UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE GRADO PARA OPTAR EL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

TÍTULO:
DESARROLLO DE UN MODELO CLASIFICADOR DE MALWARE CON ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

Autor:
JAIME ANDRES PORTILLA JAIMES

Director:
ESP. JOHRMAN DE JESÚS VIDES NIÑO

PAMPLONA-COLOMBIA

JUNIO DE 2021

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE GRADO PARA OPTAR EL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

TÍTULO:
DESARROLLO DE UN MODELO CLASIFICADOR DE MALWARE CON ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

Autor:
JAIME ANDRES PORTILLA JAIMES

Director:
ESP. JOHRMAN DE JESÚS VIDES NIÑO

JURADO CALIFICADOR:
ESP. JOHRMAN DE JESÚS VIDES NIÑO
M.Sc. LUIS ENRIQUE MENDOZA
ING. GUSTAVO QUIJADA MACUART

PAMPLONA-COLOMBIA

JUNIO DE 2021

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
DEPARTAMENTO DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA, SISTEMAS Y TELECOMUNICACIONES

PROGRAMA DE INGENIERÍA EN TELECOMUNICACIONES

TRABAJO PRESENTADO PARA OPTAR POR ÉL TITULO DE INGENIERO EN TELECOMUNICACIONES

TEMA:

DESARROLLO DE UN MODELO CLASIFICADOR DE MALWARE CON ALGORITMOS DE APRENDIZAJE AUTOMÁTICO

FECHA DE INICIO DEL TRABAJO: MAYO 2019

FECHA DE TERMINACIÓN DEL TRABAJO: JULIO 2021

NOMBRES Y FIRMAS DE AUTORIZACIÓN PARA LA SUSTENTACIÓN:

_____
JAIME ANDRÉS PORTILLA JAIMES
AUTOR

_____ _____
ESP. JOHRMAN VIDES NIÑO **M.Sc. HERNANDO VELANDIA V.**
DIRECTOR **DIRECTOR DEL PROGRAMA**

JURADO CALIFICADOR:

_____ _____
ESP. JOHRMAN VIDES NIÑO **M.Sc. LUIS ENRIQUE MENDOZA**

_____
ING. GUSTAVO QUIJADA MACUART

PAMPLONA N. S. COLOMBIA
JUNIO DE 2021

DEDICATORIA

*Dedicado a mi familia Portilla Jaimes, a mis hermanos profesionales, para ustedes **Tanton** y **Titi**,*
*mis padres **Jaime** y **Carmen**.*

*Para ti **Sharon Viviana**, y **Âtum**.*

Mi Novia y mi hijo Gatuno.

AGRADECIMIENTOS

Agradecimiento a nuestra alma mater, Universidad de Pamplona, por el proceso enseñanza y formación en el área de Ingeniería en Telecomunicación, en su línea de profundización en telemática. Al tutor y cuerpo de profesores, muchas gracias por su transmisión del conocimiento.

Un agradecimiento especial a PhD. Arash Habibi Lashkari, Assistant Professor del Canadian Institute for Cybersecurity, la Faculty of Computer Science, University of New Brunswick. Gracias por su trabajo investigativo, con filosofía open source (OSS) y por su atención en un mensaje respondido, siendo muy amable y percibir su calidad de ser humano, al instante.

RESUMEN

Las tecnologías, procesos, metodologías, herramientas y tácticas que contempla la defensa en profundidad (DiD) y específicamente en sus dos capas de seguridad de las redes y seguridad perimetral, requieren habilidades para controlar oportunamente periódicas amenazas del cibercrimen en un panorama cada vez más amplio ante vulnerabilidades y ataques. Solo considerando los ataques de denegación de servicios DoS y distribuidos DDoS, que se caracterizan en este proyecto, resulta al final con un conjunto de datos de 10 familias o clases diferentes de ataques DoS y DDoS, que junto a un tráfico benigno se busca detectar y clasificar en once 11 multiclases mediante el uso de técnicas basadas en aprendizaje automático supervisado (*Machine Learning*). El conjunto de datos etiquetado, es una colección de los años 2017, 2018 y 2019, que contiene los atributos del tráfico de red capturado y es usado con CICFlowMeter proveído por investigadores de Canadian Institute for Cybersecurity de University of New Brunswick CIC – UNB, y cuyos datos están aptos para ser la data de entrada de los algoritmos y técnicas de aprendizaje automático a implementar en este proyecto. Se diseñó un modelo detector y clasificador multiclase gracias al aprendizaje de 20 algoritmos diferentes, entrenados y aplicando validación cruzada para conocer su desempeño y comportamiento, descartando la mayoría, y estableciéndonos en solo 5 algoritmos: Bosques Aleatorios RF, Árbol Extremo ET, Árbol de Decisión DT y 2 formas diferentes de aplicar el novedoso **XGBoost**. Así, se evalúan mediante métricas y resultados de rendimiento, data que no hace parte del conjunto de entrenamiento, no ha sido vista, y que cuenta con etiquetas reales dadas, para obtener en el mejor de ellos los 5.

Los resultados obtenidos del mejor de los 5 algoritmos de aprendizaje, corresponde al de **Bosques Aleatorios**. Así se ha logrado obtener en nuestro diseño de modelo final un rendimiento, muy bueno juzgando el lector mismo. Primero con el conjunto de datos de prueba normal 10% del conjunto total, para un total de **1’359.628** flujos de tráfico de red: accuracy[f1-socre] de **1.00**; macro avg[precision] **0,98**; macro avg[recall] **0,99**; macro avg[f1-socre] **0,98**; weighted avg[precision] **1,00**; weighted avg[recall] **1,00**; weighted avg[f1-socre] **1,00**. Con una Perdida logistica lloss_RF de **0.0656859**. Perdida de Hamming de **0.0045166**.

Palabras claves: Defense in depth; Network security; Perimeter security; Intrusion detection; DoS and DDoS attack detection; network traffic analysis; Machine Learning algorithms: Multiclass Classification.

ABSTRACT

The technologies, processes, methodologies, tools and tactics contemplated by defense in depth (DiD) and specifically in its two layers of network security and perimeter security, require skills to timely control periodic cybercrime threats in an increasingly broad landscape of vulnerabilities and attacks. Just considering the DoS and DDoS distributed denial of service attacks, which are characterized in this project, results in the end with a dataset of 10 different families or classes of DoS and DDoS attacks, which together with benign traffic is sought to detect and classify into eleven 11 multiclasses by using techniques based on supervised machine learning (Machine Learning). The labeled dataset, is a collection of the years 2017, 2018 and 2019, which contains the attributes of the captured network traffic and is used with CICFlowMeter provided by researchers from Canadian Institute for Cybersecurity of University of New Brunswick CIC - UNB, and whose data is suitable to be the input data for the algorithms and machine learning techniques to be implemented in this project.

A multiclass detector and classifier model was designed thanks to the learning of 20 different algorithms, trained and applying cross validation to know their performance and behavior, discarding most of them, and settling on only five algorithms: Random Forests RF, Extreme Tree ET, Decision Tree DT and 2 different ways of applying the novel XGBoost. Thus, data that is not part of the training set, has not been seen, and has real labels given, are evaluated by means of metrics and performance results, to obtain the best of the five.

The results obtained from the best of the five learning algorithms, corresponds to the Random Forests. Thus, we have managed to obtain in our final model design a performance, very good judging the reader himself. First with the normal test data set 10% of the total set, for a total of 1'359,628 network traffic flows: accuracy[f1-socre] of 1.00; macro avg[precision] 0.98; macro avg[recall] 0.99; macro avg[f1-socre] 0.98; weighted avg[precision] 1.00; weighted avg[recall] 1.00; weighted avg[f1-socre] 1.00. With a logistic loss lloss_RF of 0.0656859. Hamming loss of 0.0045166.

Keywords: Defense in depth; Network security; Perimeter security; Intrusion detection; DoS and DDoS attack detection; network traffic analysis; Machine Learning algorithms: Multiclass Classification.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO

Clasificador - Es un algoritmo que se utiliza para asignar los datos de entrada a una categoría específica.

Modelo de clasificación - El modelo predice o saca una conclusión de los datos de entrada dados para el entrenamiento, predecirá la clase o categoría para los datos.

Característica - Una característica es una propiedad individual medible del fenómeno observado.

Clasificación binaria: es un tipo de clasificación con dos resultados, por ejemplo, verdadero o falso.

Clasificación multiclase - La clasificación con más de dos clases, en la clasificación multiclase cada muestra se asigna a una y sólo una etiqueta u objetivo.

Clasificación multietiqueta - Es un tipo de clasificación en el que cada muestra se asigna a un conjunto de etiquetas u objetivos.

Entrenar el clasificador - Cada clasificador en sci-kit learn utiliza el método fit(X, y) para ajustar el modelo para entrenar el tren X y entrenar la etiqueta y.

Predecir el objetivo - Para una observación no etiquetada X, el método predict(X) devuelve la etiqueta predicha y.

Evaluar - Esto significa básicamente la evaluación del modelo, es decir, el informe de clasificación, la puntuación de precisión, etc.

Capítulo 1

INTRODUCCIÓN

El titánico avance y desarrollo de las TIC Tecnologías de la Información y la Comunicación, ha desencadenado la revolución digital, cambiando por completo el comportamiento y diario vivir de todas las personas, así como modificado y transformado sus relaciones sociales; además con la requerimiento intrínseco, que dichas tecnologías actuales demandan de un panorama mejorado de mecanismos de seguridad avanzados que garanticen la confidencialidad, integridad, disponibilidad y autenticación de la información, y el aumento de prácticas de detección e identificación de nuevas amenazas y ataques de día cero (*0-day attack*). El crecimiento del perjuicio y daño a los sistemas informáticos, con el actuar de conductas antisociales e ilícitas que han pasado y se manifiestan al plano digital, ahora surgen formas de cometer delitos de tipo tradicional, en formas no tradicionales.

1.1 Planteamiento del Problema

Existe una preocupación global en la infinidad de formas en que se presenta y adopta un *Malware*: mediante códigos, herramientas o software y en todas congenia que se realiza de forma maliciosa, malintencionada, ladina e ilegal causando daño

y perjuicio en dispositivos finales, redes y datos(información); y en consecuencia se identifica un cambio generacional, en que la mayor parte del *malware* era creado por adolescentes bromistas, en la actualidad, el *malware* es diseñado en gran medida por y para criminales profesionales, y es definido a qué nos enfrentamos a una: empresa criminal [1].

En nuestra era es innegable el crecimiento exponencial del uso de redes informáticas en organizaciones y en el hogar, lo que conlleva grandes, constantes y renovables retos en el ámbito de la seguridad de la tecnología de la información. Precisamente, se contempla que las técnicas de detección de intrusiones comunes están lejos de ser perfectas cuando se comparan versus una variedad de técnicas y herramientas modernas utilizadas por los atacantes e intrusos. En otras palabras, ahora los atacantes rivalizan y compite a veces de forma superior frente a la defensa, protección y escudo de seguridad de nuestras redes informáticas.

Los sistemas de seguridad tradicionales como cortafuegos, los sistemas de detección de intrusiones (IDS - *Intrusion Detection System*) y demás mecanismos de seguridad de red y seguridad perimetral, aún no ofrecen un método de defensa eficaz contra los ataques de denegación de servicio distribuidos (DDoS - *Distributed Denial of Service*) tan nuevos como los tipos reflexivos, o de tipo explotación no pueden defenderse de los complejos ataques DDoS [2], [3], [4], [5] ya que la mayoría de ellos filtran el tráfico normal y el sospechoso basándose en reglas y firmas estáticas predefinidas [6]. Dichos ataques se han convertido en un problema que debe tener importancia de grupos académicos de investigación, pues los incidentes de daños graves debidos a ataques DDoS han ido en aumento, incrementando tanto su intensidad, como su complejidad; lo que ha llevado a la necesidad urgente de nuevos mecanismos, en primera medida y a nuestro alcance propuesto del proyecto, de identificación y clasificación, de forma pasiva, de ataques intrusivos de denegación de servicio. Para detectar los ataques DoS y DDoS, es necesario

analizar dinámicamente las características básicas de los ataques, ya que sus patrones, puertos y protocolos o mecanismos de funcionamiento cambian y se manipulan rápidamente [7].

La mayoría de los métodos de defensa DDoS propuestos por la literatura tienen diferentes tipos de inconvenientes, debilidades y limitaciones. Como ya hemos mencionado, la mayoría de mecanismos de detección son basados en firmas y reglas que no logran identificar nuevos ataques o ataques de día cero, o en su caso variantes que actúan totalmente diferentes a sus antecesores; y otros tienen mecanismos de defensa basados en anomalías clásicos que se limitan a tipos específicos de ataques DDoS y que aún no se pueden aplicar en entornos abiertos. Además, abordando técnicamente este problema, la debilidad de los IDS comunes se debe principalmente al déficit del protocolo TCP/IP en sí mismo, como vulnerabilidad, que facilita a un atacante iniciar un ataque DDoS, por ejemplo, utilizando el protocolo ICMP con la utilidad ping, que está disponible por defecto en todos los sistemas operativos, o utilizando herramientas especiales como **HOIC**, **LOIC**, **XOIC**, **golden-eye**, entre muchos otros, a veces hasta desconocidas [8]. El uso de protocolos estándar TCP/IP por parte de los atacantes para llevar a cabo ataques DDoS hace que el objetivo sea demasiado lento para darse cuenta de que está siendo atacado(detección), por lo que también impacta en el proceso de mitigación del ataque [9].

Ante los problemas y debilidades presentadas se hace necesario desarrollar un modelo detector de ataques DoS y DDoS utilizando técnicas de aprendizaje automático, de modo que pueda suponer una solución mejorada de los métodos de defensa contra los ataques DDoS. En conclusión, el trabajo de grado busca dar respuesta a la siguiente pregunta de investigación:

¿De qué forma se podría detectar los ataques intrusivos de Denegación DoS y DDoS analizando flujos de paquetes de tráfico de red?, ¿son los métodos de

detección de patrones con técnicas de Aprendizaje Automático un camino de solución alternativo y complementario, a la detección de intrusiones DoS y DDoS tradicionales fundamentada en bases de datos basado en enfoques de firmas y reglas?

1.2 Justificación

De los muchos tipos de actividad delictiva que ocurren en la red, pocos son más desconcertantes y difíciles de mitigar que los ataques distribuidos de denegación de servicio (DDoS). Tanto que es denominado por Norton como "una de las armas más poderosas de Internet" [10].

Los ataques de denegación de servicio pueden producirse en cualquier momento, afectar cualquier parte de las operaciones o los recursos de un sitio web, y provocar una gran cantidad de interrupciones del servicio y enormes pérdidas económicas. En cifras estadísticas actuales, primero a nivel mundial, se evidencia en el informe tendencias de ataques DDoS 2020 de F5:

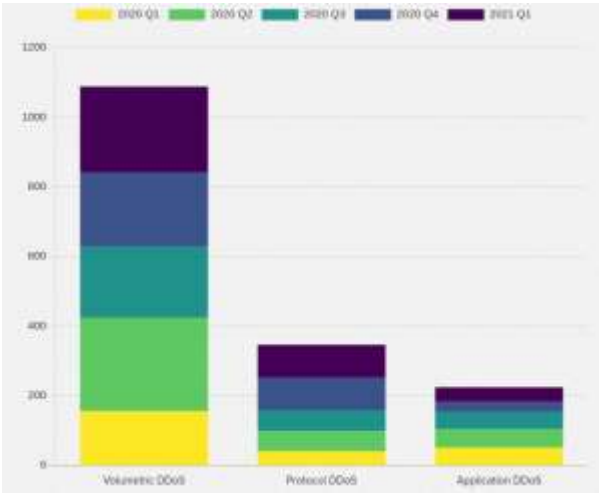


Figura 1-1 Frecuencia tipos de ataques DDoS, ene/20 a mar/21 [11].

Entre enero de 2020 y marzo de 2021, los ataques DDoS aumentaron en un 55% y se están volviendo más complejos, y el 54% de los incidentes utilizan

múltiples vectores de ataque. El ataque más grande de los últimos 15 meses midió 500 Gbps y utilizó no menos de cinco vectores de ataque diferentes. El sector de la tecnología fue el más atacado, recibiendo el 27% de todos los ataques DDoS en los últimos 15 meses. Los DDoS volumétricos representaron el 73% de todos los incidentes. Específicamente, el 53% de los ataques se aprovecharon de algún tipo de ataque de reflexión, que aprovecha los sistemas vulnerables de otras personas. [11]

Además de grafico representativo del primer trimestre de 2021 [12]. En otro informe mundial, tenemos el extenso reporte de internet anual de Cisco presentado en marzo de 2020 donde se expone:

Un crecimiento del 776% en ataques entre 100 Gbps y 400 Gbps de 2018 a 2019, y el número total de ataques DDoS se duplicará de 7,9 millones en 2018 a 15,4 millones en 2023 [12]

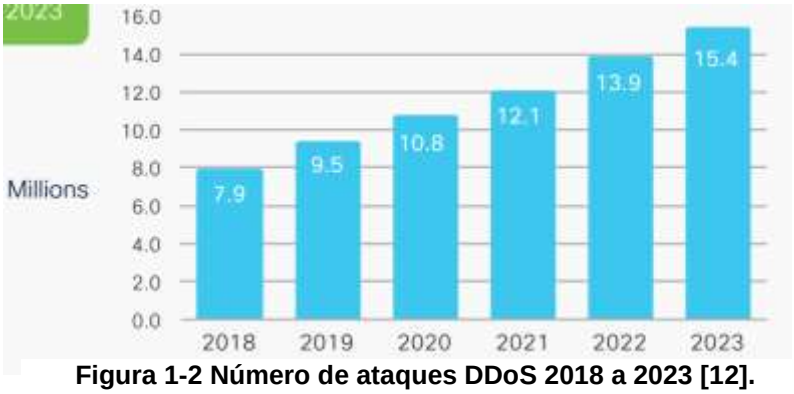


Figura 1-2 Número de ataques DDoS 2018 a 2023 [12].

De acuerdo a las cifras puntuales en la región latinoamericana, Level 3 informa, proveedor global de servicios de red, sobre amenazas que reveló que un 12 por ciento de los ataques de Denegación Distribuida de Servicio están dirigidos a Latinoamérica, y que dicho número está creciendo [13]. En el caso específico de Colombia, a través del CSOC Centro de Operaciones de Ciberseguridad de Claro, se gestionaron hasta noviembre de 2020, en promedio, más de 4 millones de eventos de seguridad al mes; desde el inicio de la pandemia, se presentó un claro aumentó de 88% en la cantidad de incidentes de seguridad, la mayor porción está relacionados con la afectación de la disponibilidad de los sistemas y/o servicios, a

través de ataques de denegación de servicio con aumento del 80% [14]. Complementariamente, según cifras del Centro Cibernético Policial, 170 empresas reportaron ataques DDoS que consiguieron interrumpir sus servicios de cara a sus clientes [15].

Figura 1-3 Crecimiento de incidentes seguridad en Colombia 2020 [15].

Conociendo el contexto del ecosistema de ataques de denegación de servicio critico a asegurar a nivel global en general, y conociendo que los ataques DDoS pueden implementarse en las capas de red, transporte y aplicación utilizando diferentes protocolos, como TCP, UDP, ICMP y HTTP se hizo importante y pertinente llevar a cabo el desarrollo de un modelo detector, que nos permite identificar y clasificar ataques maliciosos de denegación de servicios. Entregando y aportando a nivel investigativo a un enfoque de detección y clasificación de familias DoS y DDoS basado en el entrenamiento en un conjunto de atributos de flujo de tráfico de red, de donde dicho aprendizaje autónomo de máquina, sirve para predecir sobre un tráfico nuevo, que será capturado, transformado al formato requerido para entrada de nuestro modelo, y en esa toma de decisiones con pesos probabilísticos de resultado, se podrá monitorizar el tráfico de red en tiempo real y responder con una alarma, cuando detecte intrusiones maliciosas de tipo de denegación. Esto permitiría tener un modelo detector flexible, rápido en tiempo de predicción y que con un ordenador mucho más potente que el utilizado en el proyecto, y con mayor capacidad de Memoria RAM poder aprender con un mejorado y renovado conjunto de entrenamiento, lo que aumentara la eficiencia de identificación de ataques intrusivos de denegación de servicio, así como los demás

ataques componen las intrusiones, porque aprende de forma inteligente, según sea la capacidad de la maquina donde se realice el coste computacional de la fase de entrenamiento.

1.3 Delimitación

1.3.1. Objetivo General

Desarrollar un modelo clasificador de Malware aplicando algoritmos de aprendizaje automático.

1.3.2. Objetivos Específicos

- Definir las características de un malware en específico a detectar, seleccionado su año de creación, entre más actual puede saltarse la protección mediante firmas y su efecto dañino y escalable en la red.
- Analizar los requisitos funcionales del modelo clasificador preprocesando los datos de trafico de red, extrayendo y seleccionado características que indiquen parámetros anómalos.
- Diseñar el modelo aplicando una técnica de algoritmo de clasificación supervisada o One-Class, seleccionando el que mejor medidas de evaluación de desempeño presente.
- Implementar el modelo clasificador con uso de algoritmos de aprendizaje automático en una red perimetral que permita la identificación del tráfico de un malware en específico.
- Evaluar el modelo clasificador con herramientas que permitan la visualización del desempeño del algoritmo de aprendizaje automático seleccionado.

1.3.3. Acotaciones

Las herramientas de software a utilizar en el proyecto son todos de código abierto, para no lidiar con precios comerciales de licencias oficiales:

- Sistema Operativo Ubuntu 20.04.2 LTS, 64 bits, v.GNOME 3.36.8.
- Entorno de desarrollo integrado IDE Sypder 5.0.0 de la suite Anaconda de Lenguaje de programación Python 3.

- Librerías y módulos de múltiples fuentes: Scikit-learn, NumPy, Pandas, SciPy, Matploit, Seasorbn, XGBoost.
- 3 conjuntos de datos anuales (2017-2019) todos del CIC de la UNB.
- Herramienta CICFlowMeter v4.0 de CIC.

Dentro de la delimitación y el alcance de nuestro proyecto no contempla la interfaz gráfica para un usuario final pues su adopción a aplicaciones del mundo real se ha visto obstaculizada debido a la complejidad del sistema, ya que estos sistemas requieren una cantidad sustancial de pruebas, evaluación y ajuste antes de la implementación [CIC2018]. Se realizarán pruebas en tiempo real, tanto de trafico esnifado o capturado en directo, como de archivos de paquetes capturados (.pcap) previamente almacenados encontrados en una búsqueda muy exhaustiva en las fuentes electrónicas. Además, se usuaria la Herramienta de CICFlowMeter del Instituto Canadiense de Ciberseguridad CIC.

Se debe indicar que se van a detectar las técnicas que usan las herramientas de malware, en este caso a las herramientas maliciosas (*tools/software*) que explotan la denegación de servicios DoS y DDoS como lo son estos open-source softwares: **LOIC** (Low Orbit ION cannon) por Praetox Technologies en C#; **HOIC** (High Orbit Ion Cannon) por grupo Anonymous; **HULK** (HTTP Unbearable Load King) por Barry Shteiman en Python; **GoldenEye** (HTTP Keep Alive + NoCache) por Jan Seidl en Python; **Slowloris** (HTTP Low bandwidth) por Robert Hansen/Gokberk Yaltirakli en Python; **SlowHTTPTest** por Sergey Shekyan. Además, tampoco se tiene intención aplicarlas o implementar estos softwares, no se ha propuesto en nuestro enfoque o alcance.

Por último, indicar que se hará uso de un conjunto de datos validados en la investigación a nivel internacional del Instituto de Ciberseguridad de Canadá de la Universidad New Brunswick, lo mejor de dicha elección de datos es el ser realista, actualizada, completa y de fuente abierta, solo se deben citar en el caso cuatro

papers de investigación, uno por cada año usado, 2016 a 2019: 3 para entrenar y doble evaluación, y el de 2016 para realizar un experimento como paquete capturado (pcap) almacenado.

Capítulo 2

MARCO TEÓRICO

2.1 Ataques informáticos

En informática y redes informáticas, un ataque es cualquier intento de exponer, alterar, inutilizar, destruir, robar u obtener información a través de un acceso no autorizado o hacer un uso no autorizado de un activo [16]. Un atacante es una persona o proceso que intenta acceder a datos, funciones u otras áreas restringidas del sistema sin autorización, potencialmente con intención maliciosa [17]. Los ciberataques se han vuelto cada vez más sofisticados y peligrosos [18].

2.1.1. Taxonomía de los ataques informáticos

La clasificación, ordenación y tipos de ataques informáticos se presenta en varios ámbitos y esferas. Como el ataque puede ser activo o pasivo [19].

- Un **ataque activo** intenta alterar los recursos del sistema o afectar a su funcionamiento.
- Un **ataque pasivo** intenta aprender o hacer uso de la información del sistema, pero no afecta a los recursos del sistema.

Un ataque puede ser perpetrado por una persona interna o desde fuera de la organización [19].

- Un ataque interno es un ataque iniciado por una entidad dentro del perímetro de seguridad (un "*insider*"), es decir, una entidad que está autorizada a acceder a los recursos del sistema pero que los utiliza de una manera no aprobada por los que concedieron la autorización.
- Un **ataque externo** se inicia desde fuera del perímetro, por un usuario no autorizado o ilegítimo del sistema (un "outsider"). En Internet, los posibles atacantes externos van desde bromistas aficionados hasta delincuentes organizados, terroristas internacionales y gobiernos hostiles.

2.2 Seguridad de la Red

La seguridad de la red es un término que describe las herramientas, tácticas y políticas de seguridad diseñadas para monitorear, prevenir y responder a intrusiones no autorizadas en la red, al mismo tiempo que se protegen los activos digitales, incluido el tráfico de la red. La seguridad de la red incluye tecnologías de hardware y software (incluidos recursos como analistas de seguridad expertos, cazadores, respondedores de incidentes, etc.) y está diseñada para responder a la gama completa de amenazas potenciales dirigidas a su red [20].

Stephen Northcutt, escribió un manual sobre los conceptos básicos de la seguridad de la red para CSOonline hace más de una década, pero creemos firmemente que su visión de las tres fases de la seguridad de la red sigue siendo

relevante y debería ser el marco subyacente de su estrategia. En su relato, la seguridad de la red consiste en [21]:

- **Protección**: debe configurar sus sistemas y redes de la manera más correcta posible
- **Detección:** debe poder identificar cuándo ha cambiado la configuración o cuándo algo de tráfico de red indica un problema
- **Reacción:** después de identificar los problemas rápidamente, debe responder a ellos y regresar a un estado seguro lo más rápido posible.

Estas pueden considerarse estrategias de la defensa en profundidad. Si hay un tema común entre los expertos en seguridad, es que confiar en una sola línea de defensa es peligroso, porque cualquier herramienta defensiva puede ser derrotada por un adversario determinado [21].

2.3 Seguridad Perimetral

La seguridad perimetral informática son todos los sistemas destinados a proteger de intrusos un perímetro que detecta amenazas, realiza vigilancia y analiza patrones de ataque, como tal, a menudo sirve como la primera línea de defensa de una red contra muchos peligros que pueden dañar los sistemas [22].

La seguridad perimetral protege que las redes deben cumplir cuatro funciones básicas [23]:

- Resistir a los ataques externos.
- Identificar los ataques sufridos y alertar de ellos.
- Aislar y segmentar los distintos servicios y sistemas en función de su exposición a ataques.

- Filtrar y bloquear el tráfico, permitiendo únicamente aquel que sea absolutamente necesario.

2.3.1. Sistemas anti-DDos

Estos sistemas previenen o mitigan los ataques de denegación de servicio (DoS) y los ataques distribuidos de denegación de servicio (DDoS). Generalmente este tipo de sistemas necesitan un tiempo de aprendizaje para modelar cuál es el comportamiento normal o las tendencias en el tráfico de la red, estableciendo unas líneas base de los distintos volúmenes de tipos de tráfico, para que una vez se pongan en modo bloqueo, cuando se produzca un ataque y se detecten desviaciones de las líneas base, sean capaces de bloquear o mitigar dichos ataques evitando que el tráfico anómalo ingrese a la red [24].

2.4 Metodologías de Detección

Las metodologías de detección de intrusos se clasifican en tres grandes categorías [25]:

2.4.1. Detección basada en Firmas (SD - *Signature-based Detection)*

Una firma es un patrón o cadena que corresponde a un ataque o amenaza conocidos. SD es el proceso de comparar patrones con eventos capturados para reconocer posibles intrusiones. Debido a la utilización de los conocimientos acumulados por ataques específicos y vulnerabilidades del sistema, la SD también se conoce como detección basada en el conocimiento o detección de uso indebido (*Knowledge-based Detection* or *Misuse-based Detection*). Compara la actividad del usuario con patrones de intrusión conocidos, llamados firmas [26]. El sistema consta de una base de datos interna de firmas. Si la actividad de un usuario coincide con las firmas almacenados, se activará una alarma.

Ventajas:

- Método preciso y eficiente para la detección de ataques conocidos.

- Protege la red/sistema inmediatamente después de la instalación.
- El mecanismo es fácil de entender.
- Alta velocidad de detección, ya que dedica menos tiempo a tratar los falsos positivos.

Desventajas:

- Ineficaz para detectar ataques desconocidos y variaciones de ataques conocidos.
- Difícil de actualizar regularmente las firmas [27].

2.4.2. Detección basada en Anomalías (AD - *Anomaly-based Detection*)

Una anomalía es una desviación de un comportamiento conocido, y los perfiles representan los comportamientos normales o esperados derivados de la supervisión de las actividades regulares, las conexiones de red, los hosts o los usuarios durante un período de tiempo. Los perfiles pueden ser estáticos o dinámicos, y se desarrollan para muchos atributos, por ejemplo, los intentos fallidos de inicio de sesión, el uso del procesador, el recuento de correos electrónicos enviados, etc. A continuación, AD compara los perfiles normales con los eventos observados para reconocer los ataques significativos. En algunos artículos, AD también se denomina detección basada en el comportamiento (*Behavior-based Detection*). Algunos ejemplos de AD son: intento de intrusión, enmascaramiento, penetración por parte de un usuario legítimo, DoS, troyanos, etc.

Compara las actividades de un sistema en un instante con el comportamiento normal y genera la alarma siempre que una desviación del comportamiento normal supera el umbral predefinido. Consta de dos pasos: entrenamiento y detección. Debe ser entrenado a partir del comportamiento normal antes de poder ser utilizado

en un modelo de detección [29]. Durante el modo de detección, clasifica el comportamiento anormal del comportamiento normal sobre la base de técnicas heurísticas o reglas.

Ventajas:

- Eficiente para detectar ataques nuevos e imprevistos.
- Facilita la detección de abusos de privilegios en los recursos.
- Se requiere muy poco mantenimiento después de su instalación, ya que aprende continuamente de las actividades de la red y construye los respectivos perfiles.
- Menos dependiente del software del sistema.

Desventajas:

- Puede hacer una clasificación errónea en la detección debido a los datos de intrusión en la fase de entrenamiento.
- Precisión débil de los perfiles debido a que los eventos observados cambian constantemente.
- Falta de escala a velocidades de gigabit.
- Aceptar el comportamiento del ataque como normal cuando el atacante cambia sus patrones de comportamiento.
- Difícil activar las alertas en tiempo real.
- La deficiencia de las muestras anormales en la fase de entrenamiento desafía la definición del comportamiento normal [27].

2.4.3. Análisis de protocolos con Estado (SPA- *Stateful Protocol Analysis*)

El estado en SPA indica que los IDS pueden conocer y rastrear los estados del protocolo (por ejemplo, emparejar las solicitudes con las respuestas). Aunque el proceso SPA se parece a los AD, son esencialmente diferentes. AD adopta perfiles

específicos de red o de host precargados, mientras que SPA depende de perfiles genéricos desarrollados por el proveedor para protocolos específicos. Por lo general, los modelos de protocolo de red en SPA se basan originalmente en los estándares de protocolo de las organizaciones internacionales de normalización, por ejemplo, el IETF. SPA también se conoce como detección basada en especificaciones (*Specification-based Detection*).

