted Slow HTTP DDoS Attack Defense Method. *IEEE Communications Letters*, *22*(4), 688-691. https://doi.org/10.1109/LCOMM.2017.2766636

Hubballi, N., & Swarnkar, M. (2018). $BitCoding$: Network Traffic Classification Through Encoded Bit Level Signatures. *IEEE/ACM Transactions on Networking*, *26*(5), 2334-2346. https://doi.org/10.1109/TNET.2018.2868816

Hussain, F., Abbas, S. G., Pires, I. M., Tanveer, S., Fayyaz, U. U., Garcia, N. M., Shah, G. A., & Shahzad, F. (2021). A Two-Fold Machine Learning Approach to Prevent and Detect IoT Botnet Attacks. *IEEE Access*, *9*, 163412-163430. https://doi.org/10.1109/ACCESS.2021.3131014

Janabi, A., Kanakis, T., & Johnson, M. (2022). Overhead Reduction Technique for Software-Defined Network Based Intrusion Detection Systems [Export Date: 15 November 2022]. *IEEE Access*, *10*, 66481-66491. https://doi.org/10.1109/ACCESS.2022.3184722

Kamel, H., & Abdullah, M. (2022). Distributed denial of service attacks detection for software defined networks based on evolutionary decision tree model. *Bulletin of Electrical Engineering and Informatics*, *11*(4), 2322-2330. https://doi.org/10.11591/eei.v11i4.3835

Kaspersky. (2022). https://securelist.com/ddos-report-q3-2022/107860/

Kitchenham, B., & Brereton, P. (2013). A systematic review of systematic review process research in software engineering. *Information and Software Technology*, *55*(12), 2049-2075. https://doi.org/https://doi.org/10.1016/j.infsof.2013.07.010

Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, *100*, 779-796. https://doi.org/https://doi.org/10.1016/j.future.2019.05.041

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, *103*(1), 14-76. https://doi.org/10.1109/JPROC.2014.2371999

Kumar, R., Swarnkar, M., Singal, G., & Kumar, N. (2022). IoT Network Traffic Classification Using Machine Learning Algorithms: An Experimental Analysis. *IEEE Internet of Things Journal*, *9*(2), 989-1008. https://doi.org/10.1109/JIOT.2021.3121517

Macas, M., Wu, C., & Fuertes, W. (2022). A survey on deep learning for cybersecurity: Progress, challenges, and opportunities. *Computer Networks*, *212*, 109032. https://doi.org/https://doi.org/10.1016/j.comnet.2022.109032

Mestre, X., & Vallet, P. (2017). Correlation Tests and Linear Spectral Statistics of the Sample Correlation Matrix. *IEEE Transactions on Information Theory*, *63*(7), 4585-4618. https://doi.org/10.1109/TIT.2017.2689780

Mininet. (2023). http://mininet.org/

Montenegro, C., Murillo, M., Gallegos, F., & Albuja, J. (2016). DSR Approach to Assessment and Reduction of Information Security Risk in TELCO. *IEEE Latin America Transactions*, *14*(5), 2402-2410. https://doi.org/10.1109/TLA.2016.7530438

Nadeem, M. W., Goh, H. G., Ponnusamy, V., & Aun, Y. (2022). DDoS Detection in SDN using Machine Learning Techniques. *CMC-COMPUTERS MATERIALS & CONTINUA*, *71*(1), 771-789. https://doi.org/10.32604/cmc.2022.021669

Nunez-Agurto, D., Fuertes, W., Marrone, L., & Macas, M. (2022). Machine Learning-Based Traffic Classification in Software-Defined Networking: A Systematic Literature Review, Challenges, and Future Research Directions. *IAENG International Journal of Computer Science*, *49*(4).

Pérez-Díaz, J. A., Valdovinos, I. A., Choo, K.-K. R., & Zhu, D. (2020). A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning. *IEEE Access*, *8*, 155859-155872. https://doi.org/10.1109/ACCESS.2020.3019330

Rahouti, M., Xiong, K., Xin, Y., Jagatheesaperumal, S. K., Ayyash, M., & Shaheed, M. (2022). SDN Security Review: Threat Taxonomy, Implications, and Open Challenges. *IEEE Access*, *10*, 45820-45854. https://doi.org/10.1109/ACCESS.2022.3168972

Rios, V. D. M., Inácio, P. R. M., Magoni, D., & Freire, M. M. (2022). Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey. *IEEE Access*, *10*, 76648-76668. https://doi.org/10.1109/ACCESS.2022.3191430

Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., & Burgos, D. (2020). An Evolutionary SVM Model for DDOS Attack Detection in Software Defined Networks. *IEEE ACCESS*, *8*, 132502-132513. https://doi.org/10.1109/ACCESS.2020.3009733

Sangodoyin, A. O., Akinsolu, M. O., Pillai, P., & Grout, V. (2021). Detection and Classification of DDoS Flooding Attacks on Software-Defined Networks: A Case Study for the Application of Machine Learning. *IEEE Access*, *9*, 122495-122508. https://doi.org/10.1109/ACCESS.2021.3109490

Scott-Hayward, S., Natarajan, S., & Sezer, S. (2016). A Survey of Security in Software Defined Networks. *IEEE Communications Surveys and Tutorials*, *18*(1), 623-654. https://doi.org/10.1109/COMST.2015.2453114

Shahzadi, S., Ahmad, F., Basharat, A., Alruwaili, M., Alanazi, S., Humayun, M., Rizwan, M., & Naseem, S. (2021). Machine Learning Empowered Security Management and Quality of Service Provision in SDN-NFV Environment. *CMC-COMPUTERS MATERIALS & CONTINUA*, *66*(3), 2723-2749. https://doi.org/10.32604/cmc.2021.014594

Sudar, K. M., & Deepalakshmi, P. (2021). An intelligent flow-based and signature-based IDS for SDNs using ensemble feature selection and a multi-layer machine learning-based classifier. *JOURNAL OF INTELLIGENT & FUZZY SYSTEMS*, *40*(3), 4237-4256. https://doi.org/10.3233/JIFS-200850

Sufiev, H., Haddad, Y., Barenboim, L., & Soler, J. (2019). Dynamic SDN Controller Load Balancing. *FUTURE INTERNET*, *11*(3). https://doi.org/10.3390/fi11030075

Sumadi, F., Widagdo, A., Reza, A., & Syaifuddin. (2022). SD-Honeypot Integration for Mitigating DDoS Attack Using Machine Learning Approaches [Export Date: 15 November 2022]. *International Journal on Informatics Visualization*, *6*(1), 39-44. https://doi.org/10.30630/joiv.6.1.853

Tan, L., Pan, Y., Wu, J., Zhou, J., Jiang, H., & Deng, Y. (2020). A New Framework for DDoS Attack Detection and Defense in SDN Environment. *IEEE Access*, *8*, 161908-161919. https://doi.org/10.1109/ACCESS.2020.3021435

Tariq, F., & Baig, S. (2017). Machine Learning Based Botnet Detection in Software Defined Networks. *INTERNATIONAL JOURNAL OF SECURITY AND ITS APPLICATIONS*, *11*(11), 1-11. https://doi.org/10.14257/ijsia.2017.11.11.01

Tonkal, O., Polat, H., Başaran, E., Comert, Z., & Kocaoglu, R. (2021). Machine learning approach equipped with neighbourhood component analysis for ddos attack