Define un conjunto de restricciones que describen el funcionamiento correcto de un protocolo o programa y supervisa el protocolo en un instante con las restricciones definidas para identificar desviaciones [28].

Ventajas:

- Facilidad para añadir las características de especificación al analizador de protocolo.
- Identifica con eficacia las secuencias de acciones inesperadas.
- Capacidad para detectar ataques desconocidos con bajas tasas de falsos positivos.

Desventajas:

- Agotamiento de los recursos debido al seguimiento continuo del estado del protocolo.
- Imposibilidad de detectar aquellos ataques que no violan directamente el comportamiento del protocolo.
- El desarrollo de las características de la especificación es un proceso tedioso.
- Puede ser incompatible con versiones particulares de ciertas aplicaciones y software del sistema [27].

2.5 Enfoques de Detección

Tradicionalmente, se estudian los enfoques de detección de intrusiones desde dos puntos de vista principales, la detección de anomalías y la detección de usos indebidos, pero no hay una diferencia considerable en sus características. Stavroulakis y Stamp (2010) propusieron una clasificación para subdividir estos enfoques en tres subcategorías que incluyen el enfoque dependiente de la computación, la inteligencia artificial y los conceptos biológicos. Sin embargo, una clasificación de este tipo es demasiado difícil para ver todas las propiedades de los enfoques de detección. A falta de una visión más detallada de los enfoques de detección, presentamos una clasificación de cinco subclases con una perspectiva en profundidad de sus características: Basados en **estadísticas**, basados en **patrones**, basados en **reglas**, basados en **estados** y basados en **heurística**.

Los enfoques basados en estadísticas se basan principalmente en umbrales predefinidos, media y desviación estándar, y probabilidades para identificar las intrusiones. La detección basada en patrones se centra en los ataques conocidos mediante la coincidencia de cadenas. Además, en las técnicas basadas en reglas se aplican reglas *If-Then* o *If-Then-Else* para construir el modelo y el perfil de las intrusiones conocidas. En especial, los métodos basados en estados explotan las máquinas de estados finitos derivadas de los comportamientos de la red para identificar los ataques.

El último es el enfoque basado en la heurística, que se inspira en conceptos biológicos y en la inteligencia artificial. Trabajos más recientes (Fragkiadakis et al., 2012; Mar et al., 2012; Kartit et al., 2012; Farooqi et al.,2012; Modiet al., 2012; Wang et al., 2011; Couture, 2012; Li et al., 2012) integran varios enfoques de detección de cinco subclases en uno sofisticado para dar una mejor eficiencia y una menor tasa de falsas alarmas sobre los enfoques individuales [25].

Ahora bien, así como se pueden tener 3 enfoques, y mejorarlos a 5, hay trabajos que profundizan y especifican en 9 grandes enfoques y hasta subdivididos en 28 enfoques diferentes, que no es intención especificar en este marco teórico, pero si es necesario contemplar para conocer dentro de nuestro contexto o panorama del entorno. Se muestran en la Figura 2-1, y es tomada del extenso paper de la revista Security Comm. Networks [49] del año 2016.

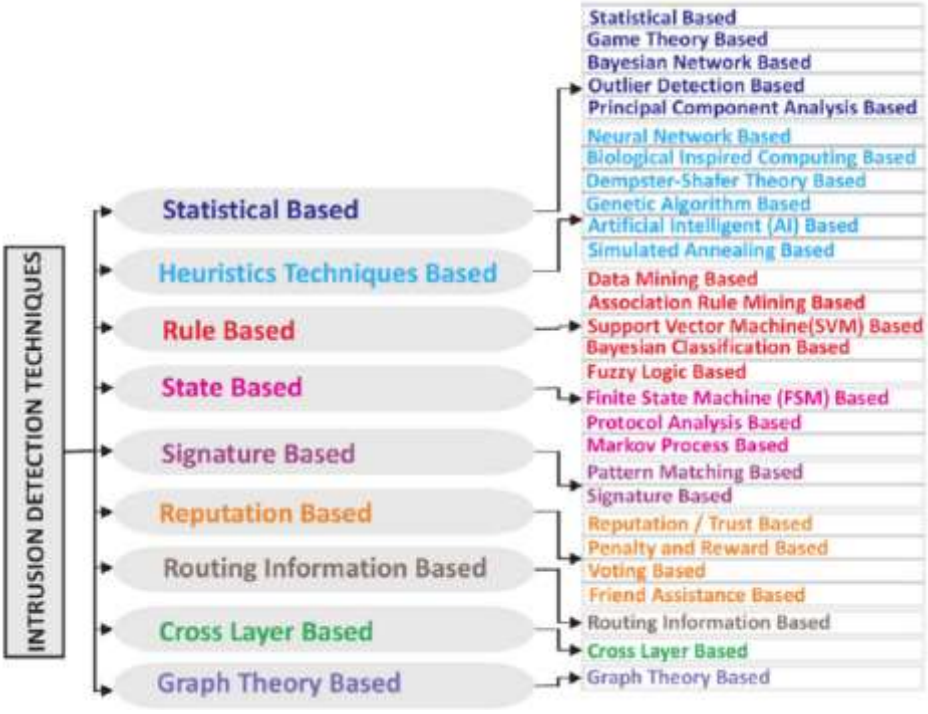


Figura 2-1 Diferentes enfoques de detección de intrusión [27].

2.6. Tipos de tecnología de detección

Categorizamos las tecnologías en cuatro clases según el lugar en el que se despliegan para inspeccionar las actividades sospechosas y los tipos de eventos que pueden reconocer (Mukherjee et al., 1994; Stavroulakis y Stamp, 2010; Sabahi y Movaghar, 2008; Modi et al., 2012). Las cuatro clases: IDS basados en el host (HIDS), IDS basados en la red (NIDS), IDS basados en la tecnología inalámbrica (WIDS), análisis del comportamiento de la red (NBA) e IDS mixtos (MIDS). Un HIDS

supervisa y recoge las características de los hosts que contienen información sensible, los servidores que ejecutan servicios públicos y las actividades sospechosas. Un NIDS captura el tráfico de red en segmentos específicos de la red a través de sensores, y posteriormente, analiza las actividades de las aplicaciones y protocolos para reconocer incidentes sospechosos. Los WIDS son similares a los NIDS, pero capturan el tráfico de las redes inalámbricas, como las redes ad hoc, las redes de sensores inalámbricos y las redes de malla inalámbricas. Además, un sistema NIDS inspecciona el tráfico de red para reconocer ataques con flujos de tráfico inesperados. La adopción de múltiples tecnologías como MIDS puede cumplir el objetivo de una detección más completa y precisa [25].

2.7. Taxonomía completa Detección de Intrusiones

Siguiendo la lógica que hemos seguido en la clasificación y categorización de las diferentes formas y proseguir de la detección de intrusión como la metodología, el enfoque, tipo de tecnología; hay muchas más clasificaciones, dónde la literatura a veces se enreda o cierta bibliografía no contempla de forma completa.

En la siguiente Figura 2-2, clasifican en 4 grandes grupos: estrategia de detección, implementación del sistema, fuente de la data y la línea de tiempo; tomado de un gran trabajo [25] publicado en *Journal of Network and Computer Applications* del 2013.

2.8. Técnicas de Detección de intrusiones basadas en Anomalías

Vamos, a especificar y aunar más en esta técnica de anomalías, ya que es la que nos compete, y ver cómo ha evolucionado, y existe diferencias de tratar eso

anómalo o fuera de lo “normal”. Se trata de técnicas basadas en la estadística, en la cognición o en el conocimiento, en el aprendizaje automático o en la computación blanda, en la minería de datos, en la identificación de la intención del usuario y en la inmunología informática [30]. Según el tipo de procesamiento relacionado con el

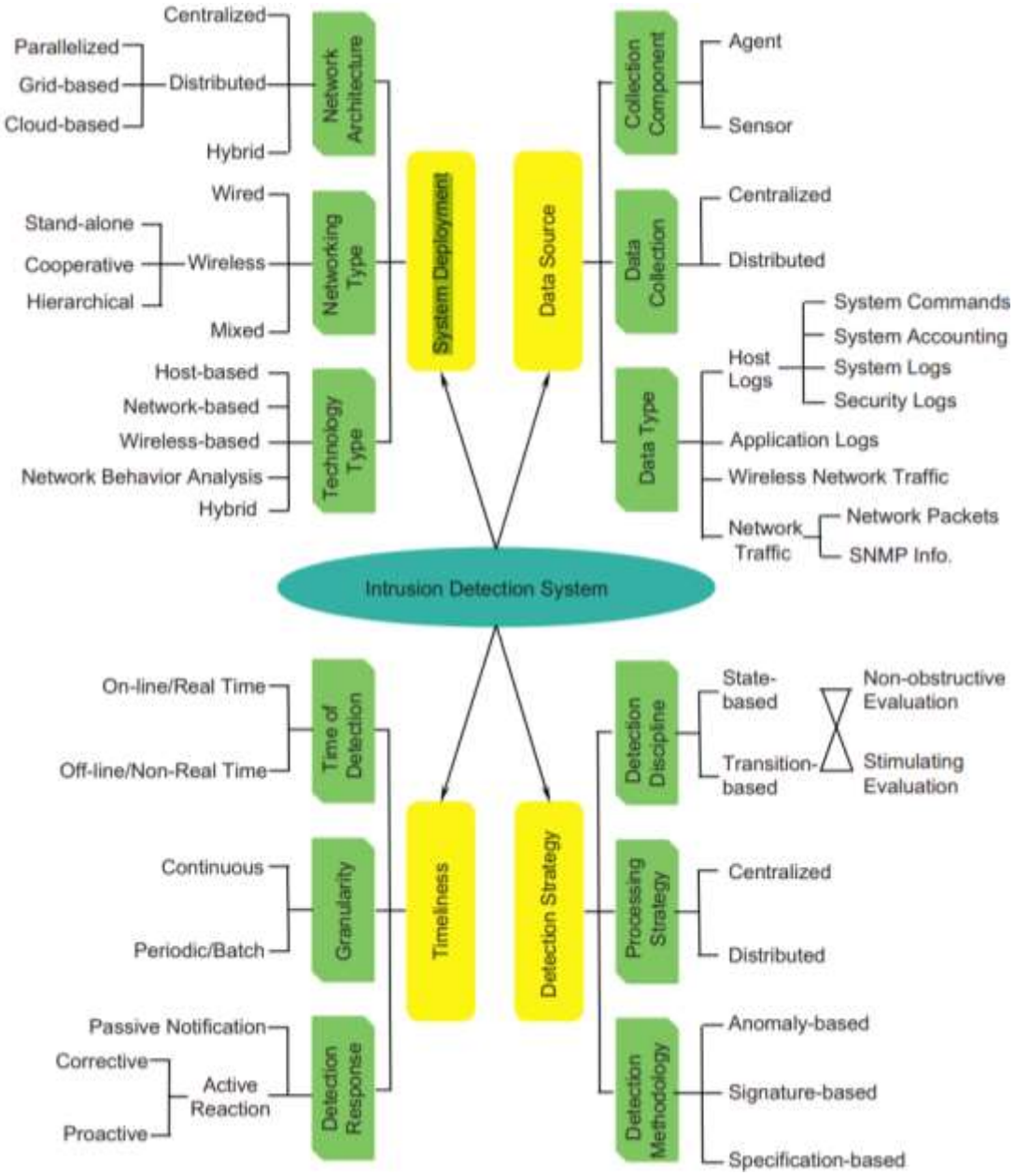


Figura 2-2 Taxonomía completa de detección de intrusión [25].

modelo de "comportamiento" del sistema objetivo, las técnicas de detección de anomalías pueden clasificarse en tres categorías principales (Lazarevic et al., 2005) basadas en la estadística, en el conocimiento y en el aprendizaje automático [31].

Los métodos pueden clasificarse en tres grupos principales: *Basados en la estadística* (Chao et al., 2015), *basados en el conocimiento* (Elhag et al., 2015; Can & Sahingoz, 2015), y *basados en el aprendizaje automático* (Buczak & Guven, 2016; Meshram & Haas, 2017). El enfoque basado en la estadística implica la recopilación y el examen de cada registro de datos en un conjunto de elementos y la construcción de un modelo estadístico del comportamiento normal del usuario. Por otro lado, el basado en el conocimiento trata de identificar las acciones solicitadas a partir de los datos existentes del sistema, como las especificaciones de los protocolos y las instancias de tráfico de la red, mientras que los métodos de aprendizaje automático adquieren complejas capacidades de coincidencia de patrones a partir de los datos de entrenamiento [32].

2.8.1. Técnicas basadas en la estadística

Un IDS basado en la estadística construye un modelo de distribución para el perfil de comportamiento normal, luego detecta los eventos de baja probabilidad y los marca como intrusiones potenciales. El sistema de detección de intrusiones anómalo (AIDS) estadístico tiene en cuenta esencialmente las métricas estadísticas, como la mediana, la media, la moda y la desviación estándar de los paquetes. En otras palabras, en lugar de inspeccionar el tráfico de datos, se supervisa cada paquete, lo que significa la huella digital del flujo.

Los IDS estadísticos se emplean para identificar cualquier tipo de diferencias en el comportamiento actual con respecto al comportamiento normal. Los AIDS estadísticos suelen utilizar uno de los siguientes modelos: **Univariante**, significa que los datos sólo tienen una variable. Esta técnica se utiliza cuando se crea un perfil estadístico normal para una sola medida de los comportamientos en los

sistemas informáticos. Los IDS univariantes buscan anomalías en cada métrica individual (Ye et al., 2002). **Multivariante**, Se basa en las relaciones entre dos o más medidas para comprender las relaciones entre las variables. Este modelo sería valioso si los datos experimentales muestran que se puede conseguir una mejor clasificación a partir de combinaciones de medidas correlacionadas en lugar de analizarlas por separado. Ye et al. examinan un método de control de calidad multivariante para identificar intrusiones mediante la construcción de un perfil a largo plazo de las actividades normales (Ye et al., 2002). El principal reto de las identificaciones estadísticas multivariantes es que resulta difícil estimar las distribuciones para los datos de alta dimensión. **Modelo de series temporales,** Una serie temporal es una serie de observaciones realizadas en un intervalo de tiempo determinado. Una nueva observación es anormal si su probabilidad de ocurrir en ese momento es muy baja. Viinikka et al. utilizaron series temporales para procesar agregados de alertas de detección de intrusión (Viinikka et al., 2009). Qingtao et al. presentaron un método para detectar anomalías en la red examinando la variación brusca encontrada en los datos de las series temporales (Qingtao & Zhiqing, 2005). La viabilidad de esta técnica se validó mediante experimentos simulados [32].

2.8.2. Técnicas basadas en el conocimiento

Este grupo de técnicas también se denomina **método de sistema experto**. Este enfoque requiere la creación de una base de conocimientos que refleje el perfil de tráfico legítimo. Las acciones que difieren de este perfil estándar se tratan como una intrusión. A diferencia de las otras clases de AIDS, el modelo de perfil estándar se crea normalmente a partir del conocimiento humano, en términos de un conjunto de reglas que intentan definir la actividad normal del sistema. La principal ventaja de las técnicas basadas en el conocimiento es la capacidad de reducir las falsas alarmas positivas, ya que el sistema tiene conocimiento de todos los comportamientos normales. Sin embargo, en un entorno informático que cambia dinámicamente, este tipo de IDS necesita una actualización periódica de los

conocimientos sobre el comportamiento normal esperado, lo cual es una tarea que requiere mucho tiempo, ya que reunir información sobre todos los comportamientos normales es muy difícil.

Máquina de estados finitos (FSM) es un modelo de computación utilizado para representar y controlar el flujo de ejecución. Este modelo podría aplicarse en la detección de intrusos para producir un modelo de sistema de detección de intrusos. Normalmente, el modelo se representa en forma de estados, transiciones y actividades. Un estado comprueba los datos del historial. Por ejemplo, se anota cualquier variación en la entrada y, en función de la variación detectada, se produce una transición (Walkinshaw et al., 2016). Un FSM puede representar el comportamiento legítimo del sistema, y cualquier desviación observada de este FSM se considera un ataque.

El **Lenguaje de descripción** define la sintaxis de las reglas que pueden utilizarse para especificar las características de un ataque definido. Las reglas podrían construirse mediante lenguajes de descripción como **N-grammars** y **UML** (Studnia et al., 2018). Un sistema experto comprende una serie de reglas que definen los ataques. En un sistema experto, las reglas suelen ser definidas manualmente por un ingeniero del conocimiento que trabaja en colaboración con un experto del dominio (Kim et al., 2014).

El **Análisis de firma** es la primera técnica aplicada en los IDS. Se basa en la sencilla idea de la comparación de cadenas. En el cotejo de cadenas, se inspecciona un paquete entrante, palabra por palabra, con una firma distinta. Si una firma coincide, se emite una alerta. Si no, la información del tráfico se compara con la siguiente firma de la base de datos de firmas (Kenkre et al., 2015b) [32].

2.8.3. Técnicas de Aprendizaje Automático

El aprendizaje automático (ML – *Machine Learning*) es el proceso de extracción de conocimientos a partir de grandes cantidades de datos. Los modelos

de aprendizaje automático comprenden un conjunto de reglas, métodos o "funciones de transferencia" complejas que pueden aplicarse para encontrar patrones de datos interesantes, o para reconocer o predecir comportamientos (Dua y Du, 2016). Las técnicas de aprendizaje automático se han aplicado ampliamente en el ámbito del AIDS. Varios algoritmos y técnicas como la agrupación, las redes neuronales, las reglas de asociación, los árboles de decisión, los algoritmos genéticos y los métodos del vecino más cercano, se han aplicado para descubrir el conocimiento de los conjuntos de datos de intrusión (Kshetri & Voas, 2017; Xiao et al, 2018).

Algunas investigaciones anteriores han examinado el uso de diferentes técnicas para construir AIDS. Chebrolu et al. examinaron el rendimiento de dos algoritmos de selección de características que incluían redes bayesianas (BN) y árboles de regresión de clasificación (CRC) y combinaron estos métodos para obtener una mayor precisión (Chebrolu et al., 2005). Bajaj et al. propusieron una técnica para la selección de rasgos utilizando una combinación de algoritmos de selección de rasgos como la ganancia de información (IG) y la evaluación de atributos de correlación. Probaron el rendimiento de las características seleccionadas aplicando diferentes algoritmos de clasificación como C4.5, naïve Bayes, NB-Tree y Multi-Layer Perceptron (Khraisat et al., 2018; Bajaj & Arora, 2013). Se ha utilizado un método de minería de reglas genético-fuzzy para evaluar la importancia de las características del IDS (Elhag et al., 2015). Thaseen et al. propusieron NIDS utilizando el modelo de árbol aleatorio para mejorar la precisión y reducir la tasa de falsas alarmas (Thaseen & Kumar, 2013). Subramanian et al. propusieron clasificar el conjunto de datos NSL-KDD utilizando algoritmos de árboles de decisión para construir un modelo con respecto a sus datos métricos y estudiando el rendimiento de los algoritmos de árboles de decisión (Subramanian et al., 2012).

Se han creado varios AIDS basados en técnicas de AA. El objetivo de utilizar técnicas de AA es crear IDS con mayor precisión y con menos necesidad de conocimientos humanos. En los últimos años ha aumentado la cantidad de AIDS que han utilizado métodos de aprendizaje automático. Un objetivo clave de los IDS basados en la investigación del AA es detectar patrones y construir un sistema de detección de intrusiones basado en el conjunto de datos. En general, hay dos tipos de métodos de AA, supervisados y no supervisados [32].

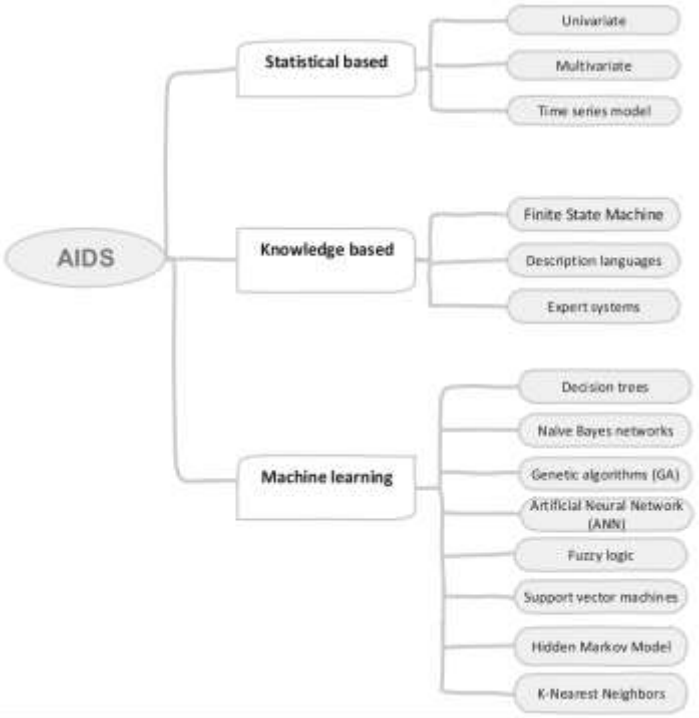


Figura 2-3 Técnicas detección de intrusos basadas anomalías [32].

El ML se puede caracterizar como la capacidad de un programa o potencialmente un marco de trabajo para aprender y mejorar su rendimiento en una tarea específica o grupo de tareas a lo largo del tiempo [33]. Las estrategias de ML hacen hincapié en la construcción de un marco que mejore su ejecución basándose en los resultados anteriores, es decir, que pueda cambiar su estrategia de ejecución basándose en los datos adquiridos recientemente. Su principal ventaja es la

flexibilidad, la adaptabilidad y la captación de interdependencias. La desventaja es la alta complejidad algorítmica y los largos tiempos de entrenamiento [30].

Se debe indicar que ésta clasificación de 3 tipos de la Figura 2-3, es la más general; revisando en profundidad se puede detallar una mejor y detallada división, con diferentes tipos de técnica basada en anomalías, como lo presenta la Figura 2-4.

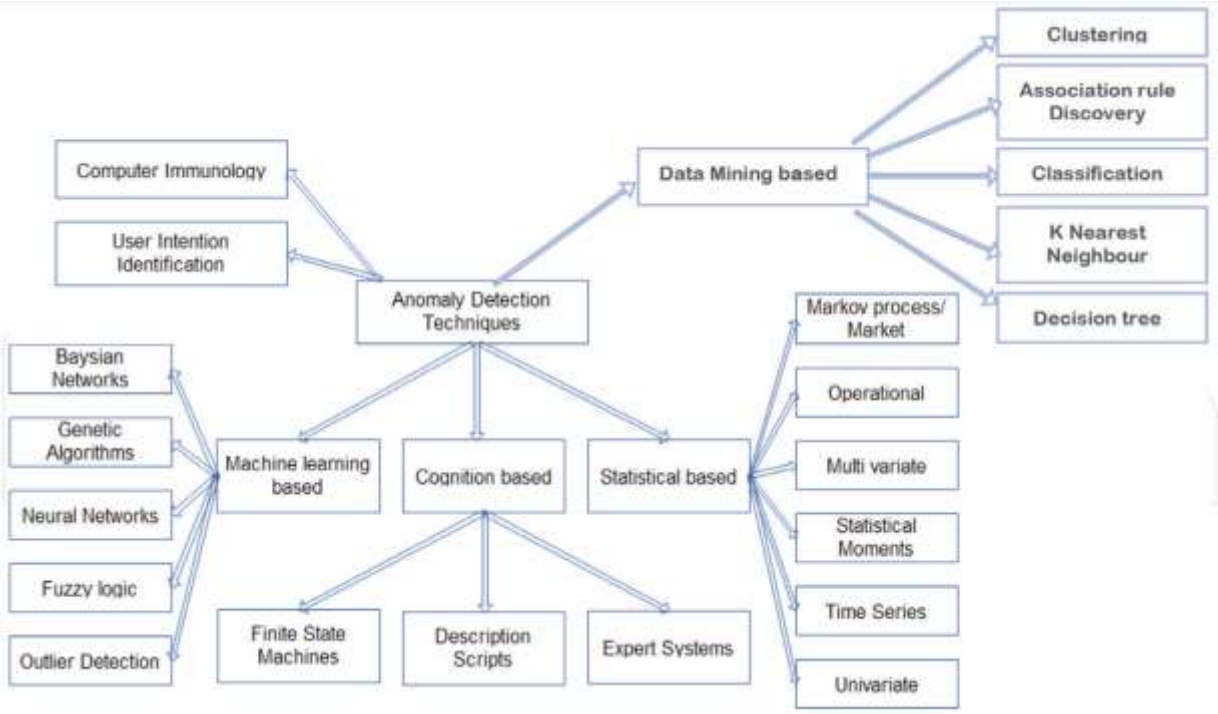


Figura 2-4 Técnicas detección de intrusos basadas anomalías [30].

2.9. Ataque DoS y ataque DDoS

El ataque de denegación de servicio (DoS) es uno de los más populares en Internet. Se lleva a cabo forzando el lanzamiento de un ordenador secuestrado o consumiendo sus recursos, como el ciclo de la CPU, la memoria y el ancho de banda de la red. Cuando el ataque DoS es generado por una gran variedad de ordenadores distribuidos, se denomina Denegación de Servicio Distribuida (DDoS). El DDoS se

ha convertido en uno de los principales retos de la ciberseguridad actual. El ataque DDoS es lanzado por algunos *Zombies* controlados a distancia. Impide que los usuarios legítimos accedan a algunos servicios de red específicos o paraliza los propios servicios de las víctimas ocupando parcial o totalmente los recursos informáticos o el ancho de banda de la red. Si hay más paquetes de datos de tráfico anormal y más hosts *Zombies* secuestrados, se producen más daños en la red. Si el número de hosts *Zombies* es lo suficientemente grande, puede llegar a perturbar todo el entorno de la red y todos los servidores de forma fugaz [34].

En el verano de 1999, la *Computer Incident Advisory Capability* (CIAC) informó del primer incidente de ataque DDoS [61]. Desde entonces, el DDoS se ha convertido en el medio de ataque más conveniente y eficaz utilizado por los hackers. En el año 2000, es la respuesta dada por los sitios de Internet (por ejemplo, Microsoft, Yahoo y Amazon) a los que no se puede acceder durante mucho tiempo, debido a un grave ataque DDoS.

Los ataques DDoS se clasifican principalmente en tres categorías en función de los diferentes sujetos atacados. El primer tipo se llama ataque **Netflow-DDoS** y hay muchos casos típicos como el ataque de amplificación DNS, el ataque de amplificación SNMP, la inundación UDP y la inundación ICMP. El segundo es el ataque **Connection-DDoS** SYN Flood y TCP Flood son los casos de ataque más influyentes. Además, hay un tipo de ataque **DDoS basado en la aplicación**, como HTTP Get Flood y SSL Flood [34].

A pesar de todos los esfuerzos realizados por la industria y el mundo académico, los ataques DDoS siguen siendo un problema abierto. En los últimos años, la técnica y el nivel de los ataques DDoS están avanzando incesantemente con la mejora de la capacidad de detección de los ataques. Con la aparición de la tecnología **Big Data**, es particularmente mucho más difícil que nunca prevenir la red de varios ataques DDoS. El continuo crecimiento del tráfico de la red hace imposible

detectar el comportamiento de los ataques de la red a partir de una escala tan grande de tráfico de la red basándose en los métodos de detección anteriores [34].

Has podido ver que existe el ataque DoS y el DDoS, una diferencia que encontramos entre un ataque y otro, es que en el primero utilizamos una máquina y una conexión y en el segundo se van a utilizar muchos computadores, por lo que el ataque es mucho más duro y efectivo.

2.9.1. Tipos de ataque DoS y DDoS

A lo largo de los años, los ciberdelincuentes han desarrollado una serie de enfoques técnicos para eliminar objetivos en línea a través de DDoS. Las técnicas individuales tienden a dividirse en tres tipos generales de ataques DDoS [35]:

- ***Ataques volumétricos*** El tipo clásico de DDoS, estos ataques emplean métodos para generar volúmenes masivos de tráfico para saturar completamente el ancho de banda, creando un atasco de tráfico que hace imposible que el tráfico legítimo ingrese o salga del sitio objetivo.
- ***Ataques de protocolo*** están diseñados para consumir la capacidad de procesamiento de los recursos de la infraestructura de red, como servidores, firewalls y equilibradores de carga, al dirigirse a las comunicaciones de protocolo de Capa 3 y Capa 4 con solicitudes de conexión maliciosas.
- ***Ataques de aplicaciones*** Algunos de los ataques DDoS más sofisticados aprovechan las debilidades en la capa de la aplicación, Capa 7, al abrir conexiones e iniciar procesos y solicitudes de transacciones que consumen recursos finitos como espacio en disco y memoria disponible.

Se debe tener en cuenta que, en escenarios de ataque del mundo real, a los cibercriminales les gusta mezclar y combinar este tipo de ataques para aumentar el dolor. Por lo tanto, una sola campaña DDoS puede superponer ataques de aplicaciones y protocolos además de los ataques volumétricos [35].

Los ataques DDoS varían según la capa de la red informática a la que se dirigen. Ejemplos incluyen [36]:

- **Capa 3**, la capa de red. Los ataques se conocen como **Smurf Attacks**, **ICMP Floods** y **Fragmentación de IP / ICMP**.
- **Capa 4**, la capa de transporte. Los ataques incluyen **SYN Floods**, **UDP Floods** y el **agotamiento de la conexión TCP**.
- **Capa 7**, la capa de aplicación. Principalmente, ataques encriptados por HTTP.

Además, hay docenas de subtipos que caen en cualquiera de los grupos genéricos anteriores, pero exhiben características únicas. Se ve un desglose completo de los métodos de ataque DDoS actuales [37]:

- **Inundación SYN.** Este ataque aprovecha el protocolo de enlace de tres vías TCP, una técnica que se utiliza para establecer cualquier conexión entre un cliente, un host y un servidor mediante el protocolo TCP. Normalmente, un cliente envía un mensaje SYN (sincronización) al servidor para solicitar una conexión. Cuando se produce un ataque SYN Flood, los delincuentes envían una gran cantidad de estos mensajes desde una dirección IP falsificada. Como resultado, el servidor receptor se vuelve incapaz de procesar y almacenar tantos paquetes SYN y niega el servicio a clientes reales.

- **Inundación UDP.** Como sugiere el nombre, este ataque DDoS aprovecha varios paquetes del **Protocolo de datagramas de usuario** (UDP). Para el registro, las conexiones UDP carecen de un mecanismo de negociación (a diferencia de TCP) y, por lo tanto, las opciones de verificación de la dirección IP son muy limitadas. Cuando esta explotación está en pleno apogeo, el volumen de paquetes ficticios excede la capacidad máxima del servidor de destino para procesar y responder a las solicitudes.

- **Slowloris.** Este ataque se destaca entre la multitud porque requiere un ancho de banda muy bajo y se puede realizar con una sola computadora. Funciona iniciando múltiples conexiones simultáneas a un servidor web y manteniéndolas abiertas durante un largo período de tiempo. El atacante envía solicitudes parciales y las complementa con encabezados HTTP de vez en cuando para asegurarse de que no alcancen una etapa de finalización. Como resultado, la capacidad del servidor para mantener conexiones simultáneas se agota y ya no puede procesar conexiones de clientes legítimos.

- **Cañón de iones de órbita baja (LOIC).** Originalmente diseñado como una herramienta de prueba de estrés de la red, LOIC se puede utilizar como arma en ataques DDoS del mundo real. Codificado en C #, este software de código abierto inunda un servidor con una gran cantidad de paquetes (UPD, TCP o HTTP) en un intento de interrumpir la operación de un objetivo. Este ataque suele estar respaldado por una *botnet* que consta de miles de máquinas y está coordinada por un solo usuario.

- **Cañón de iones de alta órbita (HOIC).** HOIC es una aplicación de acceso público que reemplazó al programa LOIC mencionado anteriormente y tiene un potencial disruptivo mucho mayor que su precursor. Se puede usar para enviar una gran cantidad de solicitudes GET y HTTP POST a un servidor al mismo tiempo, lo que termina desconectando un sitio web de destino. HOIC puede afectar hasta 256 dominios diferentes al mismo tiempo [37].

2.9.2. Vulnerabilidad explotada de los ataques DoS y DDoS

Basándose en la exposición que se aprovecha, los ataques DDoS pueden clasificarse de la siguiente manera, el protocolo explota los ataques de paquetes malformados y la amplificación [38].

- Un **ataque de inundación** consiste en congestionar el sistema de la víctima con ancho de banda mediante la transmisión de una gran cantidad de tráfico al sistema de la víctima. Si esto ocurre, el sistema de la víctima puede sufrir una reducción de la velocidad, una avería del sistema o incluso la saturación del ancho de banda. Los ataques de inundación incluyen el protocolo UDP y el protocolo de Mensajes de Control de Internet (ICMP). En el ataque UDP, es posible transmitir una gran cantidad de paquetes UDP al sistema de la víctima, lo que provoca la saturación de la red y la reducción de la disponibilidad del ancho de banda para el servicio autorizado en el sistema de la víctima. El ataque DDoS basado en UDP implica la transmisión de paquetes UDP a los puertos de las máquinas de las víctimas de forma aleatoria o precisa [39]. Por el contrario, un ataque de inundación basado en UDP implica la transmisión aleatoria de paquetes UDP a los puertos de las máquinas de las víctimas. El nombre de la aplicación que espera en el puerto del sistema de destino es resuelto por la máquina de la víctima que recibe el paquete UDP. En el caso de que no se encuentre ninguna aplicación esperando en el puerto de destino, la dirección falsa recibe un paquete ICMP de "destino inalcanzable". Cuando el UDP se transmite adecuadamente, las máquinas atacadas se descomponen. La dirección IP de origen del atacante puede encontrarse con el *spoofing*, lo que impide revelar la identidad de las otras víctimas. En este caso, los paquetes que han sido enviados de vuelta al sistema de la víctima no se transmiten de vuelta a los zombis mediante el uso de una herramienta DDoS. La explotación de los ataques de inundación basados en ICMP se produce porque la transmisión de paquetes de eco a un

host remoto para la verificación del estado de los usuarios está permitida en el ICMP. En concreto, cuando se lanza un ataque de inundación DDoS basado en ICMP, se envían grandes cantidades de **ICMP-ECHO-REPLY** (ping) al sistema de la víctima. Estos paquetes solicitan una respuesta del sistema de la víctima, provocando la saturación del ancho de banda en la red de la víctima [40]. Existe una alta probabilidad de que la dirección IP de la fuente sea falsificada cuando se lanza un ataque basado en ICMP.

- Cuando se lanzan **ataques de amplificación**, las características de la dirección IP de difusión son explotadas por el atacante; la dirección IP de difusión suele encontrarse en muchos *routers*. Esto permite reforzar y reflejar el asalto y la transmisión de mensajes hacia una dirección IP de difusión. Los routers reciben la orden de transmitir los paquetes fuera de la red a cada una de las direcciones IP que se encuentran dentro del rango de la dirección de difusión. De este modo, el ancho de banda del sistema de la víctima disminuye como resultado del tráfico extra que se ha generado. En este tipo de ataques DDoS se inicia la transmisión directa o indirecta del mensaje de difusión por parte del atacante para aumentar el tráfico. Cuando el mensaje de difusión se transmite directamente, los sistemas que están dentro del alcance de la red de difusión son utilizados por el atacante sin necesidad de instalar ningún agente de software. Algunos de los ataques más conocidos son los ataques Smurf y Fraggle. En estos ataques, los reflectores son los nodos agentes que se utilizan como lanzadores [41]. Cualquier paquete que sea recibido por el nodo reflector es enviado de vuelta. Por lo tanto, los servidores DNS y web y los routers se consideran reflectores porque envían de vuelta **SYN ACKs** o **RSTs** reconociendo SYN u otros paquetes TCP. Los paquetes que requieren acuse de recibo son enviados por el atacante a los reflectores. Estos paquetes falsifican la dirección por medio de las direcciones de la fuente ajustadas a la dirección de la víctima. Los paquetes

de respuesta son enviados de vuelta por los reflectores a la víctima dependiendo del tipo de ataque de paquetes. Los paquetes utilizados para el ataque se devuelven esencialmente en los paquetes normales a la víctima. Si el paquete que se devuelve es considerablemente grande, puede hacer que el enlace de las víctimas se sobrecargue. En los paquetes del atacante que son recibidos por el sistema de la víctima, es fácil reconocer los **reflectores** como la dirección de origen. Además, el esclavo que transmite los paquetes al reflector no puede ser identificado por el operador del reflector, porque la dirección de origen del esclavo no está contenida en los paquetes que se transmiten al reflector, sino que es la dirección de origen de la víctima la que está contenida en ellos. El patrón de ataque de los ataques al reflector es similar al utilizado para los ataques directos. Sin embargo, existen algunas variaciones notables [42]. Un ataque de reflector necesita un conjunto de reflectores que han sido preestablecidos. Los reflectores pueden propagarse utilizando Internet porque no se requiere la instalación de un agente de software.

- En los **ataques de explotación de protocolos**, se aprovecha un atributo específico o el error de implementación de un determinado protocolo existente en el sistema de la víctima, para poder consumir sus recursos adicionales. Un ejemplo clásico de este tipo de ataques es **TCP-SYN**. Los ataques basados en TCP-SYN se aprovechan de las limitaciones naturales del apretón de manos a tres vías (*three-way handshake*) que se encuentra en la conexión de configuración de TCP. Un servidor devuelve como un paquete de eco **SYN/ACK** (sincronización/reconocimiento) y espera al cliente después de que la primera solicitud **SYN** (sincronización/inicio) haya sido aceptada por un cliente para transmitir el paquete **ACK** (reconocimiento) final. Un ataque de inundación basado en SYN es iniciado por un atacante a través de la transmisión de un gran número de paquetes SYN sin reconocer

las respuestas. Esto hace que el servidor espere **ACKs** que no existen [43]. Mediante el ataque de inundación basado en SYN, el servidor se vuelve incapaz de procesar otras peticiones porque se produce una sobrecarga de colas en los servidores con una cola limitada para amortiguar nuevos enlaces [44]. Algunos de los ataques de explotación de protocolos conocidos que atacan a los servidores de autenticación son los ataques **PUSH +ACK**.

- **Ataques con paquetes malformados** (*malformed packet attacks*) [45]: este tipo de ataque se basa en paquetes IP que han sido producidos y transmitidos erróneamente desde los agentes al sistema de la víctima con el objetivo de apagarlo. Las categorías básicas de estos ataques son los ataques a paquetes IP y a direcciones IP en los que el destino del paquete IP y la dirección IP es el mismo. Por esta razón, el sistema operativo del sistema de la víctima se confunde y luego se bloquea. En un ataque a paquetes IP, los paquetes mal formados pueden desordenar los campos opcionales del paquete IP y poner cada uno de los bits de calidad a 1. En consecuencia, el tiempo de análisis de tráfico adicional es entonces consumido a la fuerza por el sistema de la víctima. En este caso, cuando muchos agentes forman parte del ataque, el sistema de la víctima se bloquea.

2.10. Detección de ataques basada en flujos

Los datos de flujo contienen paquetes agrupados en un periodo, que es la fuente de datos más extendida para los IDS. Los conjuntos de datos CIC son todos datos de flujo. La detección de ataques con flujo tiene dos ventajas: la primera, el flujo representa todo el entorno de la red, lo que permite detectar la mayoría de los ataques. La Segunda, sin análisis de paquetes ni reestructuración de sesiones, el preprocesamiento de flujos es sencillo. La detección de ataques basada en flujos

incluye principalmente métodos de **ingeniería de atributos** (*feature engineering*) [46].

Para la detección basada en la ingeniería de características los modelos tradicionales de aprendizaje automático no pueden abordar directamente los datos de flujo; por lo tanto, la ingeniería de características es un paso esencial antes de poder aplicar estos modelos. Los métodos basados en la ingeniería de rasgos adoptan un modo de **vectores de atributos + modelos superficiales** (*feature vectors* + *shallow models*). Los vectores de atributos son adecuados para la mayoría de los algoritmos de aprendizaje automático. Cada dimensión de los vectores de características tiene una semántica claramente interpretable. Las características comunes incluyen la longitud media de los paquetes, la varianza de la longitud de los paquetes, la proporción de TCP a UDP, la proporción de banderas TCP, etc. Las ventajas de este tipo de métodos de detección son que son sencillos de implementar, muy eficientes y pueden cumplir con los requisitos de tiempo real [46].

2.11. Clasificación de Aprendizaje Supervisado

La clasificación en el aprendizaje automático y la estadística es un enfoque de aprendizaje supervisado en el que el programa informático aprende de los datos que se le dan y hace nuevas observaciones o clasificaciones.

La clasificación es un proceso de categorización de un conjunto dado de datos en clases, que puede realizarse tanto en datos estructurados como no estructurados. El proceso comienza con la predicción de la clase de los puntos de datos dados. Las clases suelen denominarse objetivo, etiqueta o categorías. La modelización predictiva de la clasificación es la tarea de aproximar la función de mapeo de las variables de entrada a las variables de salida discretas. El objetivo principal es identificar en qué clase/categoría caerán los nuevos datos [47].

Las técnicas de IDS basadas en el aprendizaje supervisado detectan las intrusiones utilizando datos de entrenamiento etiquetados. Un enfoque de aprendizaje supervisado suele constar de dos etapas, a saber, el entrenamiento y la prueba. En la **etapa de entrenamiento**, se identifican las características y clases relevantes y luego el algoritmo aprende de estas muestras de datos. En los IDS de aprendizaje supervisado, cada registro es un par, que contiene una red o fuente de datos del host y un valor de salida asociado (es decir, una etiqueta), a saber, intrusión o normal. A continuación, se puede aplicar la selección de características para eliminar las innecesarias. A partir de los datos de entrenamiento de las características seleccionadas, se utiliza una técnica de aprendizaje supervisado para entrenar un clasificador que aprenda la relación inherente que existe entre los datos de entrada y el valor de salida etiquetado. En la literatura se ha explorado una amplia variedad de técnicas de aprendizaje supervisado, cada una con sus ventajas y desventajas. En la **fase de prueba**, el modelo entrenado se utiliza para clasificar los datos desconocidos en clase de intrusión o normal. El clasificador resultante se convierte en un modelo que, dado un conjunto de valores de características, predice la clase a la que podrían pertenecer los datos de entrada. El rendimiento de un clasificador en cuanto a su capacidad para predecir la clase correcta se mide en términos de una serie de métricas [32].

2.11.1. Algoritmos de aprendizaje supervisado

Árboles de Decisión

Un árbol de decisión consta de tres componentes básicos. El primer componente es un **nodo de decisión**, que se utiliza para identificar un atributo de prueba. El segundo es una **rama**, donde cada rama representa una posible decisión basada en el valor del atributo de prueba. El tercero es una **hoja** que comprende la clase a la que pertenece la instancia (Rutkowski et al., 2014). Hay muchos

algoritmos de árboles de decisión diferentes, como **ID3** (Quinlan, 1986), **C4.5** (Quinlan, 2014) y **CART** (Breiman, 1996) [32].

El algoritmo de árbol de decisión clasifica los datos mediante una serie de reglas. El modelo tiene forma de árbol, lo que lo hace interpretable. El algoritmo de árbol de decisión puede excluir automáticamente las características irrelevantes y redundantes. El proceso de aprendizaje incluye la selección de características, la generación del árbol y la poda del mismo. Al entrenar un modelo de árbol de decisión, el algoritmo selecciona individualmente las características más adecuadas y genera nodos hijos a partir del nodo raíz. El árbol de decisión es un clasificador básico. Algunos algoritmos avanzados, como el bosque aleatorio y el **boosting** de gradiente extremo (**XGBoost**), constan de múltiples árboles de decisión [46].

Métodos de conjunto (*Ensemble methods*)

Se pueden utilizar múltiples algoritmos de aprendizaje automático para obtener un mejor rendimiento predictivo que cualquiera de los algoritmos de aprendizaje constitutivos por sí solos. Se han propuesto diferentes métodos de conjunto, como **Boosting**, **Bagging** y **Stacking**. Boosting se refiere a una familia de algoritmos capaces de transformar los aprendices débiles en fuertes. Bagging significa entrenar el mismo clasificador en diferentes subconjuntos del mismo conjunto de datos. El apilamiento combina varias clasificaciones mediante un metaclasificador (Aburomman y Reaz, 2016). Los modelos de nivel básico se construyen sobre la base de un conjunto de entrenamiento completo y, a continuación, el metamodelo se entrena sobre las salidas del modelo de nivel básico como atributos. **Random Forest (RF)** mejora la precisión y reduce las falsas alarmas (Jabbar et al., 2017) [32].

Cada clasificador individual tiene sus puntos fuertes y sus defectos. Un enfoque natural es combinar varios clasificadores débiles para implementar un

clasificador fuerte. Los **métodos de conjuntos** entrenan múltiples clasificadores; luego, los clasificadores votan para obtener los resultados finales [46].

2.11.2. Matriz de Confusión clasificación multiclase

Las métricas típicas que se utilizan en la multiclase son las mismas que se utilizan en el caso de la clasificación binaria. La métrica se calcula para cada clase al procesarla como un problema de clasificación binaria después de agrupar todas las otras clases como pertenecientes a la segunda clase. A continuación, se calcula el promedio de la métrica entre todas las clases para obtener una métrica de **promedio macro** (procesar cada clase como igual) o de **media ponderada** (ponderada por frecuencia de clase).

Por definición, la entrada de una matriz de confusión es el número de observaciones que realmente están en las filas $\boldsymbol{i}$, pero que se predice que están las columnas $\boldsymbol{j}$.

La matriz de confusión de una clasificación con **n** clases, al considerar la clase **k (0<k<n)**, se pueden obtener los cuatro resultados de clasificación diferentes: **verdadero positivo** (verde), **verdadero negativo** (naranja), **falso positivo** (marrón) y **falso negativo** (rojo) [48].

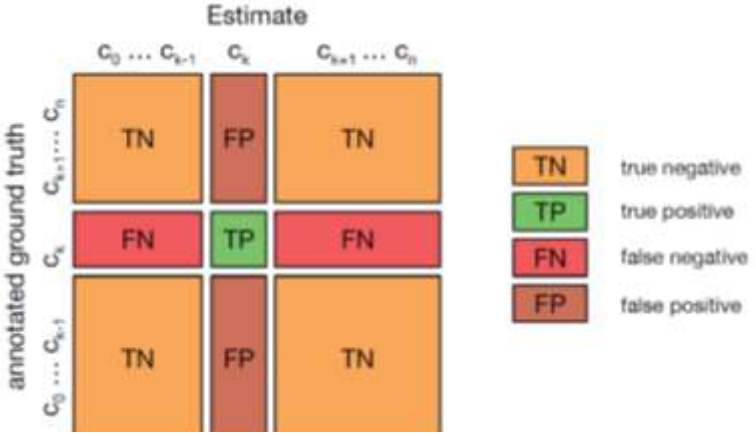


Figura 2-5 Matriz de Confusión clasificación multiclase [48]

Capítulo 3

Metodología Detección de ataques DoS y DDoS

3.1. Introducción

En el presente capitulo se presentará al lector la metodología del Trabajo de Grado de investigación, en el cual se ha diseñado un modelo detector y clasificador de ataques maliciosos de tipo de Denegación de Servicio aplicando técnicas de Aprendizaje Automático logrando la identificación y clasificación, tanto de flujos benignos como flujos maliciosos de denegación analizados a partir de los biflujos de tráfico de red.

Las familias o clases a clasificar son un total de 10 tipos de ataques maliciosos DDoS y DoS, puntualmente 6 tipos de ataques DDoS (**SYN-Flood**, **UDP-Flood**, **UDPLag**, **HOIC**, **LOIC-HTTP** y **LOIC-UDP**) y 4 tipos de ataques DoS (**Hulk**,

SlowHTTPTest, **GoldenEye** y **Slowloris**). Así se establece que se ha abordado un problema de clasificación multiclase y además de naturaleza de ML supervisado, gracias al gran valor que contiene un conjunto de datos etiquetados. Así es como se toca el primer gran pilar de inicio en el contexto de ML, decidir qué conjunto de datos (*Dataset*) se va a usar para el entrenamiento y prueba, en otras palabras, decidir cuál es la entrada dada a el modelo de caja negra de Aprendizaje Automático, y que depende en gran medida sobre la calidad del conjunto de datos elegido.

La elección de cuál usar de todos los conjuntos de datos para DoS y DDoS que existen en el estado del arte, llámese DARPA98(MIT Lincoln), KDD98, KDDCUP 99, UCLA 2001, DEFCON-8(2000), DEFCON-10(2002), CAIDA 2002-2007-2016, LBNL 2004, CDX 2009, Kyoto 2009, WAIKATO 2009, 2009 DARPA, NSLKDD 09, TUIDS 2012, ISCX 2012, FRGP 2013, BOOTER 2014, ADFA-LD and ADFA-WD 2014, UNSW-NB15, **CIC-IDS2017**, **CSE-CIC-IDS2018**, **CIC-DDoS2019**, entre otros, se pueden comparar en un marco de valoración de conjuntos de datos (Gharib et al., 2016), donde se identifica once criterios que son necesarios para construir un conjunto de datos de referencia fiable [49] para la investigación y el mundo real. En esa misma decisión interfiere dos criterios vitales, se trata tanto de la configuración de los escenarios que determina qué tipo de topología de red fue montada para adquirir la data recolectada, incluyendo dispositivos de red como Módem, Cortafuegos, Conmutadores, Enrutadores, y presencia de una variedad de sistemas operativos como Windows, Ubuntu y Mac OS X; como de los atributos y características que presentan dichos conjuntos de datos de los flujos de tráfico de red. Para nuestro caso de conjuntos CIC, se usa los 84 atributos estadísticos implementados a través de la herramienta de la UNB el CICFlowMeter.

El preprocesamiento donde se manipula los datos en crudo, en primario, y se le da forma adecuada a los datos de los problemas que presenta: como repetición

de los nombres de atributos como filas, de espacios en blanco al inicio e intermedio, de los caracteres de no palabra como “/”, letras en mayúscula, y diferencias en el escrito de los nombres, así como celdas con valores infinitos, y celdas con valores negativos. Se prosigue después de dar forma a los datos, a una etapa de normalización o escalado de los atributos, se indicará la vivencia desde los resultados experimentales. Se continua con un proceso muy importante y vital, y que consume la mayor parte del tiempo, y decide sobre el logro ó no del objetivo de clasificar óptimamente. Aquí se transforma el conjunto de datos, reduciendo su dimensionalidad en el número de atributos. Con el conjunto de datos ya procesado, tanto el separado en matriz de atributos (X), como el vector de etiquetas (y) cómo la división entre entrenamiento y prueba, se prosigue a la etapa de aprendizaje, tomando únicamente el conjunto de entrenamiento, aquí se compararán resultados experimentales de más de 20 diferentes algoritmos de aprendizaje entrenados. Dicha comparación se hace en base al desempeño y rendimiento del entrenamiento.

Los resultados de entrenamiento se seleccionan los mejores estimadores para formar el Modelo y se descartan los demás algoritmos de aprendizaje. Ahora con este modelo y el conjunto de datos de prueba, que no se ha usado y que son datos que no han sido vistos, se continua con la siguiente etapa, el cual consiste en la etapa de evaluación donde se ha valorado cuantitativamente el desempeño y rendimiento del conjunto de prueba con métricas y representaciones gráficas, para estimar el error de generalización. Tal la insistencia de querer evaluar de datos no visto que se ha realizado doble evaluación, aplicado en 2 conjuntos de prueba: el mencionado anteriormente que es caso normal, y corresponde a la elección del 10% estratificado (más de millón trecientos de flujos) del conjunto total; como de un segundo conjunto de datos creado a partir de la data CIC-DDoS2019 usando 3 ataques que se tienen las etiquetas y no ha sido usadas en la etapa de entrenamiento. Se prosigue hacer un paréntesis a modo de vista en retrospectiva, y se menciona por indicaciones del tutor a expresar un apartado del capítulo de

efectividad computacional, a darle importancia como proceso experimental vivido: a especificar y puntualizar que durante todo el proceso de manejar esta *Big Data* en nuestro caso de ataques DoS y DDoS, supero los requerimientos de la maquina(ordenador) disponible y tareas “sencillas” que no se pueden realizar, porque se presentan reinicios en el núcleo de la memoria RAM al leer un archivo superior a 15 millones de flujos con 84 columnas (>5GB .csv), o querer realizar una optimización de hiperparámetros. No solo fue esa imposibilidad que volcaba la RAM, es el coste computacional donde se convergiera en un “tiempo razonablemente humano”, por mencionar que se tuvieron algoritmos en etapa de aprendizaje que demoraron 14 horas para brindar un resultado como el Gradiente Boosting; y otros como KNN que en un tiempo de más de 22 horas fue interrumpido porque no se sabía cuánto más demoraba, o el caso de SVM que también se dejó por 9 horas, y se canceló.

Después de los resultados, se prosigue al apartado final donde se le denomino significación estadística, donde se realiza una etapa de implementación del modelo final, para la detección y clasificación tanto en tiempo real: sea capturado, esnifado en directo a través de una interfaz de tarjeta de red; como de la docena de capturas de paquetes previamente almacenados, como es el ejemplo de nuestro caso de trafico malicioso de denegación que contiene trazas y flujos de ataques representativos encontrados en una búsqueda exhaustiva en la fuentes electrónicas, para muestras que dicen contener ataques de denegación DoS y DDoS que validan el funcionamiento del modelo detector **DetD&DoS**.

Finaliza el capítulo presentando una hoja de ruta, un diagrama que plasma la metodología realizada, con todas sus etapas, se muestra el funcionamiento en diagramas de bloques del trabajo típico para el uso del Aprendizaje Automático en la identificación y clasificación familias/clases de flujos de tráfico malicioso, específicamente de denegación de servicio.

3.2. Selección del Conjunto de Datos

Se ha decidido la elección del conjunto de datos, una combinación de tres años del Instituto Canadiense de Ciberseguridad CIC de la Universidad New Brunswick UNB del IDS2017, IDS2018 y del DDoS2019. Un gran punto a favor, es su filosofía *open source,* está a disposición del público para los investigadores.

La selección se radica en el cumplimiento de los once criterios según Gharib, así como descartar los *datasets* desactualizados que no son fiables. Algunos de esos conjuntos de datos adolecen de la falta de diversidad y volumen de tráfico, algunos no cubren la variedad de ataques conocidos, mientras que otros anonimizan los datos de la carga útil de los paquetes, que no pueden reflejar las tendencias actuales. Algunos también carecen de conjunto de características y metadatos. Los once criterios (Gharib et al.,2016) son [49]:

- Configuración completa de la red
- Tráfico completo
- Etiquetado Dataset
- Interacción completa
- Captura completa
- Protocolos disponibles
- Diversidad de Ataques
- Heterogeneidad
- Conjunto de atributos
- Metadatos

3.3. Configuración de los escenarios

Para el establecimiento de la data IDS2017, IDS2018 y del DDoS2019 por parte del CIC, los investigadores se basaron en tres arquitecturas de red las cuales

serán replicadas para la obteniendo de la data de validación del modelo detector una vez implementado. Se debe tener muy presente que tenemos 3 escenarios de configuración.

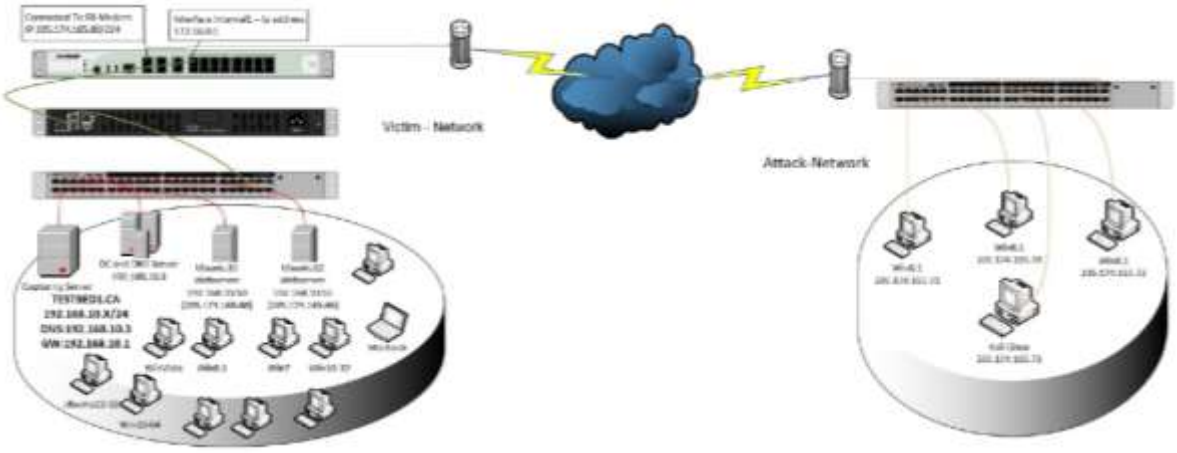


Figura 3-1 Arquitectura del banco de ensayos 2017 [49]

Iniciando con el **CIC-IDS2017** como muestra la Figura 3-1, la infraestructura del banco de pruebas se ha dividido en dos redes completamente separadas, a saber, la **Red-Víctima** y la **Red-Ataque**. A diferencia de los conjuntos de datos anteriores, en la Red-Víctima se ha cubierto todos los equipos comunes y necesarios, incluyendo el *router*, el cortafuego y el conmutador, junto con las diferentes versiones de los tres sistemas operativos comunes: Windows, Linux y Macintosh. La Figura 3-2 muestra la lista de servidores, estaciones de trabajo y cortafuegos, con los sistemas operativos instalados y las IPs públicas y privadas correspondientes. La **Red-Ataque** incluye un router, un *switch* y cuatro PCs, que tienen los sistemas operativos Kali y Windows 8.1. La **Red-Víctima** consta de tres servidores, un firewall, dos switches y diez PCs interconectados por un controlador de dominio (DC) y un directorio activo. Además, un puerto del conmutador principal

de la red víctima se ha configurado como puerto espejo y ha capturado completamente todo el tráfico de envío y recepción de la red

	Machine	OS	IPs
Victim-Network	Servers	Win Server 2016 (DC and DNS) Ubuntu 16 (Web Server) Ubuntu 12	192.168.10.3 192.168.10.50-205.174.165.68 192.168.10.51-205.174.165.66
	PCs	Ubuntu 14.4 (32, 64) Ubuntu 16.4 (32-64) Win 7Pro Win 8.1-64 Win Vista Win 10 (Pro 32-64) Mac	192.168.10.19-192168.10.17 192.168.10.16-192.168.10.12 192.168.10.9 192.168.10.5 192.168.10.8 192.168.10.14-192.168.10.15 192.168.10.25
	Firewall	Fortinet	
Attackers	PCs	Kali win 8.1 Win 8.1 Win 8.1	205.174.165.73 205.174.165.69 205.174.165.70 205.174.165.71

Figura 3-2 Red-Víctima y Red-Ataque: OS's y IP's [49]

De los 5 días de captura de data del CIC-IDS2017, solo se tiene en cuenta 2 días que es el interés respecto a ataques maliciosos DoS y DDoS. Día miércoles y viernes en la tarde, según se observa en Tabla 3-1 y Tabla 3-2

Wednesday, July 5, 2017	
DoS **slowloris** (9:47 – 10:10 a.m.)	
DoS **Slowhttptest** (10:14 – 10:35 a.m.)	
DoS **Hulk** (10:43 – 11 a.m.)	
DoS **GoldenEye** (11:10 – 11:23 a.m.)	
Attacker	**Kali**, 205.174.165.73
Victim	**WebServer Ubuntu**, 205.174.165.68 (Local IP192.168.10.50)
NAT Process on Firewall:	
Attack	205.174.165.73 ---> 205.174.165.80 (IP Valid Firewall) ---> 172.16.0.10 ---> 192.168.10.50
Reply	192.168.10.50 ---> 172.16.0.1 ---> 205.174.165.80 ---> 205.174.165.73

Tabla 3-1 Día miércoles conjunto CIC-IDS2017 [49]

Friday, July 7, 2017
Afternoon
DDoS **LOIC** (15:56 – 16:16)

Attackers	**Three Win 8.1,** 205.174.165.69 - 71
Victim	**Ubuntu16,** 205.174.165.68 (Local IP: 192.168.10.50)
NAT Process on Firewall:	
Attackers	205.174.165.69, 70, 71 -> 205.174.165.80 (IP Valid Firewall) -> 172.16.0.1
Reply	192.168.10.50 ---> 172.16.0.1 ---> 205.174.165.80 ---> 205.174.165.73

Tabla 3-2 Día viernes en la tarde conjunto CIC-IDS2017 [49]

Para la topología de red de **CSE-CIC2018** importante tener en cuenta que un perfil (perfiles B & Perfiles M) necesita una infraestructura para ser utilizado de manera efectiva. El banco de ensayos consistirá en algunas estaciones de trabajo interconectadas basadas en Windows y Linux. Para las máquinas con Windows, usaron diferentes paquetes de servicio (porque cada paquete tiene un conjunto diverso de vulnerabilidades conocidas) y para las máquinas con Linux usaron la distribución compatible con Metasploit, que está desarrollada para ser atacada por los nuevos probadores de penetración

La Figura 3-3 muestra la red implementada, que es una topología de red LAN común en la plataforma informática de AWS. Para tener una diversidad de máquinas similares a las redes del mundo real, instalaron 5 sub-redes, a saber, departamento de I + D (Dep1), departamento de gestión (Dep2), departamento de técnicos (Dep3), secretario y departamento de operaciones (Dep4), departamento de TI (Dep5) y salas de servidores. Para todos los departamentos, excepto el departamento de TI, instalaron conjuntos de diferentes sistemas operativos MS Windows (Windows 8.1 y Windows 10) y todas las computadoras del departamento de TI son Ubuntu. Para la sala de servidores, implementaron diferentes servidores MS Windows como 2012 y 2016 [50].

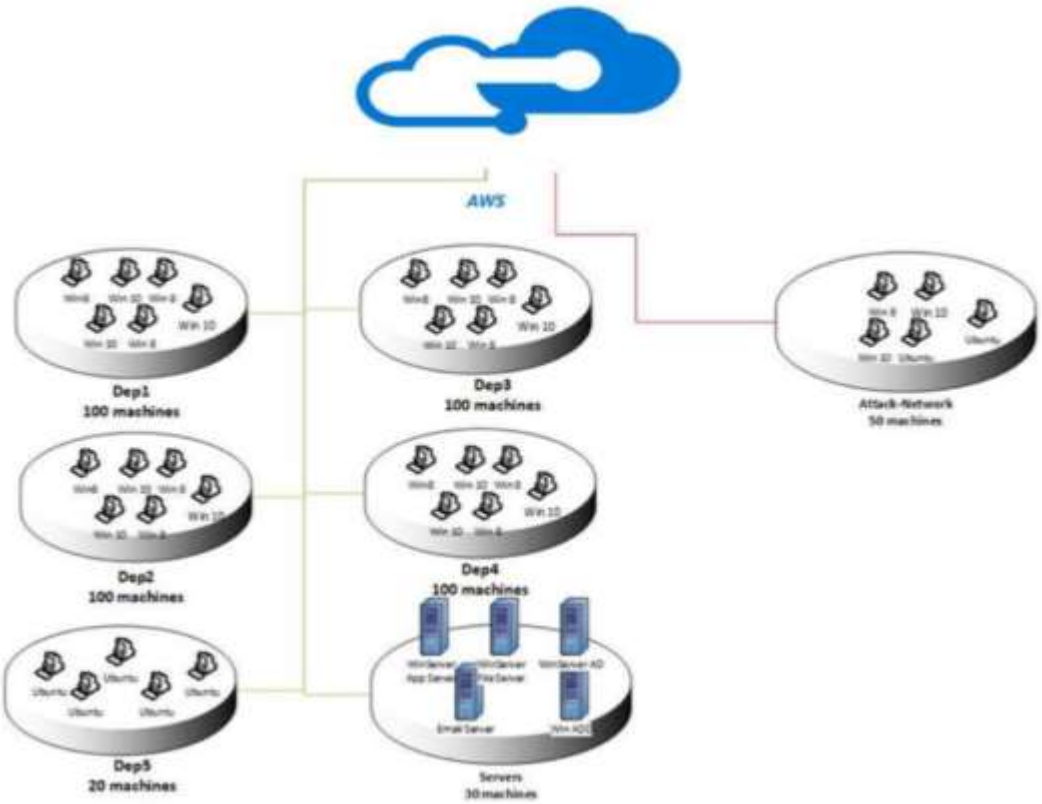


Figura 3-3 Topología de Red CIC-ISD2018 [50]

Los investigadores implementaron la infraestructura y ejecutamos los escenarios de ataque. En la Tabla 3-3 se muestra la lista de ataques que nos interesan DoS y DDoS, los IP de los atacantes y las víctimas relacionados, la fecha, la hora de inicio y de finalización de los ataques [50].

Por último, y del año más reciente, **CIC-DDoS2019** la generación de un tráfico de fondo realista fue la principal prioridad en la construcción de este conjunto de datos. Utilizaron el sistema propuesto de **Perfil-B** (Sharafaldin, et al. 2016) para perfilar el comportamiento abstracto de las interacciones humanas y generar un tráfico de fondo naturalista y benigno en el banco de ensayos propuesto Figura 3-3. Consta de dos redes completamente separadas a diferencia de los conjuntos de

datos anteriores, en la **Red-Víctima** emplearon todos los equipos necesarios y de uso común, incluidos el router, los cortafuegos y el conmutador, junto con las diferentes versiones de los sistemas operativos más utilizados [51].

Atacante	Víctima	Nombre del ataque	Fecha	Hora de inicio del ataque	Hora de finalización del ataque
172.31.70.46 (Valid IP:18.219.211.138)	18.217.21.148 - 172.31.69.25	**DoS-GoldenEye**	Thurs-15-02	9:26	10:09
172.31.70.8 (Valid IP:18.217.165.70)	18.217.21.148 - 172.31.69.25	**DoS-Slowloris**	Thurs-15-02	10:59	11:40
172.31.70.23 (Valid IP: 13.59.126.31)	18.217.21.148 - 172.31.69.25	**DoS-SlowHTTPTest**	Fri-16-02	10:12	11:08
172.31.70.16 (Valid IP:18.219.193.20)	18.217.21.148 - 172.31.69.25	**DoS-Hulk**	Fri-16-02	13:45	14:19
18.218.115.60 18.219.9.1 18.219.32.43 18.218.55.126 52.14.136.135 18.219.5.43 18.216.200.189 18.218.229.235 18.218.11.51 18.216.24.42	18.217.21.148 - 172.31.69.25	**DDoS attacks-LOIC-HTTP**	Tues-20-02	10:12	11:17
18.218.115.60 18.219.9.1 18.219.32.43 18.218.55.126 52.14.136.135 18.219.5.43 18.216.200.189 18.218.229.235 18.218.11.51 18.216.24.42	18.217.21.148 - 172.31.69.25	**DDoS-LOIC-UDP**	Tues-20-02	13:13	13:32
18.218.115.60 18.219.9.1 18.219.32.43 18.218.55.126 52.14.136.135 18.219.5.43 18.216.200.189 18.218.229.235 18.218.11.51 18.216.24.42	18.218.83.150 - 172.31.69.28	**DDOS-LOIC-UDP**	Wed-21-02	10:09	10:43
18.218.115.60 18.219.9.1 18.219.32.43 18.218.55.126 52.14.136.135 18.219.5.43 18.216.200.189 18.218.229.235 18.218.11.51 18.216.24.42	18.218.83.150 - 172.31.69.28	**DDOS-HOIC**	Wed-21-02	14:05	15:05

Tabla 3-3 Lista ataques DoS y DDoS diarios 2018, IPs máquinas, hora inicio y fin [50].

La Tabla 3-4 muestra la lista de servidores, cortafuegos y estaciones de trabajo, con sus sistemas operativos y las IPs públicas y privadas relacionadas en los días de entrenamiento y prueba. Un tercero ha ejecutado las familias de ataque en los días de entrenamiento y prueba (Red-Ataque). La Red-Víctima consta de un servidor (servidor web), un cortafuegos, dos conmutadores y cuatro PC. Además, un puerto del conmutador principal de la Red-Víctima se ha configurado como puerto espejo y captura completamente todo el tráfico de envío y recepción de la red [51].

El período de captura para el día de entrenamiento del 12 de enero comenzó a las 10:30 y terminó a las 17:15, y para el día de prueba del 11 de marzo comenzó a las 09:40 y terminó a las 17:35. Los ataques ejecutados posteriormente durante este período. Como muestra la Tabla 3-5, ejecutaron 12 ataques DDoS en el día de entrenamiento y 7 ataques en el día de pruebas [51].

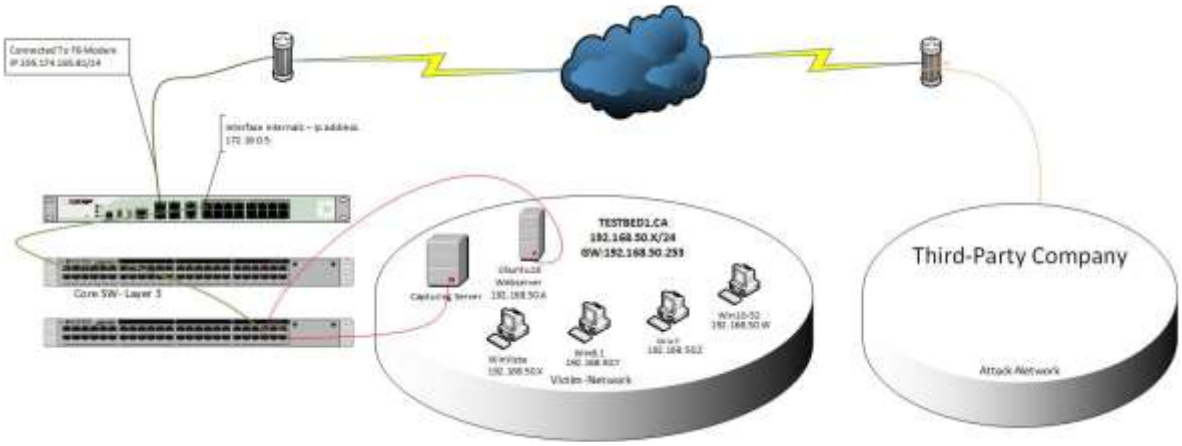


Figura 3-4 Arquitectura del banco de ensayos 2019 [51]

	Machine	OS	IPs
Victim Network	Server	Ubuntu 16.04 (Web Server)	192.168.50.1 (first day) 192.168.50.4 (second day)
	Firewall	Fortinet	205.174.165.81
	PCs (first day)	Win 7	192.168.50.8
		Win Vista	192.168.50.5
		Win 8.1	192.168.50.6
		Win 10	192.168.50.7
	PCs (second day)	Win 7	192.168.50.9
		Win Vista	192.168.50.6
		Win 8.1	192.168.50.7

	Win 10	192.168.50.8

Tabla 3-4 Sistemas operativos e IP de la red víctima [51]

Days	Attacks	Attack Time
First Day Testing Set	PortMap	9:43 - 9:51
	NetBIOS	10:00 - 10:09
	LDAP	10:21 - 10:30
	MSSQL	10:33 - 10:42
	UDP	10:53 - 11:03
	UDP-Lag	11:14 - 11:24
	SYN	11:28 - 17:35
Second Day Training Set	NTP	10:35 - 10:45
	DNS	10:52 - 11:05
	LDAP	11:22 - 11:32
	MSSQL	11:36 - 11:45
	NetBIOS	11:50 - 12:00
	SNMP	12:12 - 12:23
	SSDP	12:27 - 12:37
	UDP	12:45 - 13:09
	UDP-Lag	13:11 - 13:15
	WebDDoS	13:18 - 13:29
	SYN	13:29 - 13:34
	TFTP	13:35 – 17:15

Tabla 3-5 Etiqueta diaria del conjunto de datos 2019 [51]

3.4. Extracción de los atributos o características

CICFlowMeter es un generador de flujo de tráfico de red que ha sido escrito en Java y ofrece más flexibilidad en cuanto a la elección de los atributos que se desean calcular, la adición de nuevas, y un mejor control de la duración del tiempo de espera del flujo. Genera flujos bidireccionales (Biflow), en los que el primer paquete determina la dirección hacia adelante (fuente a destino) y hacia atrás (destino a fuente), de ahí que los 84 atributos estadísticos como la duración, el número de paquetes, el número de bytes, la longitud de los paquetes, etc. también se calculen por separado en la dirección hacia adelante y hacia atrás [50].

El resultado de la aplicación se presenta en formato de archivo CSV con seis columnas etiquetadas para cada flujo, a saber, FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort y Protocol con más de 80 atributos de tráfico de red. Normalmente, los flujos TCP terminan cuando se corta la conexión (por el paquete FIN) mientras que los flujos UDP terminan por un tiempo de espera del flujo. El valor

de *timeout* de flujo puede ser asignado arbitrariamente por el esquema individual, por ejemplo, 600 segundos tanto para el TCP como para el UDP. El CICFlowMeter-V3 puede extraer más de 80 atributos que se enumeran en el cuadro siguiente [50]:

	Nombre de Caracteristica	Descripción
1	**fl_dur**	Duración del flujo
2	**tot_fw_pk**	Total de paquetes en la dirección de avance
3	**tot_bw_pk**	El total de paquetes en dirección contraria
4	**tot_l_fw_pkt**	Tamaño total del paquete en dirección hacia adelante
5	**fw_pkt_l_max**	Tamaño máximo del paquete en dirección hacia adelante
6	**fw_pkt_l_min**	Tamaño mínimo del paquete en dirección hacia adelante
7	**fw_pkt_l_avg**	Tamaño medio del paquete en dirección hacia adelante
8	**fw_pkt_l_std**	El tamaño de la desviación estándar del paquete en dirección hacia adelante
9	**Bw_pkt_l_max**	El tamaño máximo del paquete en dirección hacia atrás
10	**Bw_pkt_l_min**	El tamaño mínimo del paquete en dirección hacia atrás
11	**Bw_pkt_l_avg**	El tamaño medio del paquete en dirección contraria
12	**Bw_pkt_l_std**	El tamaño de la desviación estándar del paquete en dirección hacia atrás
13	**fl_byt_s**	tasa de bytes de flujo que es el número de paquetes transferidos por segundo
14	**fl_pkt_s**	tasa de paquetes de flujo que es el número de paquetes transferidos por segundo
15	**fl_iat_avg**	Tiempo medio entre dos flujos
16	**fl_iat_std**	Desviación estándar tiempo dos flujos
17	**fl_iat_max**	Tiempo máximo entre dos flujos
18	**fl_iat_min**	Tiempo mínimo entre dos flujos
19	**fw_iat_tot**	El tiempo total entre dos paquetes enviados en la dirección de avance
20	**fw_iat_avg**	El tiempo medio entre dos paquetes enviados en la dirección de avance
21	**fw_iat_std**	El tiempo de desviación estándar entre dos paquetes enviados en la dirección de avance
22	**fw_iat_max**	El tiempo máximo entre dos paquetes enviados en la dirección de avance
23	**fw_iat_min**	El tiempo mínimo entre dos paquetes enviados en la dirección de avance
24	**bw_iat_tot**	El tiempo total entre dos paquetes enviados en dirección contraria
25	**bw_iat_avg**	El tiempo medio entre dos paquetes enviados en dirección contraria
26	**bw_iat_std**	El tiempo de desviación estándar entre dos paquetes enviados en dirección contraria

27	**bw_iat_max**	Tiempo máximo entre dos paquetes enviados en dirección contraria
28	**bw_iat_min**	El tiempo mínimo entre dos paquetes enviados en dirección contraria
29	**fw_psh_flag**	Número de veces que la bandera de PSH fue puesta en paquetes que viajan en dirección hacia adelante (0 para UDP)
30	**bw_psh_flag**	Número de veces que la bandera de PSH fue puesta en paquetes que viajan en dirección contraria (0 para UDP)
31	**fw_urg_flag**	Número de veces que la bandera de URG fue puesta en paquetes que viajan en la dirección de avance (0 para UDP)
32	**bw_urg_flag**	Número de veces que la bandera de la URG fue puesta en paquetes que viajan en dirección contraria (0 para UDP)
33	**fw_hdr_len**	Total de bytes utilizados para las cabeceras en la dirección de avance
34	**bw_hdr_len**	Total de bytes utilizados para las cabeceras en la dirección de avance
35	**fw_pkt_s**	Número de paquetes de avance por segundo
36	**bw_pkt_s**	Número de paquetes atrasados por segundo
37	**pkt_len_min**	Longitud mínima de un flujo
38	**pkt_len_max**	La longitud máxima de un flujo
39	**pkt_len_avg**	La longitud media de un flujo
40	**pkt_len_std**	La longitud de la desviación estándar de un flujo
41	**pkt_len_va**	Tiempo mínimo entre la llegada del paquete
42	**fin_cnt**	Número de paquetes con FIN
43	**syn_cnt**	Número de paquetes con SYN
44	**rst_cnt**	Número de paquetes con RST
45	**pst_cnt**	Número de paquetes con PUSH
46	**ack_cnt**	Número de paquetes con ACK
47	**urg_cnt**	Número de paquetes con URG
48	**cwe_cnt**	Número de paquetes con CWE
49	**ece_cnt**	Número de paquetes con ECE
50	**down_up_ratio**	Relación de carga y descarga
51	**pkt_size_avg**	Tamaño medio del paquete
52	**fw_seg_avg**	Tamaño medio observado en la dirección de avance
53	**bw_seg_avg**	El tamaño medio observado en la dirección de retroceso
54	**fw_byt_blk_avg**	Número medio de bytes de tasa de carga en la dirección de avance
55	**fw_pkt_blk_avg**	Promedio de paquetes a granel en la dirección de avance
56	**fw_blk_rate_avg**	Promedio de la tasa de carga en la dirección de avance
57	**bw_byt_blk_avg**	Número medio de bytes de tasa de carga en dirección inversa
58	**bw_pkt_blk_avg**	Promedio de la tasa de carga de los paquetes en dirección contraria
59	**bw_blk_rate_avg**	Promedio de la tasa de masa en la dirección inversa
60	**subfl_fw_pk**	El número promedio de paquetes en un subflujo en la dirección de avance
61	**subfl_fw_byt**	El número promedio de bytes en un subflujo en la dirección de avance

62	**subfl_bw_pkt**	El número promedio de paquetes en un subflujo en dirección contraria
63	**subfl_bw_byt**	El número promedio de bytes en un subflujo en la dirección de retroceso
64	**fw_win_byt**	Número de bytes enviados en la ventana inicial en la dirección de avance
65	**bw_win_byt**	num de bytes enviados en la ventana inicial en la dirección de retroceso
66	**Fw_act_pkt**	num de paquetes con al menos 1 byte de carga de datos TCP en la dirección de avance
67	**fw_seg_min**	Tamaño mínimo del segmento observado en la dirección de avance
68	**atv_avg**	El tiempo medio que un flujo estuvo activo antes de quedar inactivo
69	**atv_std**	El tiempo de desviación estándar que un flujo estuvo activo antes de volverse inactivo
70	**atv_max**	El tiempo máximo que un flujo estuvo activo antes de quedar inactivo
71	**atv_min**	El tiempo mínimo en que un flujo estuvo activo antes de quedar inactivo
72	**idl_avg**	El tiempo medio que un flujo estuvo inactivo antes de volverse activo
73	**idl_std**	El tiempo de desviación estándar que un flujo estuvo inactivo antes de volverse activo
74	**idl_max**	El tiempo máximo que un flujo estuvo inactivo antes de activarse
75	**idl_min**	El tiempo mínimo en que un flujo estuvo inactivo antes de activarse

Tabla 3-6 Extracción de Atributos o características del conjunto de datos [50]

3.5. Preprocesamiento de datos

La calidad de los datos y la cantidad de información útil que contienen, son factores clave que determinan lo bien que puede aprender un algoritmo de Aprendizaje Automático. Se sabe que no es habitual que los datos primarios, en crudo o en bruto se presenten en la forma necesaria para un rendimiento óptimo del algoritmo de aprendizaje, y en caso nuestro no fue la excepción. Cabe mencionar que, al utilizar 3 conjuntos de datos de un mismo autor, con muchos archivos individuales de variables separadas por comas (csv) pero en años distintos, presentaban algunas diferencias tanto en el número de columnas totales, cómo la escritura de los nombres de los atributos.

Dicha optimización de aprendizaje recae en primera medida por el bloque de preprocesamiento de los datos, que prepara, manipula (***data wrangling***), maneja de una manera correcta los datos vírgenes dados y capturados.

Este bloque de preprocesamiento fue dividido en 2 partes a saber: **Pre-pro I** y **Pre-pro II**, la diferencia básica entre las 2 es que en la primera parte siguen siendo muchos archivos .csv individuales, y en la segunda parte si es un único y total archivo .csv concatenado.

En **Pre-pro I** se elimina valores nulos (Null, NaN), valores faltantes o ausentes, constantes y valores infinitos como tipo cadena, problemas en el índice de columna o nombre y número de atributos; como que los nombres se repiten o se duplican como fila, eliminar un espacio en blanco que antecede al nombre de atributo, así como la diferencia de caracteres al escribir dichos atributos de un año a otro (2017-2019) y eliminar columnas que no existen en todos y son atributos raros (pero así están los datos desde su fuente), dejarlos todos por igual.

Se prosigue en **Pre-pro II** a darle formato legible a la data, realizando una limpieza completa, primero definiendo cada uno de los tipo de dato de los diferentes atributos, que se disponen en forma de columna; después pasando los archivos individuales que deseamos pasar (máximo acepto 8 ficheros .csv, para un total de 5,0GB y más de trece millones de registro de flujos de tráfico de red; deseábamos más pero el computador usado era insuficiente) por 3 funciones que básicamente hacen es elegir el valor promedio de la columna para ser sustituido por celdas que presenten tanto valores infinitos, como valores negativos, que están presentes en el conjunto de datos y agregan ruido; para así finalmente concatenar dichos archivos individuales todos con atributos numéricos tomados como tipo objeto, para darle el tipo de dato numérico a sus atributos (**`uint8, uint16, uint32, uint64, int64, float32, float64`**), que se basa en nuestro marco de datos total (dataframe), con unas dimensiones de 82 entre 79 atributos y 2 etiquetas extra

creadas (columnas) con 13’596.280 flujos de tráfico de red(filas). Y finaliza el bloque del procesamiento en la II segunda parte, separando la matriz en: 79 atributos predictores (X) y las 3 etiquetas de clase/familia de Ataque Malicioso (y), obteniendo así, 2 archivos de comas separadas por variables.

Cabe mencionar que hasta este punto solo se ha manejado conceptos básicos de Python con matrices aplicando Pandas y NumPy, pero hasta este momento no se ha tocado el ámbito o temática del AA Machine Learning, ha sido solo manejo de arreglos en Python.

3.6. Normalización de datos

Muchos de los algoritmos de ML requieren algún tipo de escalado de atributos para un rendimiento óptimo como por ejemplo los algoritmos de tipo lineal: regresión logística, descenso del gradiente estocástico, Ridge, entre otros. La normalización (*sklearn.preprocessing.MinMaxScaler*) es un método de escalado de atributos que proporciona a nuestros datos la propiedad de una distribución normal estándar, la normalización cambia la media de cada atributo para que se centre en cero y para que cada atributo tenga una desviación estándar de 1. En palabras sencillas, es ajustar los atributos (datos) a la misma escala.

La estandarización (*sklearn.preprocessing.StandardScaler*) puede ser más práctica para los algoritmos de AA, centra las columnas de atributos a una media con una desviación estándar de 1, así tiene una distribución normal, lo que hace más fácil aprender los pesos.

Resulta muy interesante puntualizar que tanto en teoría como en la práctica experimental se pudo confirmar: los arboles de decisión y los bosques aleatorios son dos de los poquísimos algoritmos de AA en los cuales **no** debemos preocuparnos por el escalado de atributos. Estos algoritmos son invariables [52-

pag142] en cuanto al escalado en resultados en Python, se observa que, al hacerlo, los resultados son inferiores en rendimiento y desempeño cuando efectivamente se procede a normalizar.

3.7. Procesamiento

Aplicar reducción de dimensionalidad. Siendo el paso más importante de la temática de AA y la ciencia de datos en cualquiera de sus aplicaciones, se debe tener muy en cuenta el análisis de esta etapa, que tomo la mayor parte de tiempo de este proyecto.

Gracias que la extracción de los 84 atributos ya está dada, por el completo conjunto de datos de la CIC de la UNB y todo su previo estudio con sus análisis, que es lo más difícil dentro del proceso de la ingeniería de atributos (*Feature Engineering*). Se debe realizar es la reducción de dimensionalidad con la selección de atributos, y a la división del *dataframe* de Pandas en dos (2), conjuntos de datos de entrenamiento y conjunto de datos de prueba.

Algunos de los atributos dados pueden estar altamente relacionados y, por tanto, llegan a ser redundantes. En estos casos, las técnicas de selección de atributos significativos son muy útiles para comprimir los atributos en un subespacio dimensional más pequeño. Reducir la dimensionalidad tiene la ventaja de que requiere menos espacio de almacenamiento y el algoritmo de aprendizaje funciona mucho más rápido, además lo más importante se pudo comprobar con resultados a lo largo del proyecto investigativo, mejora el rendimiento predictivo de nuestro modelo. Reducir la complejidad del modelo y evitar el sobreajuste, que se presenta como una alta varianza en los datos nuevos. Se prosigue así, a reducir la dimensionalidad al aplicar la selección de atributos (*Feature Selection*), que para nuestro diseño propuesto se ha realizado mediante 2 pasos encadenados, uno después del otro, aplicando dos métodos diferentes: **Desviación estándar** en el primero, y **correlaciones de Spearman** en el segundo paso. Para obtener un

procesamiento bien elaborado y pasar de 80 atributos, a solo 37 atributos que concentran toda la información original, pero de una forma más óptima mejorando el desempeño del algoritmo de aprendizaje; aquí, se presenta el primer entregable del segundo objetivo, al analizar la gráfica de mapa de calor, de solo una parte del total del conjunto de datos, pues toda por volcado y reinicio de mi RAM, era imposible hacerlo.

La selección de atributos que realiza una compresión de los datos es un tema importante en el AA que nos ayudara a almacenar y analizar las crecientes cantidades de datos que se producen y recogen en la era moderna de la tecnología [52].

Ahora para finalizar, se completa una de las partes más básicas del AA. Para determinar si nuestro algoritmo de aprendizaje automático no solo funciona bien en el conjunto de entrenamiento, sino que también se generaliza en datos nuevos, es allí donde nace la filosofía de dividir de forma aleatoria, y en nuestro caso estratificado (muestras objetivos por igual), el conjunto de datos en un conjunto de prueba y de entrenamiento individuales. Utilizaremos el conjunto de entrenamiento para entrenar y optimizar nuestro modelo de aprendizaje automático, al tiempo que mantenemos el conjunto de prueba hasta el final para evaluar el modelo final.

3.8. Resultados experimentales

En este apartado se trata sobre la etapa de Entrenamiento y la selección de 5 modelos predictivos de clasificación multiclase, que se podrían llamar, algoritmos de pódium.

En las etapas anteriores, se ha dado forma a los datos de flujos de tráfico de red antes de proporcionárselos a los algoritmos de aprendizaje [52]. Se han desarrollado diferentes tipos de algoritmos de aprendizaje automático para resolver

distintas tareas problemáticas, ahora bien, cada algoritmo de clasificación tiene sus sesgos inherentes, y ninguna clasificación individual es superior si no hacemos suposiciones sobre la tarea. En la práctica, resulta esencial comparar como mínimo un puñado de algoritmos distintos para entrenar y seleccionar el mejor modelo de rendimiento [52]. Es así como en nuestro diseño de modelo, se propusieron más de 30 algoritmos de aprendizaje diferentes, pero llegando solo a obtener resultados de más 20 de ellos, los demás algoritmos su tiempo de ejecución no finiquita en un tiempo de hasta 25 horas, era imposible por el costo computacional, tal era el caso de los algoritmos de SVM y la mayoría de KNN, y algunos lineales.

Ahora para poder comparar los diferentes modelos, debemos decidir las unidades de para medir el rendimiento; en este punto es también de vital importancia, pues aunque la precisión es con frecuencia la unidad de medida para la clasificación, en nuestro caso puntual de diseño, decidimos por el ***Recall***, traducido como sensibilidad o exhaustividad, ya que es nuestro propósito de detección necesitamos que las trazas benignas sean clasificadas correctamente como benigna, así intentamos responder a la siguiente pregunta: ¿Qué proporción de positivos reales se identificó correctamente?, así se busca lo más cercano a la idealización que sea o tienda a uno (1) y la que resume todo es completamente la Matriz de confusión multiclase, donde se percibe el análisis interno que realiza cada algoritmo en su aprendizaje especifico y propio en cada uno. Con estas comparaciones dichas, se eligen hasta 5 algoritmos de aprendizaje para así realizar la selección del modelo predictor definitivo, y descartar los demás algoritmos de aprendizaje, ya que por múltiples razones de tiempo y de resultados de rendimiento, no los hace competentes. Ahora, se vuelve a realizar el entrenamiento, pero ahora más detallado y haciendo uso de unos de las técnicas de validación cruzada, para nuestro caso el mejor método que implementa Scikit-learn, el ***cross validate***, ya que permite hacer uso de múltiples métricas y no una única, además de otorgar tiempos, sin necesidad de implementarlo. Dicha validación cruzada funciona al

tomarse el conjunto de datos de entrenamiento y dividirse en 10 subconjuntos de validación y entrenamiento para estimar el rendimiento de generalización del modelo.

En este punto, concretando nuestra etapa de aprendizaje y entrenamiento, habiendo ya hecho la medición del rendimiento, después la selección del modelo, seguido de una validación cruzada, se ha tratado de realizar la optimización de hiperparámetros pero fue imposible en tiempos computacionales para que convergiera en un resultado, y no se sabía cuánto iba a demorar (se dejó hasta 10 horas una vez) ni con el método RandomizedSearchCV(), ni con el método GridSearchCV; cabe mencionar que es por el tamaño tan grande del conjunto de datos, se debió tomar el 5% de 13 millones, para tener resultados que después se compararon con el conjunto completo, y no aplicaban, eran peores, en vez de mejorar.

Por último, se hace muchas gráficas para el análisis con el uso de los atributos funcionales del método del algoritmo que implementa la librería Scikit-learn, como la importancia de los atributos, si son arboles unitarios el diagrama de árbol.

Es vital importancia indicar que se llevará a cabo el uso matemático, estadístico y probabilístico de los diferentes algoritmos de aprendizaje automático, gracias a la poderosa herramienta de la librería de Scikit-learn, que proporciona una interfaz intuitiva para usar estos algoritmos de forma eficiente y productiva.

La elección de un algoritmo de clasifican apropiado para una tarea problemática concreta requiere practica; cada algoritmo posee sus propias peculiaridades y está basado en determinadas suposiciones [52]. Eventualmente, el rendimiento de un clasificador, tanto de rendimiento computacional como de poder predictivo, depende sobre todo de los datos de entrada disponibles para el aprendizaje [52].

Por último, finaliza esta etapa de entrenamiento, que es de mayor coste computacional al solicitar tantos requerimientos de máquina, y en el largo tiempo que demora en su procesamiento lógico interno, aplicando un script importando la librería Pickles para guardar el modelo, en un archivo de extensión binaria. bien, y poder ser cargado posteriormente.

3.9. Evaluación

Después de haber seleccionado un modelo instalado en el conjunto de datos de entrenamiento, podemos utilizar el conjunto de datos de prueba para estimar como funciona con los datos no vistos para estimar como funciona con los datos no vistos y así estimar el error de generalización. Es así, como nuestro conjunto de prueba es el 10% del conjunto total, con **1'359.628 flujos** de tráfico de red(filas).

Además del anterior conjunto de prueba que obviamente contiene etiquetas, se ha utilizado uno adicional, doble evaluación, creado gracias a la gran disponibilidad de trafico malicioso de denegación, que se presenta en el conjunto total de CIC-2019, y se ha conformado con los mismos ataques que se ha entrenado, pero con otros flujos diferentes a su etapa de aprendizaje, lo que harán de 2 conjuntos de prueba etiquetados no vistos, y obtener una doble evaluación de resultados para respaldas la decisión de elección del mejor rendimiento, desempeño y resultados de los 5 algoritmos de aprendizaje que había elegido como pódium.

Los resultados se presentan en matriz de confusión multiclase; del reporte de clasificación; de todas las métricas de precisión, exactitud, exhaustividad, valor-F tanto micro como macro; en pérdidas como ***Log Loss*** (pérdida logística o pérdida de entropía cruzada), Hamming; también en representaciones gráficas como la

curva ROC que compara Tasa de Verdaderos Positivos Vs Tasa de Falsos Positivos, curva Precisión Vs Recall.

En llegado caso que el rendimiento de evaluación de las pruebas nos satisface, ya podemos utilizar este modelo detector y clasificador para predecir nuevos y futuros datos de flujos de tráfico de red, poder detectar ataques maliciosos de los benignos, y en los maliciosos clasificar en sus clases o familias de ataques de denegación DoS y DDoS.

Es importante observar que los parámetros para los procedimientos hechos anteriormente como preprocesamiento y procesamiento, y que los mismos parámetros vuelven a aplicarse más tarde para transformar cualquier nueva muestra de datos [52]. de trafico de red en paquetes capturados (pcap).

3.10. Efectividad computacional

Se ha deseado hacer este aparatado en el capítulo, por la necesidad de expresar que fue muy difícil implementar muchos procedimientos de código de lenguaje Python por el consumo de recursos computacionales tanto de memoria como de tiempo. La eficiencia de muchas de las clases de métodos de la librería de Scikit-learn como se han mencionado al largo del presente capitulo, fue imposible obtener un resultado, porque sencillamente la memoria RAM llegaba a su límite, volcaba el proceso, y el núcleo de la RAM se reiniciaba. Tal fue el caso en muchos algoritmos de aprendizaje como Máquinas de Vectores de Soporte SMV, los K-vecinos más cercanos KNN; en la optimización de los hiperparámetros por diferentes formas; y en la adopción de un conjunto de datos más grande, que supere los 5,0GB, pues después de ese tamaño, es imposible siquiera leerlo, captarlo. En estos casos implementación práctica en tiempo y costo, fue imposible.

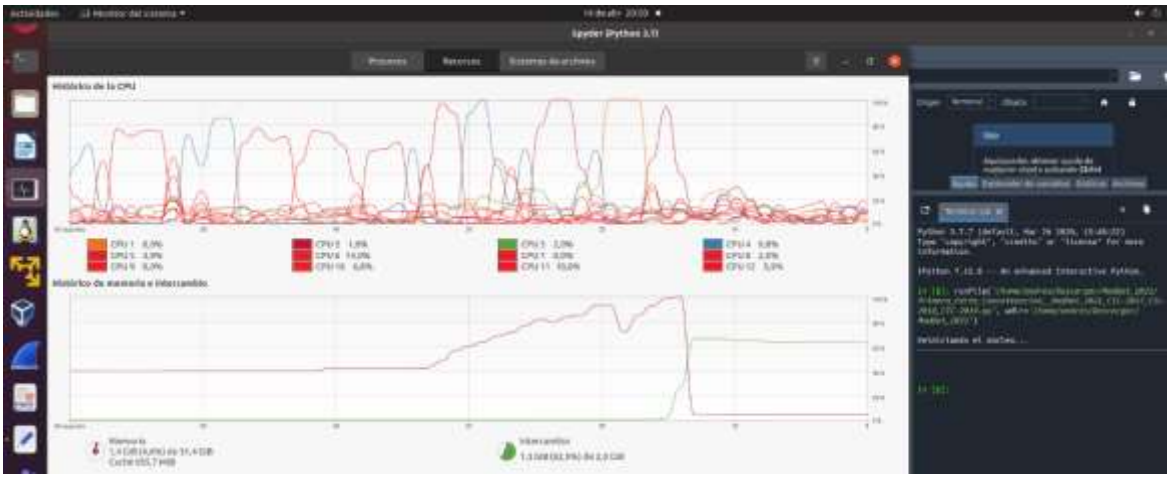


Figura 3-6 Reiniciando núcleo del kernel de la Memorira RAM [Elaboración propia]

Figura 3-7 ejemplo de un trabajo representativo llevado al límite [Elaboración propia]

Figura 3-5 ejemplo del uso excesivo y requerimiento de Memoria RAM [Elaboración propia]

3.11. Significación estadística

Si bien, todas las anteriores métricas y resultados que se presentan del apartado anterior de evaluación, y en nuestro caso hasta doble, son gracias al poder de las etiquetas, y el poder que encarna el aprendizaje supervisado que de hecho es muy tedioso obtener en las bases de datos públicas y disponibles, por su arduo

trabajo de tiempo que lleva hacer dichas etiquetas de clases. Ahora, esa es la misma idea con la que se ha diseñado nuestro **modelo DetD&DoS** propuesto, la de mediante el poder de la máquina de aprender de métodos matemáticos de ensamble de árboles y bosques, después dar y brindar una respuesta a la pregunta de clasificación multiclase a la cual fue enseñada. Así, los resultados de dicha clasificación de múltiples 10 clases de ataques maliciosos de denegación [codifica 1-10], incluido el tráfico benigno [codifica 0], en su método de ***.predict***, brindar la solución a esta toma de decisión de etiqueta de clase de ataques DoS y DDoS. En resumen, todo se ha realizado para una única columna, decisión tomada por técnicas de aprendizaje automático, que lleva el mundo estadístico analítico a un nivel superior. Así con los resultados únicos de decisión de la predicción del Modelo Detector/Clasificador de las clases DoS, DDoS y Benigno, se hace un conteo de sus valores, para plasmarlos en gráficos de barras para ver la cantidad de flujos y en gráficos de torta para ver su cantidad porcentual.

Ahora, gracias al poder de la predicción de probabilidades de modelo a mención, se realiza otras gráficas para representar la matriz de resultados probabilísticos de 11 x 11, para hacer tanto un gráfico de líneas, como un gráfico de dispersión por cada clase/familia individual para ver la probabilidad de cada clase.

Por último, en el siguiente diagrama de la Figura 3-8 presentamos una hoja de ruta, diagrama de flujo de la metodología propuesta de diseño e implementación llevado a cabo.

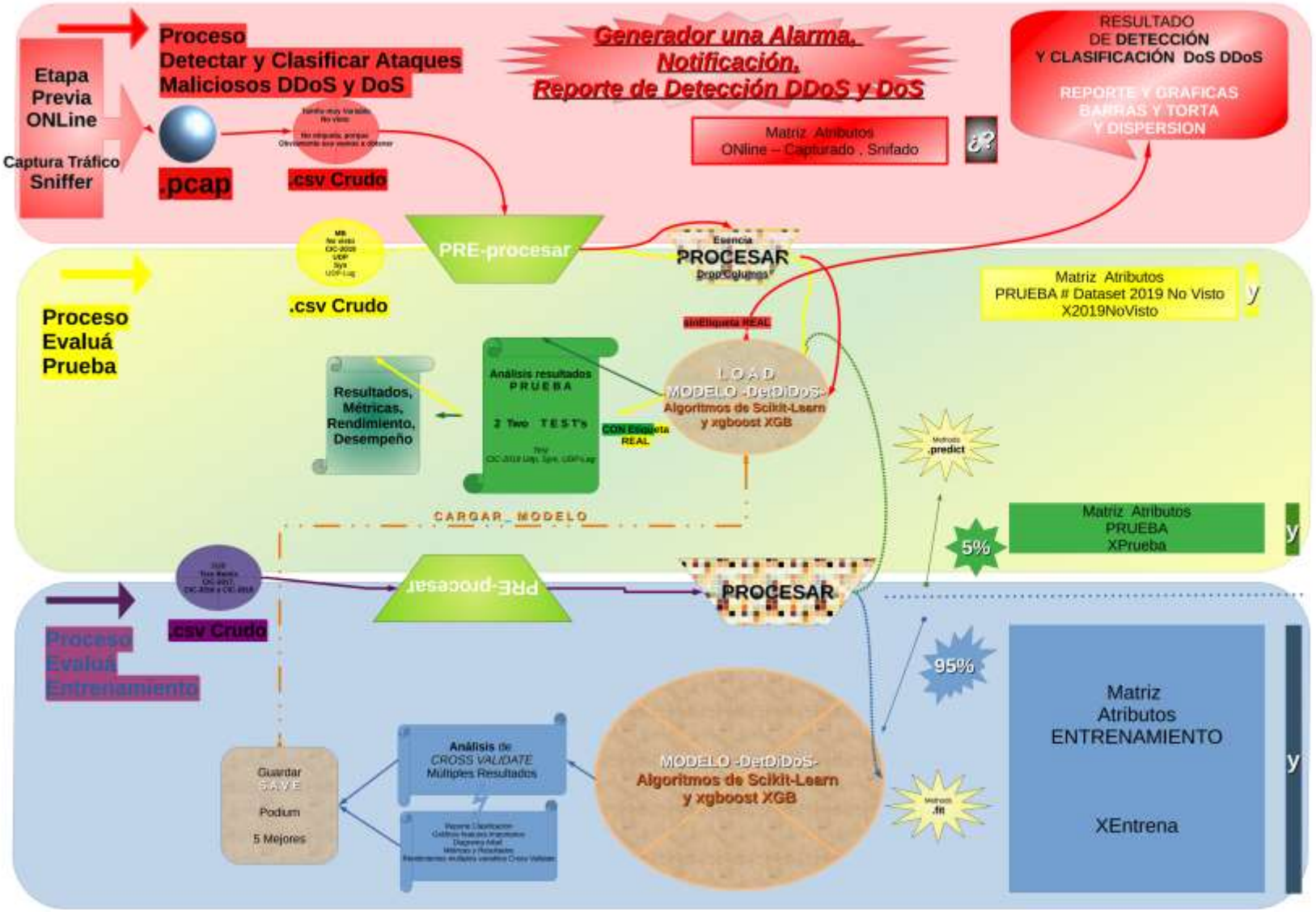


Figura 3-8 Hoja de ruta, Diagrama de flujo de la Metodología propuesta [Elaboración propia]

Capítulo 4

DESARROLLO DEL MODELO DETECTOR Y CLASIFICADOR –DetD&DoS-

4.1. Obtención del conjunto de datos total

Un conjunto de datos (del anglicismo *dataset*) es una colección de datos e información habitualmente tabulada, se compone de elementos separados pero que puede ser manipulado como una unidad por una computadora.

En general un conjunto de datos corresponde a los contenidos de una única tabla de base de datos o una única matriz de datos estadísticos, donde cada columna de la tabla representa una variable en particular o atributo, característica (***feature***) y cada fila representa a un miembro determinado del conjunto de datos, un flujo de tráfico de red para nuestro caso.

4.1.1 Licencia del conjunto de datos CSE-CIC-IDS2018

Se Puede redistribuir, republicar y reflejar el conjunto de datos del CSE-CIC-IDS2018 en cualquier forma. Sin embargo, cualquier uso o redistribución de los datos debe incluir una cita del conjunto de datos del CSE-CIC-IDS2018 y un enlace a la página en AWS.

El Documento de investigación que describe los detalles del análisis del conjunto de datos similares de IDS/IPS y los principios relacionados [53]

4.1.2. Licencia conjunto de datos CICDDoS2019

Se Puede redistribuir, republicar y reflejar el conjunto de datos de CICDDoS2019 en cualquier forma. Sin embargo, cualquier uso o redistribución de los datos debe incluir una cita del conjunto de datos CICDDoS2019 y del documento publicado correspondiente. Un documento de investigación en el que se esbocen los detalles del análisis del conjunto de datos similares de IDS/IPS y los principios relacionados [54]

4.1.3. Licencia CIC-IDS2017

El conjunto de datos del CICIDS2017 consiste en flujos de red etiquetados, incluidos paquetes completos de carga útil en formato pcap, los perfiles correspondientes y los flujos etiquetados (GeneratedLabelledFlows.zip) y los archivos CSV para fines de aprendizaje automático y profundo (MachineLearningCSV.zip) están a disposición del público para los investigadores. Se ha utilizado el conjunto de datos, citando el documento relacionado, en el que se esbozan los detalles del conjunto de datos y sus principios subyacentes [55].

4.1.4. Unión especifica de Conjunto de Datos CIC-IDS2017, CSE-CIC-IDS2018 & CIC-DDoS2019 para ataques DoS y DDoS

Se ha Conformado nuestro conjunto de datos definitivo, enfatizando y especializándonos en los ataques maliciosos de Denegación de Servicios DoS y los ataque Distribuido Denegación de Servicio DDoS: combinamos, unimos, enlazamos y mezclamos nuestro Conjunto de Datos CIC-IDS2017, CSE-CIC-IDS2018 & CIC-DDoS2019, para un total de 8 archivos .CSV, con un Tamaño de 5,0 GB, y cuyos archivos originales fueron modificados por cuestiones de capacidad de la Memoria RAM de 32GB DDR4. Así dicho conjunto de datos, nuestro propuesto, se conforma:

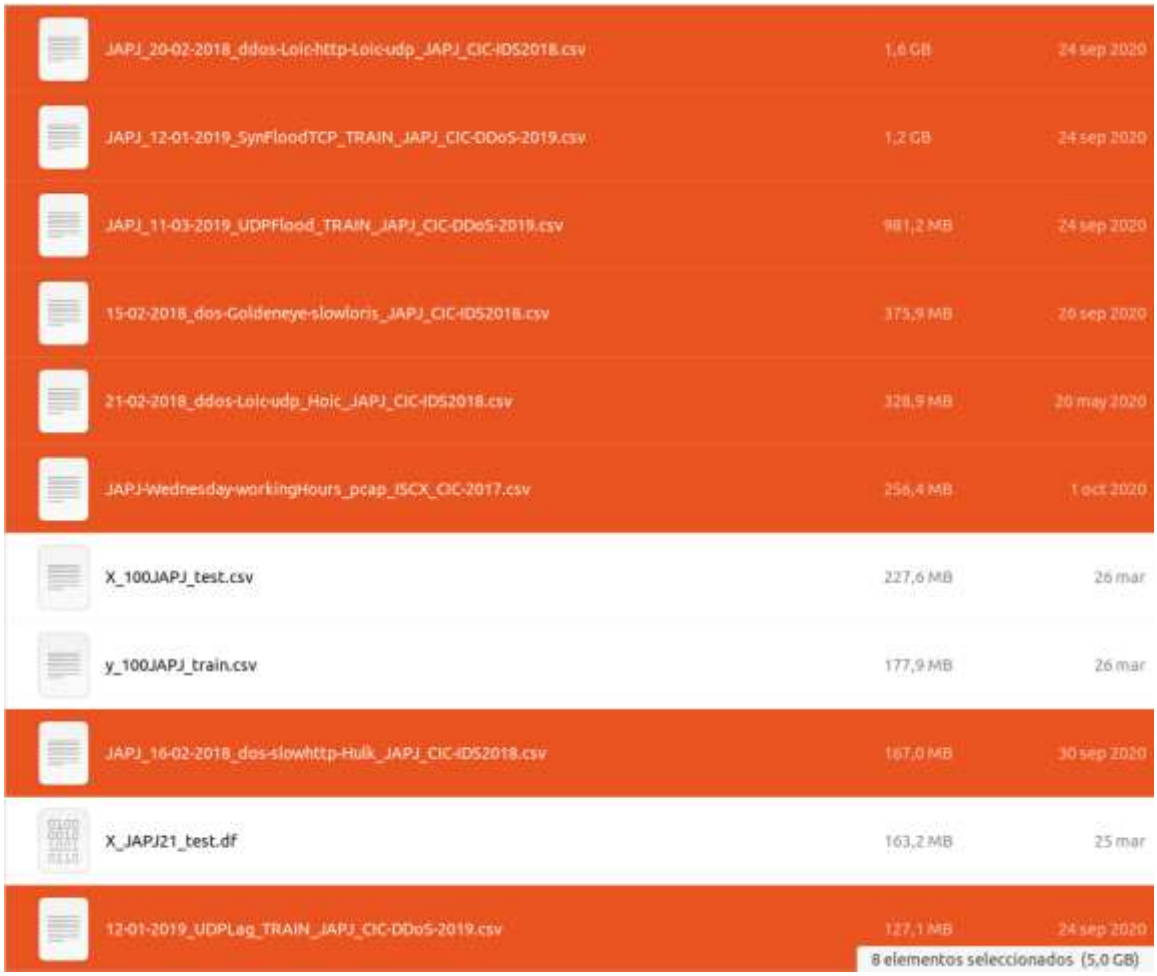


Figura 4-1 Combinación especifica de Conjunto de Datos CIC [elaboración propia]

4.2. Cargar los Módulos y Librerías de Python

A la hora de programar, casi siempre, es necesario recurrir a librerías, módulos, clases, paquetes, entre otros, para facilitarnos el desarrollo de un programa sin tener que repetir código o inventar la rueda nuevamente. Así podemos reconocer una librería como un conjunto de módulos cuya agrupación tiene una finalidad específica y que también puede ser invocada, tal como un módulo. Pero no es un módulo, sino un conjunto de ellos con una estructura determinada para lograr una finalidad.

Después de instalar y actualizar los módulos de Python se procede a usar cualquier archivo fuente de Python como módulo ejecutando una declaración de importación en algún otro archivo fuente de Python. Cuando el intérprete encuentra una declaración de importación, importa el módulo. Así, todos las librerías y módulos importados de Python usados en Detector **DetD&DoS** son explicadas en detalle:

4.2.1. NumPy

```
import numpy as np
```

NumPy es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos. Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.

Figura 4-2 Librería NumPy [Logo Corporativo]

4.2.2. Pandas

```
import pandas as pd
```

Pandas es una librería de Python especializada en el manejo y análisis de estructuras de datos.

Las principales características de esta librería son:

Define nuevas estructuras de datos basadas en los arrays de la librería NumPy pero con nuevas funcionalidades.
Permite leer y escribir fácilmente ficheros en formato CSV, Excel y bases de datos SQL.
Permite acceder a los datos mediante índices o nombres para filas y columnas.
Ofrece métodos para reordenar, dividir y combinar conjuntos de datos.
Permite trabajar con series temporales.
Realiza todas estas operaciones de manera muy eficiente.

Tipos de datos de Pandas

Pandas dispone de tres estructuras de datos diferentes:

Series: Estructura de una dimensión.
DataFrame: Estructura de dos dimensiones (tablas).
Panel: Estructura de tres dimensiones (cubos).

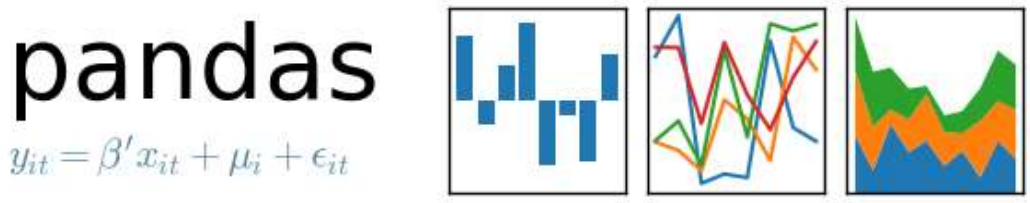


Figura 4-3 Librería Pandas [Logo Corporativo]

4.2.3. Matplotlib

```
import matplotlib.pyplot as plt
```

Matplotlib es una librería de Python especializada en la creación de gráficos en dos dimensiones.

Permite crear y personalizar los tipos de gráficos más comunes, entre ellos:

Diagramas de barras
Histograma
Diagramas de sectores
Diagramas de caja y bigotes
Diagramas de violín
Diagramas de dispersión o puntos
Diagramas de líneas
Diagramas de áreas
Diagramas de contorno
Mapas de color

Figura 4-4 Librería Matplotlib [Logo Corporativo]

4.2.4. Seaborn

```
import seaborn as sns
```

Seaborn se trata de una biblioteca de visualización de datos de Python basada en Matplotlib, que proporciona una interfaz de alto nivel para construir gráficos estadísticos. Así como también está integrada con la biblioteca de Pandas para facilitar la estructura de datos. Tiene como finalidad la visualización gráfica de los datos de forma más atractiva y descriptiva con el objetivo de facilitar la comprensión.

Figura 4-5 Librería Seaborn [Logo Corporativo]

4.2.5. Glob

```
import glob
```

Usa las reglas de shell de Unix para buscar nombres de archivos que coincidan con un patrón.

Aunque la interfaz de glob es pequeña, el módulo incluye una gran cantidad de poder. Es útil en cualquier situación donde un programa necesita buscar una lista de archivos en el sistema de archivos con nombres que coinciden con un patrón. Para crear una lista de nombres de archivos que tengan una cierta extensión, prefijo, o cualquier cadena común en el medio, usa glob en lugar de escribir código personalizado para escanear los contenidos del directorio.

4.2.6. Scikit-learn

Es la Librería de mayor uso, y la cual se realiza todas las técnicas de ML: en todas sus etapas de procesamiento, aprendizaje y prueba evaluando con métricas de rendimiento y desempeño.

4.2.6.1. train_test_split

```
from sklearn.model_selection import train_test_split
```

Entre las herramientas para la selección de modelos de Scikit-learn nos podemos encontrar con la función train_test_split. Una función que nos permite dividir un conjunto de datos en uno de entrenamiento y otro de prueba. En la bibliografía es habitual encontrar que se tiene que dividir los conjuntos de datos para el entrenamiento de los modelos en tres: entrenamiento, validación y prueba, no en dos. Actualmente no existe una función que haga esto de forma automática en Scikit-learn. El método train_test_split de Scikit-learn nos permite fácilmente dividir un conjunto de datos de una matriz o DataFrame en dos aleatorios con un tamaño dato. No existe una forma directa de dividir el conjunto de datos en tres. Pero es algo que se puede hacer fácilmente anidando los valores.

4.2.6.2. confusion_matrix

```
from sklearn.metrics import confusion_matrix
```

Calcular la matriz de confusión para evaluar la exactitud de una clasificación.

Por definición, una matriz de confusión es tal que es igual al número de observaciones que se sabe que están en el grupo y que se prevé que estén en el grupo. Por lo tanto, en la clasificación binaria, el recuento de los verdaderos negativos es C0,0, los falsos negativos es C1,0, los verdaderos positivos es C1,1 y los falsos positivos es C0,1.

Figura 4-6 Librería Scikit-learn [Logo Corporativo]

4.2.6.3. DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
```

Los árboles de decisión (DT) son un método de aprendizaje supervisado no paramétrico utilizado para la clasificación y la regresión. El objetivo es crear un modelo que prediga el valor de una variable objetivo mediante el aprendizaje de reglas de decisión simples inferidas a partir de las características de los datos. Cuanto más profundo es el árbol, más complejas son las reglas de decisión y más ajustado es el modelo.

El Clasificador de Árbol de Decisión es una clase capaz de realizar una clasificación multiclase en un conjunto de datos. Como con otros clasificadores, el DecisionTreeClassifier toma como entrada dos matrices: una matriz X, dispersa o densa, de tamaño [n_muestras, n_características] que contiene las muestras de formación, y una matriz Y de valores enteros, de tamaño [n_muestras], que contiene las etiquetas de clase para las muestras de formación.

4.2.6.4. ExtraTreeClassifier

```
from sklearn.tree import ExtraTreeClassifier
```

Un clasificador de árbol extremadamente aleatorio. Los extra-árboles difieren de los árboles de decisión clásicos en la forma en que se construyen. Cuando se busca la mejor división para separar las muestras de un nodo en dos grupos, se extraen divisiones aleatorias para cada una de las características seleccionadas al azar de max_features y se elige la mejor división entre ellas. Cuando max_características se establece en 1, esto equivale a construir un árbol de decisión totalmente aleatorio.

4.2.6.5. RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
```

En los bosques al azar, cada árbol del conjunto se construye a partir de una muestra extraída con reemplazo (es decir, una muestra de *bootstrap*) del conjunto de entrenamiento. Además, al dividir cada nodo durante la construcción de un árbol, la mejor división se encuentra entre todas las características de entrada o un subconjunto aleatorio de tamaño max_features.

El propósito de estas dos fuentes de aleatoriedad es disminuir la varianza del estimador forestal. De hecho, los árboles de decisión individual suelen mostrar una alta varianza y tienden a sobreajustarse. La aleatoriedad inyectada en los bosques da como resultado árboles de decisión con errores de predicción algo desacoplados. Tomando un promedio de esas predicciones, algunos errores pueden anularse. Los bosques aleatorios logran una varianza reducida al combinar diversos árboles, a veces a costa de un ligero aumento del sesgo. En la práctica, la reducción de la varianza suele ser significativa, lo que permite obtener un modelo general mejor.

Un bosque aleatorio es un metaestimador que se ajusta a una serie de clasificadores de árboles de decisión en varias submuestras del conjunto de datos

y utiliza el promediado para mejorar la precisión de la predicción y controlar el sobreajuste. El tamaño de la submuestra se controla con el parámetro max_samples si bootstrap=True (por defecto), de lo contrario se utiliza todo el conjunto de datos para construir cada árbol.

4.2.6.6. ExtraTreesClassifier

```
from sklearn.ensemble import ExtraTreesClassifier
```

Un clasificador de árboles adicionales o suplementarios. Esta clase implementa un metaestimador que ajusta un número de extra-árboles de decisión aleatorios en varias sub-muestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobre-ajuste.

4.2.6.7. AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
```

Un clasificador AdaBoost (Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.) es un meta-estimador que comienza ajustando un clasificador en el conjunto de datos original y luego ajusta copias adicionales del clasificador en el mismo conjunto de datos, pero donde los pesos de las instancias clasificadas incorrectamente se ajustan de manera que los clasificadores posteriores se centran más en los casos difíciles. Esta clase implementa el algoritmo conocido como **AdaBoost-SAMME** (Zhu, H. Zou, S. Rosset, T. Hastie, “Multi-class AdaBoost”, 2009.)

4.2.6.8. GradientBoostingClassifier

```
from sklearn.ensemble import GradientBoostingClassifier
```

Gradient Boosting para la clasificación. El GB construye un modelo aditivo de forma progresiva por etapas; permite optimizar funciones de pérdida diferenciables arbitrarias. En cada etapa se ajustan árboles de regresión de n_clases_ sobre el gradiente negativo de la función de pérdida de desviación binomial o multinomial.

La clasificación binaria es un caso especial en el que sólo se induce un único árbol de regresión.

4.2.7. SciPy

4.2.7.1. Spearmanr de SciPy

```
from scipy.stats import spearmanr
```

Primero, la correlación se define como una técnica estadística que puede mostrar si los pares de variables están relacionados o son interdependientes y con qué intensidad. Cuando miramos dos variables a lo largo del tiempo, si una variable cambia, ¿cómo afecta esto al cambio en otra variable?

Las Medidas de correlación que diferentes variables son interdependientes. Las variables dentro de un conjunto de datos pueden estar relacionados por muchas razones. Una variable podría causar o depender de los valores de otra variable. Una variable podría asociarse ligeramente con otra variable. Dos variables podrían depender de una tercera variable desconocida. La correlación larga y corta es la siguiente: La correlación es un número entre –1,0 y +1,0.

Una correlación puede ser positiva, lo que significa que ambas variables se mueven en la misma dirección, o negativa, lo que significa que cuando el valor de una variable aumenta, los valores de las otras variables disminuyen. La correlación también puede ser neural o cero, lo que significa que las variables no están relacionadas. Correlación positiva: ambas variables cambian en la misma dirección. Correlación Neutral: Sin relación en el cambio de las variables. Correlación negativa: las variables cambian en direcciones opuestas.

El rendimiento de algunos algoritmos puede deteriorarse si dos o más variables están estrechamente relacionadas, lo que se denomina multicolinealidad.

Segundo, el coeficiente de correlación de Spearman (llamado así por **Charles Spearman**) se puede utilizar para resumir la fuerza entre las dos muestras de datos.

El coeficiente de correlación de orden de rango de Spearman es una medida no paramétrica de la monotonicidad de la relación entre dos conjuntos de datos. A diferencia de la correlación de Pearson, la correlación de Spearman no supone que ambos conjuntos de datos estén distribuidos normalmente. Al igual que otros coeficientes de correlación, éste varía entre -1 y +1 con 0 que implica que no hay correlación. Las correlaciones de -1 o +1 implican una relación monótona exacta. Las correlaciones positivas implican que a medida que x aumenta, también lo hace y. Las correlaciones negativas implican que a medida que x aumenta, y disminuye.

El valor p indica aproximadamente la probabilidad de que un sistema no correlacionado produzca conjuntos de datos que tengan una correlación de Spearman al menos tan extrema como la calculada a partir de esos conjuntos de datos.

Figura 4-7 Librería SciPy [Logo Corporativo]

4.2.8. ELI5

4.2.8.1. import eli5

ELI5 es un paquete Python que ayuda a depurar los clasificadores de aprendizaje de las máquinas y a explicar sus predicciones. Tiene incorporado soporte para varios marcos ML y proporciona una forma de explicar los modelos de caja negra.

ELI5 es otra biblioteca de visualización que es útil para depurar modelos de aprendizaje automático y explicar las predicciones que han producido. Funciona con las bibliotecas de aprendizaje automático de Python más comunes, incluyendo scikit-learn, XGBoost y Keras. ELI5 sirve para inspeccionar las características importantes del modelo que hemos entrenado anteriormente.

Por defecto, el método show_weights utiliza la ganancia para calcular el peso, pero se pueden especificar otros tipos añadiendo el argumento importance_type. También se puede usar show_prediction para inspeccionar las razones de las predicciones individuales.

4.2.9. XGBoost

XGBoost **eXtreme Gradient Boosting** es una librería de gradient boosting distribuida y optimizada, diseñada para ser altamente eficiente, flexible y portable. Implementa algoritmos de aprendizaje automático bajo el marco del Gradient Boosting. XGBoost proporciona un boosting de árbol paralelo (también conocido como GBDT, GBM) que resuelve muchos problemas de ciencia de datos de forma rápida y precisa. El mismo código se ejecuta en los principales entornos distribuidos (Hadoop, SGE, MPI) y puede resolver problemas más allá de miles de millones de ejemplos.

Figura 4-8 Librería XGBoost [Logo Corporativo]

4.3. Preprocesamiento: Transformación conjunto datos

4.3.1. Primer Preprocesamiento: Preparación del conjunto de datos en crudo

La calidad de los datos y la cantidad de información útil que contiene son factores clave que determinan qué tan bien puede aprender un algoritmo de aprendizaje automático. Por lo tanto, es absolutamente crítico que nos aseguremos de examinar y procesar un conjunto de datos antes de alimentarlo a un algoritmo de aprendizaje [56].

El primer paso en nuestro script del modelo detector **DetD&DoS** y preliminar al pre-procesamiento fuerte, es necesario preparar el conjunto de datos, pues sus datos están en crudo (raw), y de esta forma es imposible un correcto análisis, y la capacitación directa así en crudo, genera error de entrada en los algoritmos de aprendizaje supervisado y es irrealizable usarse.

La preparación y limpieza de los Ocho (8) archivos csv individuales, se inicia con renombrar cada uno de los 8 archivos de Variables separadas por Comas con un nombre para distinguir el tipo de ataque de denegación y denegación distribuida de sus etiquetas multiclase.

Para el estricto orden metodológico de nuestro script en lenguaje Python, es necesario definir la ruta donde se aloja nuestro conjunto de datos en los ficheros de nuestra PC, [dataset_directorio_raiz], así como también definir la carpeta donde nuestros archivos individuales se concatenaran y se eslabonaran en un único, y solo archivo .csv; pero antes de ello, definimos 3 funciones con un valor de retorno f, llamadas `def eliminar_cabeceras(f)`;`def reemplazar_infinitos(f)` y `def emilinar_no_caracteres_para_nombres_columnas(f)` que realizan las tareas de: primero, eliminar los encabezados duplicados que se presentan como filas del conjunto de datos pues si examinamos visualmente el dataframe de pandas,

se confirma que los encabezados están presentes varias veces dentro del archivo, entrelazados con las filas de datos sin procesar, tantas veces como el número de archivos concatenados .csv que repiten los encabezados en el proceso, y quedan como filas del archivo grande y único a crear extensión .csv; la segunda, función se usa para reemplazar y sustituir los valores en las columnas y celdas de `'Infinity'` por el valor de cadena `'inf'` ya que pandas no puede analizar correctamente otros valores de infinito pues solo reconoce las cadenas `inf / -inf` como una representación válida de infinito; y por último, para la tercera función tiene como fin cambiar la denominación de los nombres de columna para eliminar espacios en blanco, convertir las letras en minúsculas y eliminar los caracteres que no sean palabras, así dichos nombres de atributo o columna sean claros y al invocarlos del dataframe no surjan errores con el índice de columna.

Por ultimo para finalizar el primer paso de nuestro Script, en el ciclo del `for` se hace uso del método eliminar columnas `drop` porque de los 8 archivos, el más pesado de 4.1GB que fue modificado por problemas de mi capacidad de RAM del día martes 20 de febrero de 2018, contiene cuatro columnas de atributo (`Flow ID, Src IP, Dst IP, Src Port`) que son genéricos, independientes e indiferentes, y al no ser necesarios, se eliminan.

4.3.2. Segundo Preprocesamiento: a la concatenación

Datos de formato (*Formatting Data*) una de las formas más comunes de limpieza de datos es hacer que sus datos y tipos de datos ilegibles o difíciles de leer se ajusten a un formato legible adecuado. Especialmente si necesita crear informes con los datos o archivos descargables, querrá asegurarse de que sea legible por máquina a legible por humanos [57].

Iniciamos nuestro segundo paso del proceso metodológico, continuando a nuestro segundo pre-procesamiento, definiendo cada uno de tipo de dato de los diferentes atributos de columna de nuestro dataset, para ello asignamos un

diccionario con el nombre de cada uno de los 78 atributos de columna, más 1 de etiqueta para completar 79 columnas del dataframe legibles modificado en el paso anterior, encontrando tipos de datos como: `uint8`, `uint16`, `uint32`, `uint64`, `int64`, `float32`, `float64`, `object` y `category`. Después se prosigue definiendo 3 funciones que me retornan no un valor en este caso, sino un dataframe de pandas que profundizan la trasformación del csv final como un único archivo, dichas funciones las exprese como: `def remplazar_infinitos_por_promedio(df)`; `def remplazar_valores_negativos_por_promedio(df)`; y `def cargar_dataset(files, dtypes, cols=None)`.

La primera de estas funciones, reemplaza los valores de celda de la cadena `inf / -inf` y también los valores `fillna Fill NA/NaN` por el valor medio (*mean*) de celda; análogamente, la segunda función reemplaza los valores de celda negativos o menores de cero, por el valor medio (*mean*) de celda; y finalmente, la tercera función sirve para concatenar y unir todos los registros o filas individuales, y además creo 2 columnas finales adicionales para solucionar una lógica que me plantee pues mi objetivo final siempre fue pensado para primero dar una respuesta de **Clasificación Binaria**: traza Benigna / Ataque DoS-DDoS, así como la segunda columna extra, adicionada tiene la codificación del nombre del tipo de ataque para **Clasificación Multiclase** (que es la columna *label*) en un número, para así tener una etiqueta de clasificación multiclase codificada, no categórica; para finalmente agruparlos en un único archivo con extensión .csv ya limpio y con ningún valor en crudo, haciendo uso del método `concat`. Cabe mencionar que este archivo transformado de un único archivo, es el primer entregable de mi proyecto **DetD&DoS** planteado.

Como parte final del segundo paso, del segundo pre-procesamiento, el dataframe total unitario de nuestro conjunto de datos, es separado en la **matriz de**

variables predictoras [*predictor*] **X** & las **variables del vector objetivo** [*target*] **y**, haciendo uso nuevamente por el método eliminar columnas `drop`.

4.4. Procesamiento: reducción dimensionalidad

4.4.1. Ingeniería de atributos del conjunto de datos

La selección de atributos se presenta para solucionar el problema de aquellos atributos que pueden estar altamente relacionados, y por tanto, pueden ser redundantes hasta cierto punto.

Use 2 pasos de selección: 2 filtros. En el primer paso sencillo que pasa de 78 atributos originales a 69 atributos, reduciendo apenas 9 columnas; se aplica como primer filtro, usando el concepto de **desviación estándar** se eliminan todos los atributos con una cantidad cero, nula de variación; en el segundo filtro se crean estadísticas descriptivas a través de Pandas. Se seleccionan todos los atributos que tienen una de desviación estándar 0[Cero], y se asigna a los atributos de no varianza, y donde el listado de los valores más cortos, más bajos se eliminan dichos atributos nuevamente con `drop`. Para finalizar este primer paso, se elimina un último atributo `'timestamp'`, que además presenta un tipo de datos Object y que suele complicarse para codificar, y se procede a eliminar basados en la razón de ser atributo no deseado, y es independientes de los atributos de marca de tiempo para reconocer ataques indiferentes

Ahora, en el segundo paso, después de usar múltiples métodos, leer y consultar mucho. Se logra una reducción considerable de la dimensionalidad de nuestro *DataFrame*, al pasar de los 69 atributos del primer paso a tan solo 37 atributos en este segundo paso, donde se ha aplicado **alta correlación** para reducir dimensionalidad estos atributos correlacionados no traerán ninguna previsibilidad adicional, pero si como contra pueden introducir ruido, usando la **agrupación**

jerárquica en las correlaciones de **orden de rango de Spearman**, así se define un umbral de grupo, o racimo (*cluster*) igual a la unidad, a 1 de distancia como criterio, creando así una lista de dichos atributos, y aplicando un ciclo, o barrido por un `for` a la variable que denominados `cluster_id_atributos_ids`, creando así unas columnas resultante que llamamos `atributos_seleccion`, ,muy reducida en su dimensionalidad, y que representa fidedignamente toda la información contenida en la matriz de atributos original, que será reemplazada por la nueva la matriz de atributos X reducida.

4.4.2. División en conjunto de entrenamiento y conjunto prueba

Haciendo uso del método `train_test_split` de la librería scikit-learn, el archivo total unitario de nuestro conjunto de datos **CIC-IDS2017**, **CSE-CIC-IDS2018** & **CIC-DDoS2019** .csv, es dividido en 2 partes, haciendo uso del método de *split*, para así obtener dos (2) subconjuntos dados: entrenamiento prueba; cuyos porcentajes del archivo unitario creado son **90%** y **10%** respectivamente. Además de especificar, y a su vez tener por separado la matriz de predicción $\boldsymbol{X_i}$ y el vector $\boldsymbol{y_i}$, cada uno con 2 subconjuntos asignados, obteniendo 4 arreglos llamados $\boldsymbol{X_{entrena}}$, $\boldsymbol{X_{prueba}}$, $\boldsymbol{y_{entrena}}$, $\boldsymbol{y_{prueba}}$. Así con esto ya está dada la forma correcta de los datos, todo el preprocesamiento fue realizado con arduo trabajo porque aquí radica la importancia vital del aprendizaje: tanto la adquisición del conjunto de datos, como el formato correcto para ser la data de ingreso a los algoritmos de clasificación usados, que en verdad los datos estén traducidos correctamente para que el lenguaje maquina puede leer y nuestro modelo detector entrene, aprenda y sea correctamente leído por nuestros múltiples algoritmos de aprendizaje automático que aplicamos.

Para finalizar y ser previo al proceso de aprendizaje, hacemos dos pequeños análisis. El primero, de distribución de las etiquetas Dos y DDoS, por supuesto en

los 3 <*Vector y*>, del total y de los 2 subconjuntos aplicados tanto entrena, como prueba, para mirar que tan desequilibrada podrían encontrase en las etiquetas. Y el segundo, el porcentaje de benigno que en el solo conjunto original de CIC-2018 era muy alto, en comparación de los otros, era tan desequilibrado que su porcentaje de trazas benignas era muy alto, y se ha solucionado adicionando y agregando trazas del CIC-2019 y CIC-2017. Pues la primera solución que se aplicó, y se ha descartado de crear trafico sintético usando ***SMOTE()*** y ***SMOTENC()***, la verdad no cumplía mis expectativas, y es poco fiable y tienen muchas equivocaciones generar trafico sintético, por lo visto en trabajos que hasta generan en sus conjuntos de datos total de trafico sintético.

4.5. Etapa Aprendizaje: Entrenamiento

Es vital importancia indicar que llevaremos a cabo el uso matemático, estadístico y probabilístico de los diferentes algoritmos de aprendizaje automático, gracias a la poderosa herramienta de la librería de Scikit-learn y de la librería XGBoost, que proporciona una interfaz intuitiva para usar estos algoritmos de forma eficiente y productiva.

Ya determinado el ***Recall*** como unidad de medida para la clasificación, ya en el diseño del modelo, antes de aplicar los dichos algoritmos de clasificación, es como definimos 2 funciones para medir y comparar el rendimiento de cada algoritmo de clasificación, llamadas `def graficar_matriz_confusion` y `def imprimir_reporte`, la primera función realiza la construcción de la gráfica de la matriz de confusión gracias a la librería matplotlib.pyplot, y la segunda función, para plasmar y resumir toda las unidades métricas más importantes y que reflejan cabalmente el rendimiento: ***precision***, ***recall*** y ***f1-score***; y dos gráficas de curvas de aprendizaje.

4.5.1. Selección del Modelo: Algoritmos de Clasificadores -Classifiers-

Como ya definimos los datos de entrenamiento, el siguiente paso es seleccionar el algoritmo de clasificación que se va a emplear. En scikit-learn, esto se hace mediante la creación de un objeto ***estimator*** que cumple la función de estimador. En concreto, este objeto almacena el nombre del algoritmo, sus parámetros e hiperparámetros y contiene los métodos ***fit***(X, y) y ***predict***(T) que le permiten aprender de los datos y predecir nuevas observaciones, respectivamente.

- ***Supervised learning Classifier***
  - Linear Models
    - **Logistic regression**
    - **Passive Aggressive Classifier**
    - **Perceptron**
    - **SGD Classifier (Descenso de Gradiente Estocástico)**
    - **Ridge Classifier**
  - Discriminant Analysis
    - **Linear Discriminant Analysis LDA**
    - **Quadratic Discriminant Analysis QDA**
  - Naive Bayes
    - **Gaussian Naive Bayes**
    - **Bernoulli Naive Bayes**
    - **Multinomial Naive Bayes**
    - **Complement Naive Bayes**
  - Nearest Neighbors

- **Nearest Centroid**

- Neural Network
  - **MLPClassifier**
- Tree
  - **Decision Tree Classifier**
  - **Extra Tree Classifier**
- Ensemble
  - **Random Forest Classifier**
  - **Extra Trees Classifier**
  - **AdaBoost Classifier**
  - **Gradient Boosting Classifier**
- Extreme Gradient Boosting
  - **xgboost.DMatrix** and **xgboost.train**
  - **xgboost.XGBClassifier**

Solo se obtuvo y convergió resultados de estos 21 algoritmos de aprendizaje, donde se presenta una tabla comparativa de resultados muy concluyente y que resume todo, ver en el capítulo 5 de resultados experimentales, sección 5.3 resultados de entrenamiento. Aunque se aplicaron muchos más, por múltiples fallas como cuestión de tiempo requerido pasando a esperar 25 horas de trabajo y nada, no convergían en tiempos de entrenamiento ***K Neighbors Classifier*** y los ***Support Vector Machines Classification*** tanto en máquinas de vectores tipo SVC – Linear y de SVR en varios kernel.

4.5.2. Medición del Rendimiento: Conjunto Entrenamiento

El aprendizaje, basado en la etapa de entrenamiento, gracias al método ***fit***(X, y), podemos realizar como análisis y obtener de resultados: matrices de confusión multiclase del entreno, un ***reporte de clasificación del entreno***, grafica de ***curva ROC*** del entrenamiento y la gráfica del entrenamiento de la ***curva de precisión-recall***, y en base a esos resultados decidir de todos los 22 algoritmos de aprendizaje de clasificación, cuál de ellos tiene mejor rendimiento, y en nuestro caso tanto por excelentes métricas, como el tiempo de duración: elegimos del estimador final a `estimator_final = clf_RandomForest`.

4.5.3. Validación cruzada: 3 mejores algoritmos de aprendizaje

Ahora, de la selección de 22 modelos anteriores solo se prosigue con tres de ellos, los tres mejores, el pódium. De todas las técnicas de validación cruzada existentes de la librería scikit-learn, se decidió por la más completa sin lugar a dudas:

```
from sklearn.model_selection import cross_validate, cross_val_predict
```

Esta ***Cross Validate*** de forma estratificada, esto es que cada una de las 10 porciones a dividir, se distribuyan por igual el porcentaje de clases o familias de ataques de negación. Es la mejor en cuanto a resultados de métricas, ya que permite elegir muchas, y es una ventaja para nosotros para saber resultados tanto micro, como macro.

```
scoring = ['accuracy', 'precision_micro', 'recall_micro', 'f1_micro', 'precision_macro', 'recall_macro' , 'f1_macro']
cv = StratifiedKFold(n_splits=10)
```

4.5.4. Importancia de los atributos: método del algoritmo de aprendizaje

Para saber la importancia de los 37 atributos que son ingresados como la entrada del algoritmo de aprendizaje, su importancia radica tanto para los primeros, los más importantes; como los últimos, o los que casi no representan peso en su decisión, o muy cercano a cero, en el capítulo posterior de resultamos experimentales se dirá algo acerca de este análisis de la importancia de los atributos.

```
importancia_RandomForest =
clf_RandomForest.feature_importances_

plt.barh(X_train.columns, importancia_RandomForest)
sorted_idxx = importancia_RandomForest.argsort()
plt.barh(X_train.columns[sorted_idxx],
importancia_RandomForest[sorted_idxx])
plt.xlabel("Random Forest  Feature Importance: ")
```

4.5.5. Diagrama de árbol unitarios: DT y ET

Para solo los algoritmos de árbol unitarios, esto es: de árbol de decisión y el árbol extremo se les puede aplicar las funciones de `plot_tree`, `export_text`, `export_graphviz` de la clase ***.tree*** para así saber cómo piensa, aprende y decide un árbol, con toda la complejidad matemática, y formalismo de probabilidad en su toma de decisión de clasificación multiclase.

```
from sklearn.tree import plot_tree, export_text,
export_graphviz

_ = plot_tree(clf_ExtremelyTree, feature_names=feac,
class_names=cn, filled=True)
fig_ET.savefig("arbol_ExtremelyTree.png")
```

4.5.6. Guardar el modelo: conservar el entrenamiento

Como ya se ha mencionado en el capítulo 3 de la metodología planteada, es necesario guardar (***save***) y almacenar el entrenamiento y aprendizaje .***fit***(X, y) de los algoritmos del modelo, esto ya que se tiene en la RAM, y antes que se volquee y se pierda, con otra depuración diferente, o si debe apagar: se guarda como fichero en extensión binaria ***.bin*** gracias al empleo de la librería **Pickle**, que además es la misma usada para cargar (***load***) y así ser uso de lo ya entrenado y aprendido del modelo.

```
import pickle

pickle_salida = open('clf_RandomForest.bin', 'wb')
pickle.dump(clf_RandomForest, pickle_salida)
pickle_salida.close()

pickle_entrada = open('clf_RandomForest.bin', 'rb')
clf_RandomForest = pickle.load(pickle_entrada)
```

4.6. Evaluación de Prueba: Métricas de rendimiento de conjunto de Prueba

Se prosigue con la etapa que valida que tan bien estamos realizando las cosas, los resultados de dicho estimador final, para el conjunto de datos de prueba, de un total de **1’359.628 flujos** de tráfico de red o filas, se representan en esta evaluación.

Se usaron múltiples, bastantes y muchas métricas, tanto, cómo las disponibles en la librería scikit-learn; y fue tomado en serio el aunar e inspeccionar. Para detallar los siguientes:

```
from sklearn.metrics import accuracy_score,
balanced_accuracy_score, precision_score,
average_precision_score, f1_score, log_loss, recall_score,
roc_auc_score, brier_score_loss, hamming_loss,
```

```
confusion_matrix, plot_confusion_matrix, hinge_loss, plot_roc_curve

from sklearn.metrics import plot_roc_curve, zero_one_loss, roc_curve, precision_recall_curve, jaccard_score, dcg_score, top_k_accuracy_score
```

Donde todas comparan la etiqueta predicha por el modelo con la etiqueta real, flujo a flujo, así se ha evaluado con el poder de la etiqueta, el aprendizaje supervisado.

A continuación también observamos los resultados de la lista de clasificaciones erróneas en el conjunto de prueba, gracias a la función creada llamada `def calcular_error_clasificaciones` que demuestran que muy pocos ataques a menudo se clasifican erróneamente, es decir, las clases minoritarias no están tan desequilibradas, y se mejoró inmensamente el balanceo de las diferentes etiquetas de ataques DoS y DDoS, gracias al mejoro los datos de entrada con la unión y adición de los conjuntos de datos de 3 años: CIC-IDS2017, CSE-CIC-IDS2018 & CIC-DDoS2019.

4.7. Etapa Evaluación de Pruebas DDoS 2019: Doble métricas de rendimiento de clasificación de flujos de datos no vistos

Ya seleccionamos un modelo detector cimentado en el conjunto de datos de entrenamiento, ya utilizamos el conjunto de datos de prueba para estimar como funciona y se desempeña con el 10% no visto y estratificado de (más de millón trescientos flujos). Pero queremos ir más allá, otra etapa de evaluación, sería una doble etapa evaluación, y esta vez, con un adicional de dificultad pues será un poco

más grande, de un millón quinientos flujos de tráfico de red, y totalmente desequilibrado hacia ataques de denegación, muy pocos flujos de tráfico benigno.

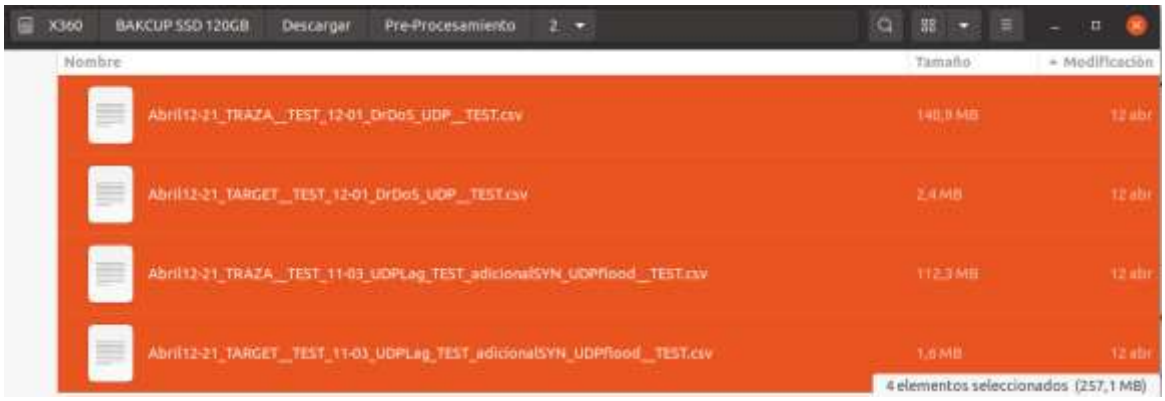


Figura 4-9 Conjunto Datos etiquetado no visto, Segunda Evaluación [elaboración propia]

La idea con esto, es poner el modelo detector DetD&DoS propuesto, en una situación de dificultad, haciendo la idea de un simulacro de un ataque de denegación de servicios de gran cantidad de datos, y ver cómo funciona en una situación fuera de confort. La idea también nació tanto en la delimitación y alcance del proyecto planteado, dicho en la sección de anotaciones 1.3.3, donde siguiendo la catedra del profesor BCS (IAU), MCS (UM), PhD (UTM), Post-Doc (UNB) Arash Habibi Lashkari, quien ha implementado los conjuntos de datos de CIC de nuestro estudio, su adopción en aplicaciones del mundo real se ha visto obstaculizada por la complejidad del sistema, ya que estos sistemas requieren una cantidad considerable de pruebas (*testing*), evaluaciones(*evaluation*) y ajustes(*tuning*) antes de su implantación (*deployment*) [50]. Además, también nace de la disponibilidad, y gran valor que tiene el etiquetado de un conjunto de datos, y aprovechando la extensión y grandeza que se compone el conjunto de datos de CIC-DDoS2019, se toman de 3 tipos de Ataques DDoS aprendidos: **SYN_Flood**, **UDP_Flood** y **UDPLag.** Para nuestro caso puntual, ver Figura 4-9, la transformación de unos días específicos en CIC-DDoS2019, para formar 4 .csv a saber, con sus respectivas etiquetas reales.

Donde dichos 4 subconjuntos son de los 3 tipos de Ataques DDoS, que ha aprendido: uno de **UDP_Flood** y otro de una mezcla de **UDPLag, SYN_Flood y UDP_Flood**. Y su preprocesamiento, para obtener dichas trazas de archivos .CSV, ha sido igual a la forma desarrollada, pues deben tener el mismo y exacto formato de entrada del modelo para poder realizar el método .***predict()*** correcta e idóneamente, para ello se ha usado otro *script* aparte, para plasmar igual proceso de prepocesamiento en sus II partes y el procesamiento de una forma simple, eliminado mediante .*drop*() las columnas que no pertenecen. Este mismo análisis, y *script* se aplican de igual manera, idéntico al trafico capturado y esnifado de la etapa próxima de implementación.

Lo más importante a resaltar en dicho *script*, es que los datos del CIC-DDoS2019, cambian un poco, muy somero, pues encontramos más número de columnas, pero se proceden a eliminar: **`Unnamed 0`**, **`Fwd HeaderLength.1`**, **`SimillarHTTP`**, **`Inbound`**.

Y de allí, todo el preprocesamiento debe ser el mismo, hasta que llegamos a seleccionar los **37 atributos** que, en el procesamiento fuerte, se ha procesado su análisis de selección. al final, se busca que todos los nombres de atributos, como los nombres de las etiquetas cambia de un año a otro, se aplica un diccionario para relacionar los mismos e idénticos nombres de ataques de DoS y DDoS. Para finalmente, dicho dataframe sea guardado como un archivo de variables separadas por coma (csv).

4.8. Implementación en red hogar *realtime*

Se ha usado para monitorizar y esnifar el tráfico TCP/IP y hacer capturas de paquetes de tráfico de red en formato paquetes capturados (pcap) en tiempo real,

en vivo, se ha implementado, se ha esnifando, es decir, capturado a través de mi tarjeta de red por la interfaz del puerto ethernet, en mi caso llamado **eno1**.

4.8.1. Etapa Previa: Capturar Tráfico de red

Esta etapa se ha ubicado en la lógica de la metodología planteada, como etapa cero, ó preliminar. Se ha usado la herramienta **CICFlowMeter** v4.0 y v3.0, de forma manualmente. Se usa la pestaña <<**Network**>> como se puede ver la Figura 4-11, en modo ***realtime*** para esnifar y capturar tráfico de red. Primero oprimir el botón **load** para seleccionar la interfaz gráfica **eno1** como se aprecia en la Figura 4-10, y botón **star**, para iniciar y al instante que se dé al botón **stop**, se almacena el archivo esnifado en formato de variable separa por comas crudo, o primario. Posterior hay un paso previo para poder ser utilizado, pues al hacer al

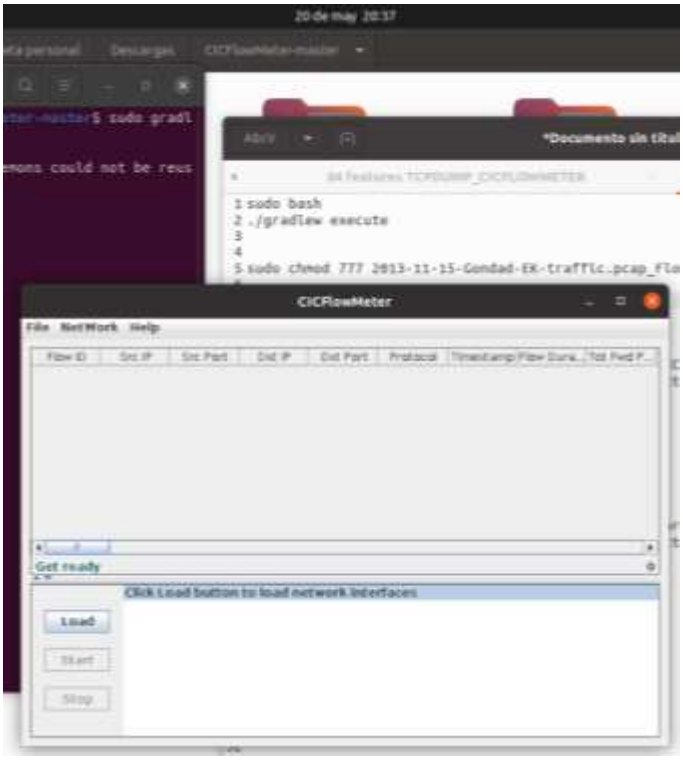


Figura 4-11 herramienta CICFlowMeter botón Cargar [Elaboración propia]

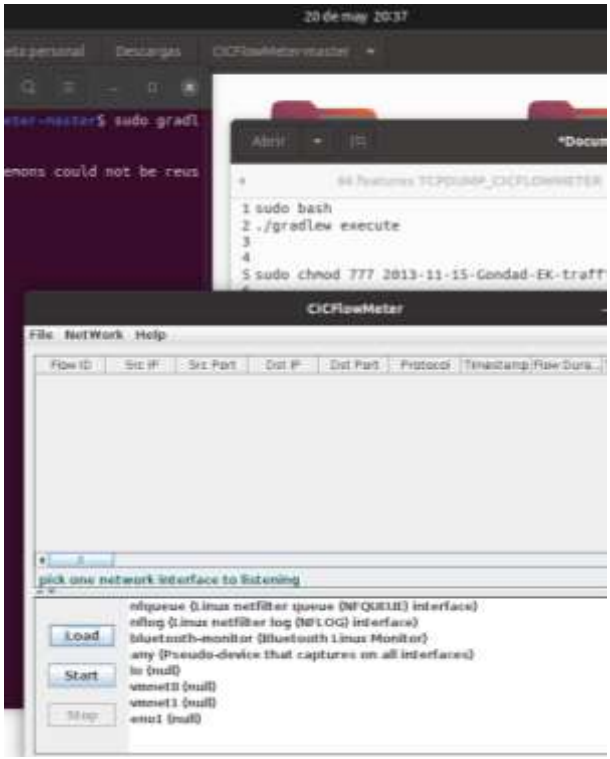


Figura 4-10 herramienta CICFlowMeter botón Iniciar [Elaboración propia]

CICFlowMeter desde sudo, dichos archivos de paquetes capturados quedan con seguridad de *root*, y se debe quitar dicha seguridad editando sus permisos.

El otro modo que tienen esta herramienta del CIC, es el modo ***offline*** cuya funcionalidad podemos pasar una traza o captura de paquetes ya almacena de antes, o que se dese analizar; para ello se dispone en el apartado superior, pestaña, ver Figura 4-12 para presenciar <<**Network**>>, **offline**. A lo cual, ahora será una pantalla en ventana tipo ficheros, donde se debe especificar **carpeta origen** y **carpeta destino**, y procesara con el botón **OK** el número de paquetes, tanto de

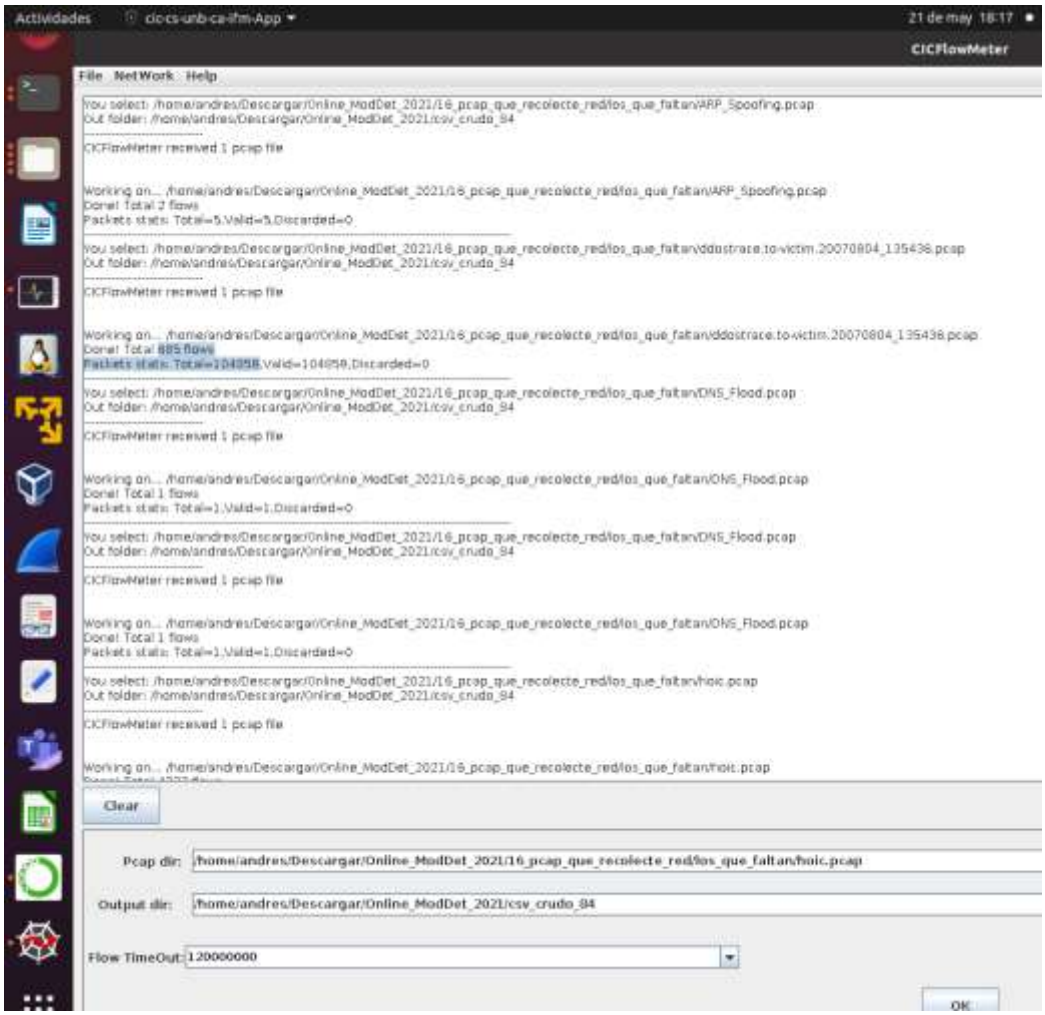


Figura 4-12 herramienta CICFlowMeter modo *offline* [Elaboración propia]

paquetes válidos, ó paquetes descartados, y los reducirá a los flujos que contenga dicha captura de paquetes de tráfico.

Se debe tener muy en claro que, así sea que se esnife, o que se convierta un archivo paquetes de captura (pcap) ya previo, el resultado siempre es obtener un **archivo de variables separadas por comas en crudo**, en bruto. Dicho archivo csv en crudo será la entrada de nuestro modelo de aprendizaje que toma la decisión de identificar y clasificar.

4.8.2. Usar el modelo definitivo para tomar una decisión final

Cuando se ha obteniendo el archivo en variables separadas por comas (csv) en bruto, se hace los mismo como ya se había mencionado anteriormente:

Primero se pre-procesa, seguido del procesamiento sencillo que deja con 37 atributos, para que directo sea aplicado al modelo cargado, y se utiliza tanto método ***predict***(X) como el método ***predict_proba***(X), para predecir las probabilidades de clase de ataque maliciosos o en su defecto benigno de las muestras de flujos de tráfico de red, en este caso en tiempo real, en directo.

Lo que resulta en un vector columna de $\boldsymbol{n} * \mathbf{1}$ y en una matriz de probabilidades $\boldsymbol{n} * \mathbf{11}$, que contienen las etiquetas de las 11 clases identificadas y clasificadas, en el primer caso la decisión del algoritmo del aprendizaje de nuestro modelo, y en el segundo caso de la matriz, expresa las probabilidades de su decisión, que sumarían siempre la unidad, y es sumamente importante en el análisis de resultados, y que sería información valiosa para un empleado Soporte IT, Seguridad o analista de tráfico.

Se concluye esta implementación, interpretando los resultados como representaciones gráficas:

- Para el caso del vector columna de predicción se usan gráfico de **barras** y gráfico de **torta**.
- Y para el caso de la matriz de predicción de probabilidades, se muestra 12 gráficos de **dispersión**, uno conjunto, y 11 individuales, uno por cada clase, para poder apreciar todo con detalle y minuciosidad.

A modo de conclusión y claridad, CICFlowMeter, nunca dispone de brindar información de la etiqueta como se ve en la Figura 4-13. Ese el poder del etiquetado de los datos de forma manual por un experto en redes, o del poder del modelo de aprendizaje automático.

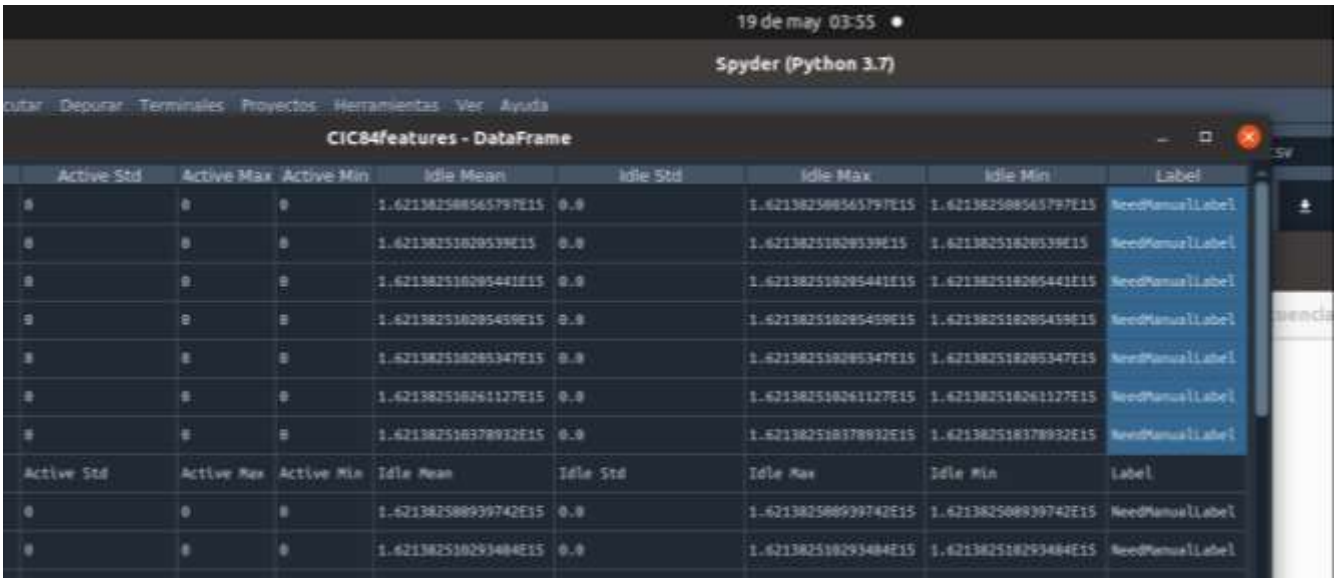


Figura 4-13 Conversion de PCAP a CSV sin la obia etiqueta de Ataque Malicioso [Elaboración propia]

Capítulo 5

Resultados Experimentales

En este quinto capítulo se manifiestan todos los resultados prácticos y experimentales que se han obtenido a lo largo de la implementación de código de programación de lenguaje Python, para crear nuestro modelo detector y clasificador de ataques maliciosos de denegación **DetD&DoS** siguiendo siempre la misma hoja de ruta, o flujo de trabajo, del diseño que se ha propuesto.

5.1. Resultados de Preprocesamiento

5.1.1. Primer Preprocesamiento: Preparación

Para ver gráficamente, y diferenciar las dos situaciones de evolución de distribución de clases, cantidad de ataques maliciosos por clases de ataques de denegación de servicio, que se ha recorrido por todo el proceso del proyecto de investigación. Veremos dos graficas de barras, la primera Figura 5-1 solo se ha usado el conjunto **CSE-CIC-IDS2018** únicamente y muy extenso en diferentes

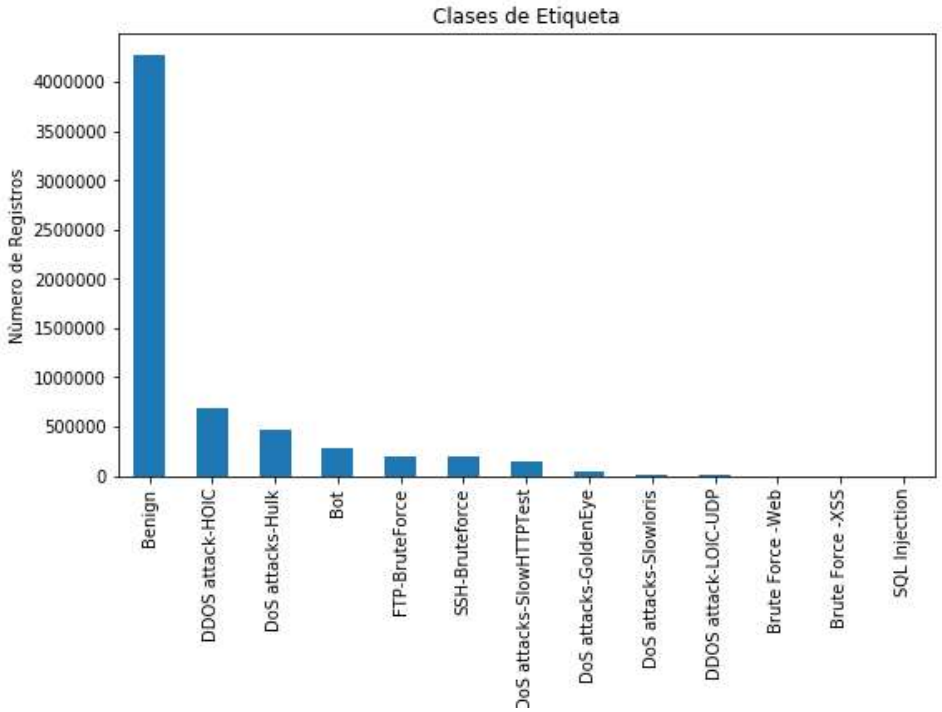


Figura 5-1 Distribución ataque maliciosos CSE-CIC-IDS2018 [Elaboración propia]

tipos/familias de ataques, como son los ataques de Red de Robots (*Bot*), Fuerza Bruta (*FTP, SSH, XSS Web BruteForce*) y una mínima cantidad de *SQL Injection*

En comparación a la segunda Figura 5-2, se ha propuesto para la mejora del conjunto de datos de entrada de entrenamiento, en cuanto a una renovada distribución de clases, de una combinación específica y editada (pues se ha eliminado exceso de tráfico benigno) de ataques maliciosos de denegación de servicios de **CIC-IDS2017, CSE-CIC-IDS2018 & CIC-DDoS2019**:

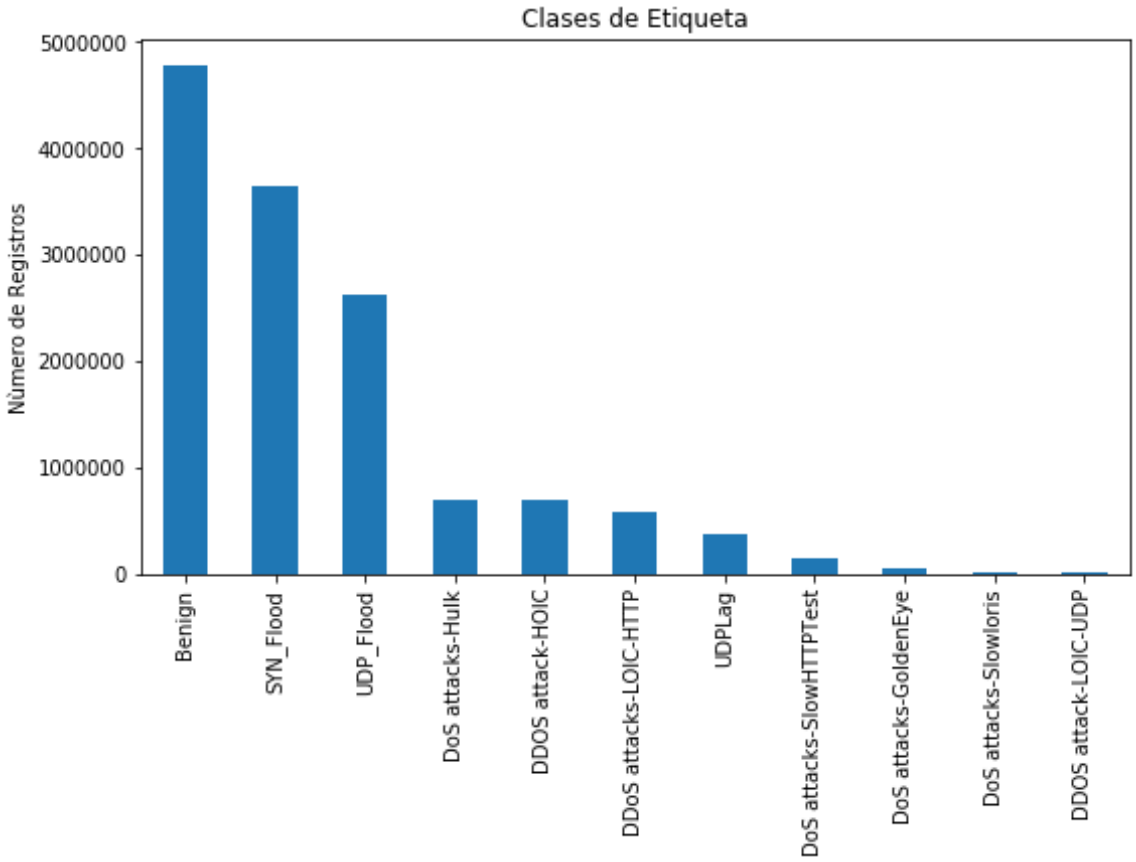


Figura 5-2 Distribución de ataque maliciosos específicos DoS y DDoS CIC-IDS2017, CSE-CIC-IDS2018 & CIC-DDoS2019 [Elaboración propia]

Veremos a continuación las cuatro formas distintas que han escrito los nombres de atributos presentando desigualdades adicionalmente del tamaño de

atributos, dependiendo el año del conjunto de datos del CIC, y se han incluido las versiones del CICFlowMeter que se ha usado. Así, los cuatro siguientes serían los formatos de los archivos de variables separadas por comas en **crudo**, en **primario**, en **bruto**; y el quinto es el formato que se ha tomado por estándar en la escritura de cada uno de los 37 atributos finales.

Formato de nombres de atributos CIC-IDS-2018

```
Df2018          =          pd.read_csv(r'/media/andres/X360/CIC_MIX_JAPJ_2019-2018-2017/Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7948748 entries, 0 to 7948747
Data columns (total 84 columns):
 #   Column             Dtype
---  ------             -----
 0   Flow ID            object
 1   Src IP             object
 2   Src Port           int64
 3   Dst IP             object
 4   Dst Port           int64
 5   Protocol           int64
 6   Timestamp          object
 7   Flow Duration      int64
 8   Tot Fwd Pkts       int64
 9   Tot Bwd Pkts       int64
 10  TotLen Fwd Pkts    float64
 11  TotLen Bwd Pkts    float64
 12  Fwd Pkt Len Max    float64
 13  Fwd Pkt Len Min    float64
 14  Fwd Pkt Len Mean   float64
 15  Fwd Pkt Len Std    float64
 16  Bwd Pkt Len Max    float64
 17  Bwd Pkt Len Min    float64
 18  Bwd Pkt Len Mean   float64
 19  Bwd Pkt Len Std    float64
 20  Flow Byts/s        float64
 21  Flow Pkts/s        float64
 22  Flow IAT Mean      float64
 23  Flow IAT Std       float64
 24  Flow IAT Max       float64
 25  Flow IAT Min       float64
 26  Fwd IAT Tot        float64
 27  Fwd IAT Mean       float64
 28  Fwd IAT Std        float64
 29  Fwd IAT Max        float64
 30  Fwd IAT Min        float64
 31  Bwd IAT Tot        float64
```

```
 32  Bwd IAT Mean       float64
 33  Bwd IAT Std        float64
 34  Bwd IAT Max        float64
 35  Bwd IAT Min        float64
 36  Fwd PSH Flags      int64
 37  Bwd PSH Flags      int64
 38  Fwd URG Flags      int64
 39  Bwd URG Flags      int64
 40  Fwd Header Len     int64
 41  Bwd Header Len     int64
 42  Fwd Pkts/s         float64
 43  Bwd Pkts/s         float64
 44  Pkt Len Min        float64
 45  Pkt Len Max        float64
 46  Pkt Len Mean       float64
 47  Pkt Len Std        float64
 48  Pkt Len Var        float64
 49  FIN Flag Cnt       int64
 50  SYN Flag Cnt       int64
 51  RST Flag Cnt       int64
 52  PSH Flag Cnt       int64
 53  ACK Flag Cnt       int64
 54  URG Flag Cnt       int64
 55  CWE Flag Count     int64
 56  ECE Flag Cnt       int64
 57  Down/Up Ratio      float64
 58  Pkt Size Avg       float64
 59  Fwd Seg Size Avg   float64
 60  Bwd Seg Size Avg   float64
 61  Fwd Byts/b Avg     int64
 62  Fwd Pkts/b Avg     int64
 63  Fwd Blk Rate Avg   int64
 64  Bwd Byts/b Avg     int64
 65  Bwd Pkts/b Avg     int64
 66  Bwd Blk Rate Avg   int64
 67  Subflow Fwd Pkts   int64
 68  Subflow Fwd Byts   int64
 69  Subflow Bwd Pkts   int64
 70  Subflow Bwd Byts   int64
 71  Init Fwd Win Byts  int64
 72  Init Bwd Win Byts  int64
 73  Fwd Act Data Pkts  int64
 74  Fwd Seg Size Min   int64
 75  Active Mean        float64
 76  Active Std         float64
 77  Active Max         float64
 78  Active Min         float64
 79  Idle Mean          float64
 80  Idle Std           float64
 81  Idle Max           float64
 82  Idle Min           float64
 83  Label              object
dtypes: float64(45), int64(34), object(5)
memory usage: 5.0+ GB
```

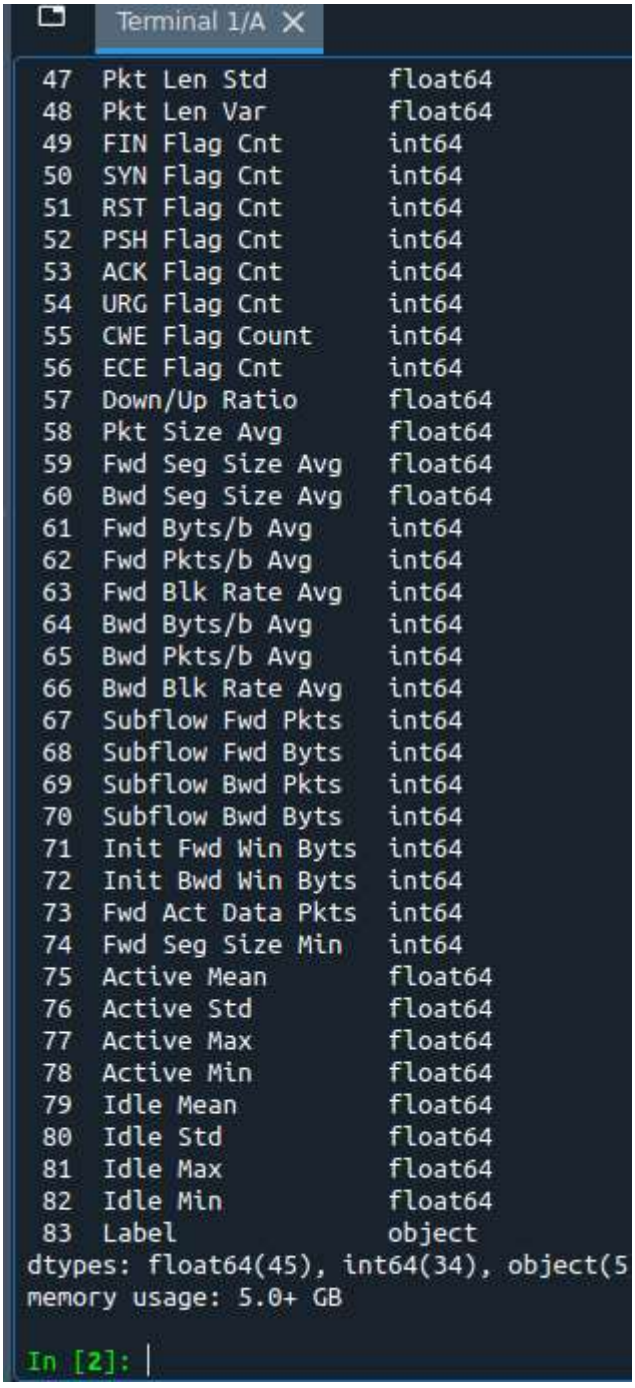


Figura 5-3 Formato nombres atributos CIC-IDS2018 [Elaboración propia]

Formato de nombres de atributos CIC-DDoS-2019

df2019 = pd.read_csv(r'/media/andres/X360/BAKCUP SSD 120GB/Descargar/nuevo, lo que va agregando/**CSV-01-12 (1)/01-12/Syn.csv**')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1582681 entries, 0 to 1582680
Data columns (**total 88 columns**):

#	Column	**Non-Null Count**	Dtype
0	**Unnamed: 0**	1582681	
1	Flow ID	1582681	
2	Source IP	1582681	
3	Source Port	1582681	
4	Destination IP	1582681	
5	Destination Port	1582681	
6	Protocol	1582681	
7	Timestamp	1582681	
8	Flow Duration	1582681	
9	Total Fwd Packets	1582681	
10	Total Backward Packets	1582681	
11	**Total Length of Fwd Packets**	1582681	
12	**Total Length of Bwd Packets**	1582681	
13	Fwd Packet Length Max	1582681	
14	Fwd Packet Length Min	1582681	
15	Fwd Packet Length Mean	1582681	
16	Fwd Packet Length Std	1582681	
17	**Bwd Packet Length Max**	1582681	
18	**Bwd Packet Length Min**	1582681	
19	Bwd Packet Length Mean	1582681	
20	Bwd Packet Length Std	1582681	
21	Flow Bytes/s	1380404	
22	Flow Packets/s	1582681	
23	Flow IAT Mean	1582681	
24	Flow IAT Std	1582681	
25	Flow IAT Max	1582681	
26	Flow IAT Min	1582681	
27	Fwd IAT Total	1582681	
28	Fwd IAT Mean	1582681	
29	Fwd IAT Std	1582681	
30	Fwd IAT Max	1582681	
31	Fwd IAT Min	1582681	
32	Bwd IAT Total	1582681	
33	Bwd IAT Mean	1582681	
34	Bwd IAT Std	1582681	
35	Bwd IAT Max	1582681	
36	Bwd IAT Min	1582681	
37	Fwd PSH Flags	1582681	
38	Bwd PSH Flags	1582681	
39	Fwd URG Flags	1582681	
40	Bwd URG Flags	1582681	
41	Fwd Header Length	1582681	
42	Bwd Header Length	1582681	
43	Fwd Packets/s	1582681	
44	Bwd Packets/s	1582681	

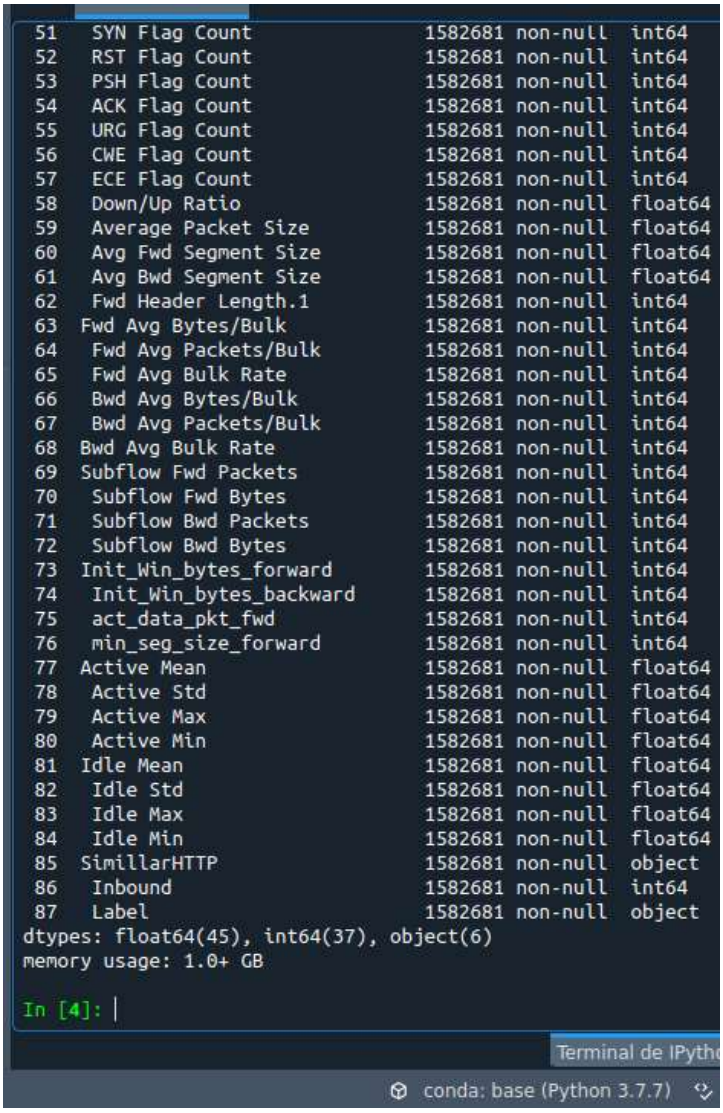


Figura 5-4 Formato de nombres de atributos CIC-DDoS-2019 [Elaboración propia]

```
 45   Min Packet Length            1582681 non-null  float64
 46   Max Packet Length            1582681 non-null  float64
 47   Packet Length Mean           1582681 non-null  float64
 48   Packet Length Std            1582681 non-null  float64
 49   Packet Length Variance       1582681 non-null  float64
 50  FIN Flag Count                1582681 non-null  int64
 51   SYN Flag Count               1582681 non-null  int64
 52   RST Flag Count               1582681 non-null  int64
 53   PSH Flag Count               1582681 non-null  int64
 54   ACK Flag Count               1582681 non-null  int64
 55   URG Flag Count               1582681 non-null  int64
 56   CWE Flag Count               1582681 non-null  int64
 57   ECE Flag Count               1582681 non-null  int64
 58   Down/Up Ratio                1582681 non-null  float64
 59   Average Packet Size          1582681 non-null  float64
 60   Avg Fwd Segment Size         1582681 non-null  float64
 61   Avg Bwd Segment Size         1582681 non-null  float64
 62   Fwd Header Length.1          1582681 non-null  int64
 63  Fwd Avg Bytes/Bulk            1582681 non-null  int64
 64   Fwd Avg Packets/Bulk         1582681 non-null  int64
 65   Fwd Avg Bulk Rate            1582681 non-null  int64
 66   Bwd Avg Bytes/Bulk           1582681 non-null  int64
 67   Bwd Avg Packets/Bulk         1582681 non-null  int64
 68  Bwd Avg Bulk Rate             1582681 non-null  int64
 69  Subflow Fwd Packets           1582681 non-null  int64
 70   Subflow Fwd Bytes            1582681 non-null  int64
 71   Subflow Bwd Packets          1582681 non-null  int64
 72   Subflow Bwd Bytes            1582681 non-null  int64
 73  Init_Win_bytes_forward        1582681 non-null  int64
 74   Init_Win_bytes_backward      1582681 non-null  int64
 75   act_data_pkt_fwd             1582681 non-null  int64
 76   min_seg_size_forward         1582681 non-null  int64
 77  Active Mean                   1582681 non-null  float64
 78   Active Std                   1582681 non-null  float64
 79   Active Max                   1582681 non-null  float64
 80   Active Min                   1582681 non-null  float64
 81  Idle Mean                     1582681 non-null  float64
 82   Idle Std                     1582681 non-null  float64
 83   Idle Max                     1582681 non-null  float64
 84   Idle Min                     1582681 non-null  float64
 85  SimillarHTTP                  1582681 non-null  object
 86   Inbound                      1582681 non-null  int64
 87   Label                        1582681 non-null  object
dtypes: float64(45), int64(37), object(6)
memory usage: 1.0+ GB
```

Formato de nombres de atributos CIC-IDS-2017

```
df2017 = pd.read_csv(r'/media/andres/X360/CIC_MIX_JAPJ_2019-2018-2017/Wednesday-workingHours.pcap_ISCX.csv')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 692703 entries, 0 to 692702
Data columns (total 85 columns):
```

#	Column	Non-Null Count	Dtype
0	Flow ID	692703	
1	Source IP	692703	
2	Source Port	692703	
3	Destination IP	692703	
4	Destination Port	692703	
5	Protocol	692703	
6	Timestamp	692703	
7	Flow Duration	692703	
8	Total Fwd Packets	692703	
9	Total Backward Packets	692703	
10	**Total Length of Fwd Packets**	692703	
11	**Total Length of Bwd Packets**	692703	
12	**Fwd Packet Length Max**	692703	
13	**Fwd Packet Length Min**	692703	
14	**Fwd Packet Length Mean**	692703	
15	**Fwd Packet Length Std**	692703	
16	**Bwd Packet Length Max**	692703	
17	**Bwd Packet Length Min**	692703	
18	**Bwd Packet Length Mean**	692703	
19	**Bwd Packet Length Std**	692703	
20	Flow Bytes/s	691695	
21	Flow Packets/s	692703	
22	Flow IAT Mean	692703	
23	Flow IAT Std	692703	
24	Flow IAT Max	692703	
25	Flow IAT Min	692703	
26	Fwd IAT Total	692703	
27	Fwd IAT Mean	692703	
28	Fwd IAT Std	692703	
29	Fwd IAT Max	692703	
30	Fwd IAT Min	692703	
31	Bwd IAT Total	692703	
32	Bwd IAT Mean	692703	
33	Bwd IAT Std	692703	
34	Bwd IAT Max	692703	
35	Bwd IAT Min	692703	
36	Fwd PSH Flags	692703	
37	Bwd PSH Flags	692703	
38	Fwd URG Flags	692703	
39	Bwd URG Flags	692703	
40	Fwd Header Length	692703	
41	Bwd Header Length	692703	
42	Fwd Packets/s	692703	
43	Bwd Packets/s	692703	
44	Min Packet Length	692703	
45	Max Packet Length	692703	
46	Packet Length Mean	692703	
47	Packet Length Std	692703	
48	Packet Length Variance	692703	
49	FIN Flag Count	692703	
50	SYN Flag Count	692703	
51	RST Flag Count	692703	
52	PSH Flag Count	692703	
53	ACK Flag Count	692703	

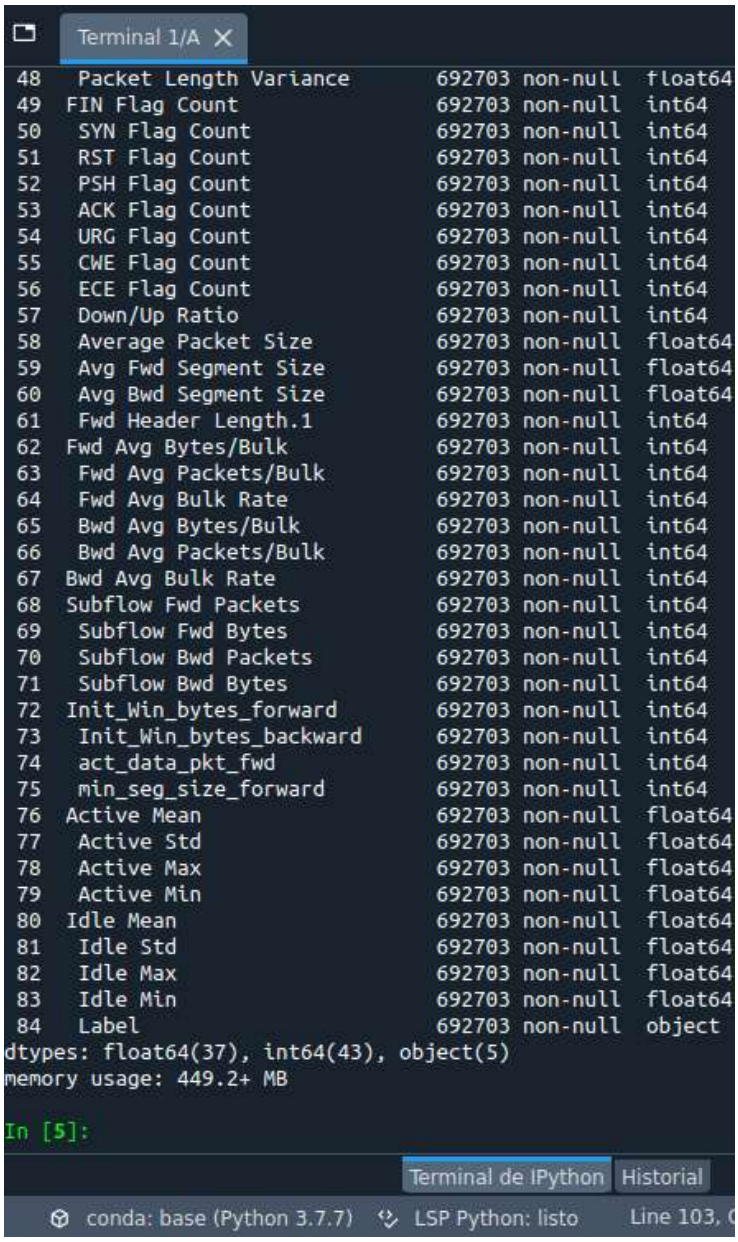


Figura 5-5 Formato de nombres de atributos CIC-IDS-2017 [Elaboración propia]

```
 54   URG Flag Count              692703 non-null  int64
 55   CWE Flag Count              692703 non-null  int64
 56   ECE Flag Count              692703 non-null  int64
 57   Down/Up Ratio               692703 non-null  int64
 58   Average Packet Size         692703 non-null  float64
 59   Avg Fwd Segment Size        692703 non-null  float64
 60   Avg Bwd Segment Size        692703 non-null  float64
 61   Fwd Header Length.1         692703 non-null  int64
 62  Fwd Avg Bytes/Bulk           692703 non-null  int64
 63   Fwd Avg Packets/Bulk        692703 non-null  int64
 64   Fwd Avg Bulk Rate           692703 non-null  int64
 65   Bwd Avg Bytes/Bulk          692703 non-null  int64
 66   Bwd Avg Packets/Bulk        692703 non-null  int64
 67  Bwd Avg Bulk Rate            692703 non-null  int64
 68  Subflow Fwd Packets          692703 non-null  int64
 69   Subflow Fwd Bytes           692703 non-null  int64
 70   Subflow Bwd Packets         692703 non-null  int64
 71   Subflow Bwd Bytes           692703 non-null  int64
 72  Init_Win_bytes_forward       692703 non-null  int64
 73   Init_Win_bytes_backward     692703 non-null  int64
 74   act_data_pkt_fwd            692703 non-null  int64
 75   min_seg_size_forward        692703 non-null  int64
 76  Active Mean                  692703 non-null  float64
 77   Active Std                  692703 non-null  float64
 78   Active Max                  692703 non-null  float64
 79   Active Min                  692703 non-null  float64
 80  Idle Mean                    692703 non-null  float64
 81   Idle Std                    692703 non-null  float64
 82   Idle Max                    692703 non-null  float64
 83   Idle Min                    692703 non-null  float64
 84   Label                       692703 non-null  object
dtypes: float64(37), int64(43), object(5)
memory usage: 449.2+ MB
```

Formato de nombres de atributos CICFlowMeter v3.0 y v4.0

```
pd.read_csv(r'/home/andres/Descargar/Online_ModDet_2021/csv_crudo_84/goldeneye.pcap_Flow.csv')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 819 entries, 0 to 818
Data columns (total 84 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Flow ID           819 non-null    object
 1   Src IP            819 non-null    object
 2   Src Port          819 non-null    int64
 3   Dst IP            819 non-null    object
 4   Dst Port          819 non-null    int64
 5   Protocol          819 non-null    int64
 6   Timestamp         819 non-null    object
 7   Flow Duration     819
 8   Tot Fwd Pkts      819
 9   Tot Bwd Pkts      819
```

10	**TotLen Fwd Pkts**	819
11	**TotLen Bwd Pkts**	819
12	**Fwd Pkt Len Max**	819
13	**Fwd Pkt Len Min**	819
14	**Fwd Pkt Len Mean**	819
15	**Fwd Pkt Len Std**	819
16	**Bwd Pkt Len Max**	819
17	**Bwd Pkt Len Min**	819
18	**Bwd Pkt Len Mean**	819
19	**Bwd Pkt Len Std**	819
20	Flow Byts/s	819
21	Flow Pkts/s	819
22	Flow IAT Mean	819
23	Flow IAT Std	819
24	Flow IAT Max	819
25	Flow IAT Min	819
26	Fwd IAT Tot	819
27	Fwd IAT Mean	819
28	Fwd IAT Std	819
29	Fwd IAT Max	819
30	Fwd IAT Min	819
31	Bwd IAT Tot	819
32	Bwd IAT Mean	819
33	Bwd IAT Std	819
34	Bwd IAT Max	819
35	Bwd IAT Min	819
36	Fwd PSH Flags	819
37	Bwd PSH Flags	819
38	Fwd URG Flags	819
39	Bwd URG Flags	819
40	Fwd Header Len	819
41	Bwd Header Len	819
42	Fwd Pkts/s	819
43	Bwd Pkts/s	819
44	Pkt Len Min	819
45	Pkt Len Max	819
46	Pkt Len Mean	819
47	Pkt Len Std	819
48	Pkt Len Var	819
49	FIN Flag Cnt	819
50	SYN Flag Cnt	819
51	RST Flag Cnt	819
52	PSH Flag Cnt	819
53	ACK Flag Cnt	819
54	URG Flag Cnt	819
55	CWE Flag Count	819
56	ECE Flag Cnt	819
57	Down/Up Ratio	819
58	Pkt Size Avg	819
59	Fwd Seg Size Avg	819
60	Bwd Seg Size Avg	819
61	Fwd Byts/b Avg	819
62	Fwd Pkts/b Avg	819
63	Fwd Blk Rate Avg	819
64	Bwd Byts/b Avg	819
65	Bwd Pkts/b Avg	819

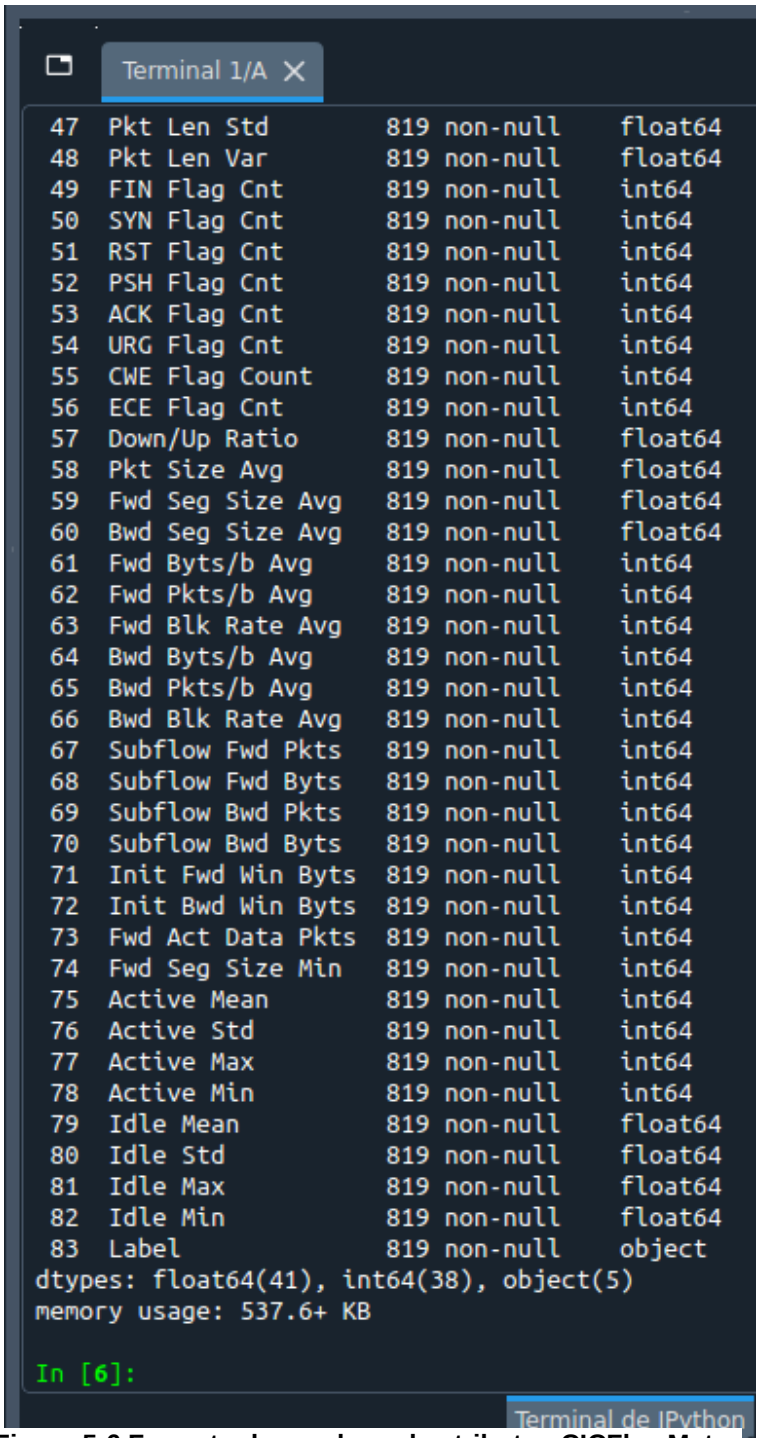


Figura 5-6 Formato de nombres de atributos CICFlowMeter v3.0 y v4.0 [Elaboración propia]

```
 66  Bwd Blk Rate Avg   819 non-null    int64
 67  Subflow Fwd Pkts   819 non-null    int64
 68  Subflow Fwd Byts   819 non-null    int64
 69  Subflow Bwd Pkts   819 non-null    int64
 70  Subflow Bwd Byts   819 non-null    int64
 71  Init Fwd Win Byts  819 non-null    int64
 72  Init Bwd Win Byts  819 non-null    int64
 73  Fwd Act Data Pkts  819 non-null    int64
 74  Fwd Seg Size Min   819 non-null    int64
 75  Active Mean        819 non-null    int64
 76  Active Std         819 non-null    int64
 77  Active Max         819 non-null    int64
 78  Active Min         819 non-null    int64
 79  Idle Mean          819 non-null    float64
 80  Idle Std           819 non-null    float64
 81  Idle Max           819 non-null    float64
 82  Idle Min           819 non-null    float64
 83  Label              819 non-null    object
dtypes: float64(41), int64(38), object(5)
memory usage: 537.6+ KB
```

Formato estándar que se ha aplicado al procesamiento 37 atributos

```
estandar        =        pd.read_csv(r'/media/andres/X360/CIC_MIX_JAPJ_2019-2018-
2017/X_100JAPJ_test.csv')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1359628 entries, 0 to 1359627
Data columns (total 37 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   dst_port           1359628 non-null  int64
 1   protocol           1359628 non-null  int64
 2   flow_duration      1359628 non-null  float64
 3   tot_fwd_pkts       1359628 non-null  int64
 4   tot_bwd_pkts       1359628 non-null  int64
 5   totlen_fwd_pkts    1359628 non-null  int64
 6   totlen_bwd_pkts    1359628 non-null  int64
 7   fwd_pkt_len_min    1359628 non-null  int64
 8   fwd_pkt_len_mean   1359628 non-null  float64
 9   fwd_pkt_len_std    1359628 non-null  float64
 10  bwd_pkt_len_min    1359628 non-null  int64
 11  bwd_pkt_len_std    1359628 non-null  float64
 12  flow_byts_s        1359628 non-null  float64
 13  flow_pkts_s        1359628 non-null  float64
```

```
 14  flow_iat_std        1359628
 15  flow_iat_min        1359628
 16  fwd_iat_tot         1359628
 17  fwd_iat_min         1359628
 18  bwd_iat_tot         1359628
 19  fwd_psh_flags       1359628
 20  fwd_header_len      1359628
 21  bwd_pkts_s          1359628
 22  pkt_len_max         1359628
 23  pkt_len_std         1359628
 24  fin_flag_cnt        1359628
 25  rst_flag_cnt        1359628
 26  psh_flag_cnt        1359628
 27  ack_flag_cnt        1359628
 28  urg_flag_cnt        1359628
 29  cwe_flag_count      1359628
 30  init_fwd_win_byts   1359628
 31  init_bwd_win_byts   1359628
 32  fwd_act_data_pkts   1359628
 33  fwd_seg_size_min    1359628
 34  active_mean         1359628
 35  active_std          1359628
 36  idle_mean           1359628
dtypes: float64(17), int64(20)
memory usage: 383.8 MB
```

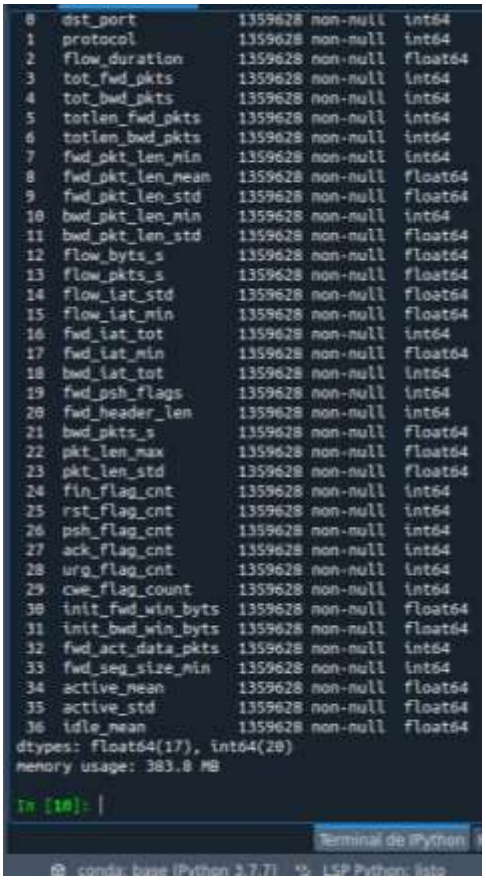


Figura 5-7 Formato estándar aplicado a los 37 atributos [Elaboración propia]

Como se pudo observar en las cinco figuras Figura 5-3, Figura 5-4, Figura 5-5, Figura 5-6, Figura 5-7 además de los cambios en los nombres de los atributos, también presentan para cada año un tamaño de atributos diferente, a saber (sin el *label* /Objetivo): 83, 87 y 84 atributos para 2018 ó Herramienta CICFlowMeter, 2019 y 2017 respectivamente.

También es importante, la diferencia de tipos de datos de cada atributo que presenta también cada año.

5.1.2. Segundo Preprocesamiento: a la concatenación

En este paso se obtiene todo el conjunto de datos concatenado en un único y total archivo de variables separadas por comas (.csv), se expresa como un dataframe de pandas completo, con un total de **13'596.273** de registros de **flujos de tráfico de red** como se aprecia en la Figura 5-8.

Figura 5-8 Dataframe unificado combinación específica CIC-IDS2017, CIC-IDS2018 & CIC-DDoS2019 [Elaboración propia]

Se ha proseguido a tener por separado, la matriz de atributos **X** de su del vector objetivo **y** con sus 3 etiquetas: ***label*** (nombre ataque), ***etiqueta_cat*** (codificación), ***etiqueta_binaria*** (codificación binaria); ver Figura 5-9.

Figura 5-9 Conjuntos Entrenamiento y Prueba combinación específica CIC-2017-2018-2019 [Elaboración propia]

5.2. Procesamiento: reducción dimensionalidad

5.2.1. Ingeniería de atributos: Selección de atributos

En esta fase, se ha realizado en primera medida, la compresión de **78 atributos** a **69** atributos como se puede apreciar en la Figura 5-10, gracias a la **desviación estándar**. Es decir, de los 83 atributos en crudo, ya sabemos que eliminamos 5 atributos que son genéricos e independientes del aprendizaje, pues

debe ser indiferentes a esa identificación del flujo en el proceso de entrenamiento, estos son la 5-tupla: **`Flow ID`**, **`Src IP`**, **`Src Port`**, **`Dst IP`** y **`Timestamp.`**

Figura 5-10 Primera selección de atributos: desviación estándar [Elaboración propia]

```
<class 'pandas.core.frame.DataFrame'>
Data columns (total 68 columns):
 0   protocol
 1   flow_duration
 2   tot_fwd_pkts
 3   tot_bwd_pkts
 4   totlen_fwd_pkts
 5   totlen_bwd_pkts
 6   fwd_pkt_len_max
 7   fwd_pkt_len_min
 8   fwd_pkt_len_mean
 9   fwd_pkt_len_std
 10  bwd_pkt_len_max
 11  bwd_pkt_len_min
 12  bwd_pkt_len_mean
 13  bwd_pkt_len_std
 14  flow_byts_s
 15  flow_pkts_s
 16  flow_iat_mean
 17  flow_iat_std
 18  flow_iat_max
 19  flow_iat_min
 20  fwd_iat_tot
 21  fwd_iat_mean
 22  fwd_iat_std
 23  fwd_iat_max
 24  fwd_iat_min
 25  bwd_iat_tot
 26  bwd_iat_mean
 27  bwd_iat_std
 28  bwd_iat_max
 29  bwd_iat_min
 30  fwd_psh_flags
 31  fwd_header_len
```

```
32  bwd_header_len                  52  subflow_fwd_pkts
33  fwd_pkts_s                      53  subflow_fwd_byts
34  bwd_pkts_s                      54  subflow_bwd_pkts
35  pkt_len_min                     55  subflow_bwd_byts
36  pkt_len_max                     56  init_fwd_win_byts
37  pkt_len_mean                    57  init_bwd_win_byts
38  pkt_len_std                     58  fwd_act_data_pkts
39  pkt_len_var                     59  fwd_seg_size_min
40  fin_flag_cnt                    60  active_mean
41  syn_flag_cnt                    61  active_std
42  rst_flag_cnt                    62  active_max
43  psh_flag_cnt                    63  active_min
44  ack_flag_cnt                    64  idle_mean
45  urg_flag_cnt                    65  idle_std
46  cwe_flag_count                  66  idle_max
47  ece_flag_cnt                    67  idle_min
48  down_up_ratio                   dtypes:     float32(22),     float64(4),
49  pkt_size_avg                    int64(6),     uint16(7),     uint32(16),
50  fwd_seg_size_avg                uint64(2), uint8(11)
51  bwd_seg_size_avg                memory usage: 4.0 GB
```

Como se observa en la Figura 5-10 los nueve atributos que se han eliminado cuya desviación estándar es nula son: **Bwd_PSH_Flags**, **Fwd_URG_Flags**, **Bwd_URG_Flags**, **Fwd_Byts/b_Avg**, **Fwd_Pkts/b_Avg**, **Fwd_Blk_Rate_Avg**, **Bwd_Byts/b_Avg**, **Bwd_Pkts/b_Avg** y **Bwd_Blk_Rate_Avg**. Estos 9 atributos sin varianza, se ven gráficamente representados con color blanco en la gráfica de mapa de calor de la Figura 5-12, donde su relación es redundante y solo aporta ruido, por ello se eliminan.

Como segundo paso, con una matemática más rigurosa, y usando el ecosistema científico de SciPy, específicamente la correlación jerárquica del ranking de Spearman, se **redujo en 32 atributos**, de 69 a 37 atributos finales como se puede observar en la Figura 5-11. Para una compresión y reducción considerable de atributos, y lo mejor no es sólo que sea menos de la mitad, **46 atributos total eliminados** de 83 a 37. Es el hecho qué en esa reducción de 46 atributos, sigue estando representando fidedignamente toda la información de los 83 atributos completos.

Figura 5-11 Segunda selección de atributos: correlación jerárquica de Spearman [Elaboración propia]

0 dst_port	19 fwd_psh_flags
1 protocol	20 fwd_header_len
2 flow_duration	21 bwd_pkts_s
3 tot_fwd_pkts	22 pkt_len_max
4 tot_bwd_pkts	23 pkt_len_std
5 totlen_fwd_pkts	24 fin_flag_cnt
6 totlen_bwd_pkts	25 rst_flag_cnt
7 fwd_pkt_len_min	26 psh_flag_cnt
8 fwd_pkt_len_mean	27 ack_flag_cnt
9 fwd_pkt_len_std	28 urg_flag_cnt
10 bwd_pkt_len_min	29 cwe_flag_count
11 bwd_pkt_len_std	30 init_fwd_win_byts
12 flow_byts_s	31 init_bwd_win_byts
13 flow_pkts_s	32 fwd_act_data_pkts
14 flow_iat_std	33 fwd_seg_size_min
15 flow_iat_min	34 active_mean
16 fwd_iat_tot	35 active_std
17 fwd_iat_min	36 idle_mean
18 bwd_iat_tot	

Se ha realizado un archivo en excel (xlsx) donde se ha plasmado un resumen de este procesamiento plasmando los resultados, junto a una próxima fase detallado de la importancia de los atributos de cinco algoritmos de aprendizaje pertenecientes al pódium elegido. Se presentará en la Tabla 5-4.

5.2.2. Mapa de calor y dendrograma

En este apartado se ha realizado tres representaciones gráficas concernientes a mapas de calor y dendrograma de atributos.

El mapa de calor es una forma de representar los datos en un formato bidimensional. Los valores de los datos son representados como colores en el gráfico. La meta del mapa de calor es proporcionar un resumen visual a color de la información. Para nuestro caso, el color **rojo** representa la relación de **+1,0** , mientras el **azul** representa el caso extremo opuesto **-1,0** , y el color **blanco** una relación nula de **0,0**. El **Rojo intenso** avisa de los sitios con **más calor**, como se aprecian en la diagonal principal. Hasta un **azul claro** o **zona fría**, avisa para revelar aquellas relaciones de atributos menos concurridos o relacionados.

Ahora, los resultados del *hierarchical clustering* pueden representarse como un árbol en el que las ramas representan la jerarquía con la que se van sucediendo las uniones de *clusters*. En la base del dendrograma, cada observación forma una terminación individual conocida como **hoja** o *leaf* del árbol. A medida que se asciende por la estructura, pares de hojas se fusionan formando las primeras **ramas**. Estas uniones se corresponden con los pares de observaciones más similares. También ocurre que las ramas se fusionan con otras ramas o con hojas. Cuanto más temprana (más próxima a la base del dendrograma) ocurre una fusión, mayor es la similitud. Para cualquier par de observaciones, se puede identificar el punto del árbol en el que las ramas que contienen dichas observaciones se fusionan. La **altura** a la que esto ocurre (**eje vertical**) indica cómo de

similares/diferentes son las dos observaciones. Los dendrogramas, por lo tanto, se deben interpretar únicamente en base al eje vertical

Dicho esto, y ya explicado el primer mapa de calor de la Figura 5-12 de los 78 atributos en la sección anterior 5.2.1; proseguimos a las dos siguientes.

La Figura 5-13 representa el mapa de calor y el dendrograma de 68 Atributos luego de aplicar la eliminación de los 9 atributos de desviación estándar, se puede concluir por ejemplo que atributos como **cwe_flag_count** presenta unos colores muy cercanos a cero 0. Otra conclusión, encontramos en la esquina inferior izquierda, se forma un cuadrado de 8 x 8, altamente en rojo, esto significa que los 8 atributos **active_mean**, **active_std**, **active_max**, **active_mean**, **idle_mean**, **idle_std**, **idle_max** y **idle_min** están altamente relaciones. Y el dendrograma realizado de forma no supervisada, sin ver las etiquetas, aplicar conceptos de agrupamiento. Se puede expresar interpretando dicho dendrograma que existen 17 atributos de muy baja altura, en la mínima base, nombrado de izquierda a derecha: **totlen_fwd_pkts** & **subflow_fwd_byts**; **totlen_bwd_pkts** & **subflow_bwd_byts**; **tot_bwd_pkts** & **subflow_bwd_pkts**; **bwd_iat_tot** & **bwd_iat_mean** & **bwd_iat_max**; **idle_min** & **idle_mean** & **idle_max**; **active_min** & **active_mean** & **active_max**; **tot_fwd_pkts** & **subflow_fwd_pkts**. Ahora los de mayor altura, o mayor valor del eje y, son en su orden descendente, por ejemplo: **ack_flag_cnt**, **init_fwd_win_bytes**, **flow_iat_min**, **dst_port**, **init_bwd_win_byts**, **bwd_pkt_len_min**, **fwd_act_data_pkts**, **fwd_pkt_len_std**, **fwd_header_len**, **bwd_iat_min.**

Por último, la Figura 5-14 representa el mapa de calor final, y se presenta más colorido(rojo), y más azul, siendo las zonas muy tenues cercanas a blanco (cero), solo **cwe_flag_count** por ejemplo**.**

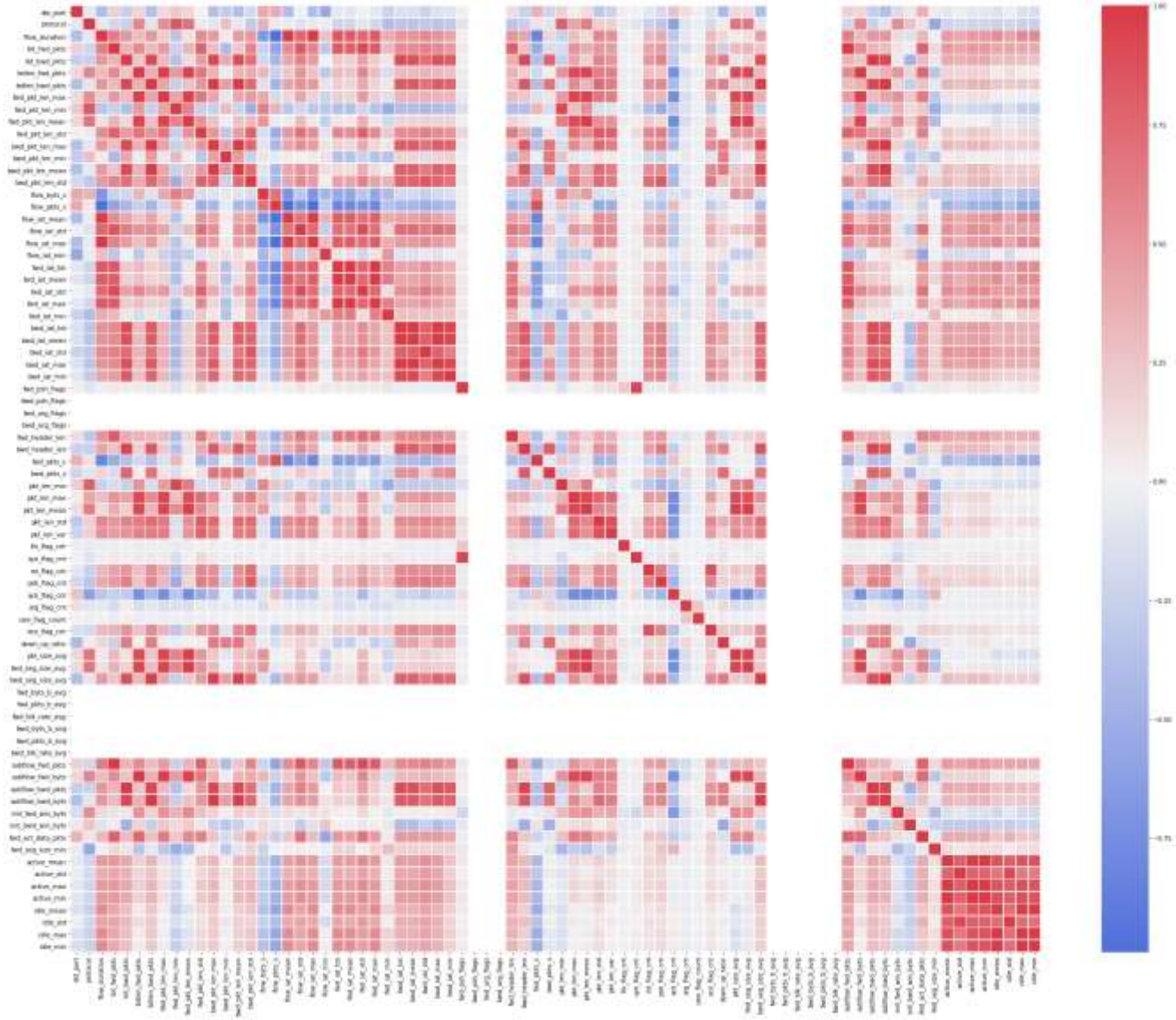


Figura 5-12 Mapa de Calor de los 78 Atributos iniciales [Elaboración propia]

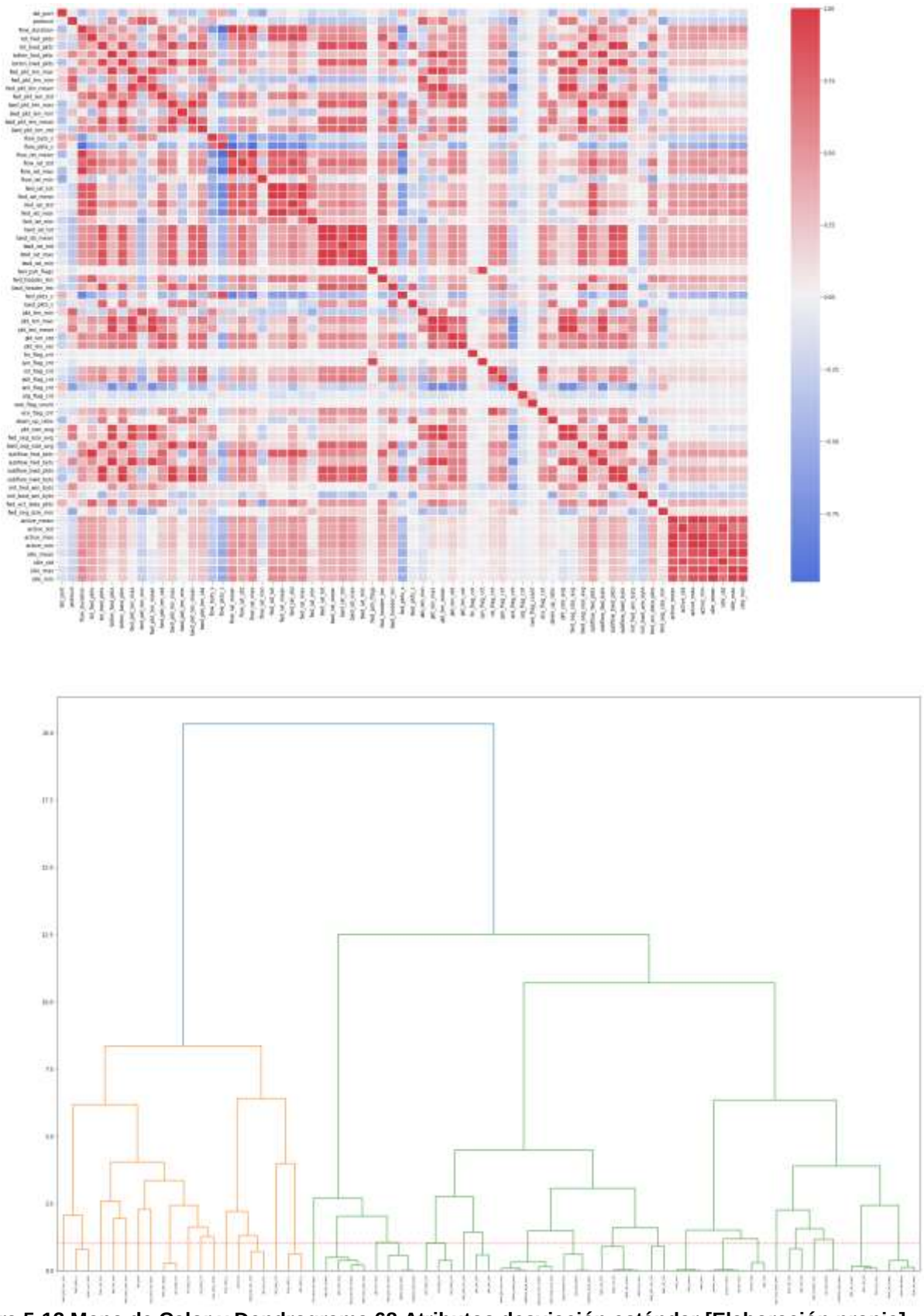


Figura 5-13 Mapa de Calor y Dendrograma 68 Atributos desviación estándar [Elaboración propia]

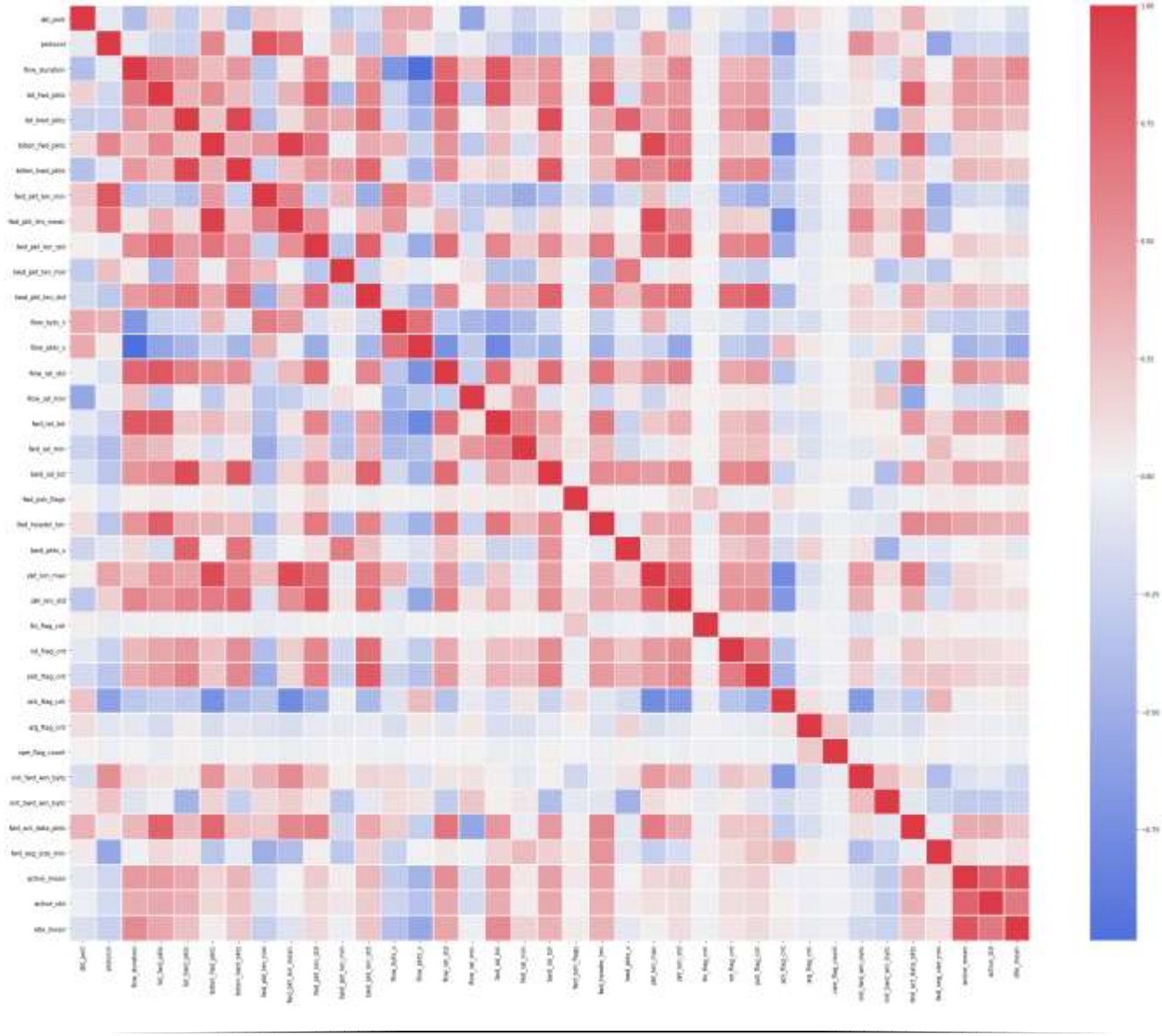


Figura 5-14 Mapa de Calor de los 37 Atributos definitivos: correlación jerárquica de Spearman [Elaboración propia]

5.2.3. Histograma y gráficos de densidad de los 37 atributos finales

Para analizar los datos y representar visualmente de manera efectiva los 37 atributos finales se ha optado por aplicar los gráficos de densidad que son análogos a los gráficos de histograma, pero con una curva suave dibujada en la parte superior de cada intervalo. Los gráficos de densidad no se ven afectados por el número de *bins*, que es un parámetro importante cuando se trata de histogramas, por lo que nos permite visualizar mejor la distribución de nuestros datos.

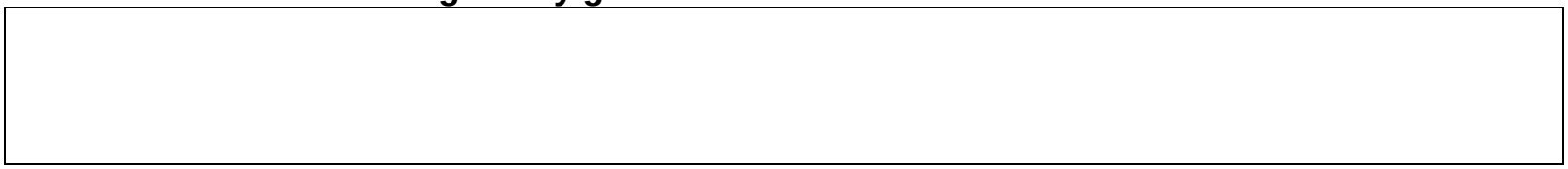
Así visualizamos la forma de la distribución de los 37 atributos finales de tráfico de red contemplados en la Tabla 5-1, con 37 figuras cuyas nueve principales atributos se muestran en amarillo, dorado, y se saben por análisis y resultados posteriores que se plasman en la Tabla 5-4 y sección 5.3.4

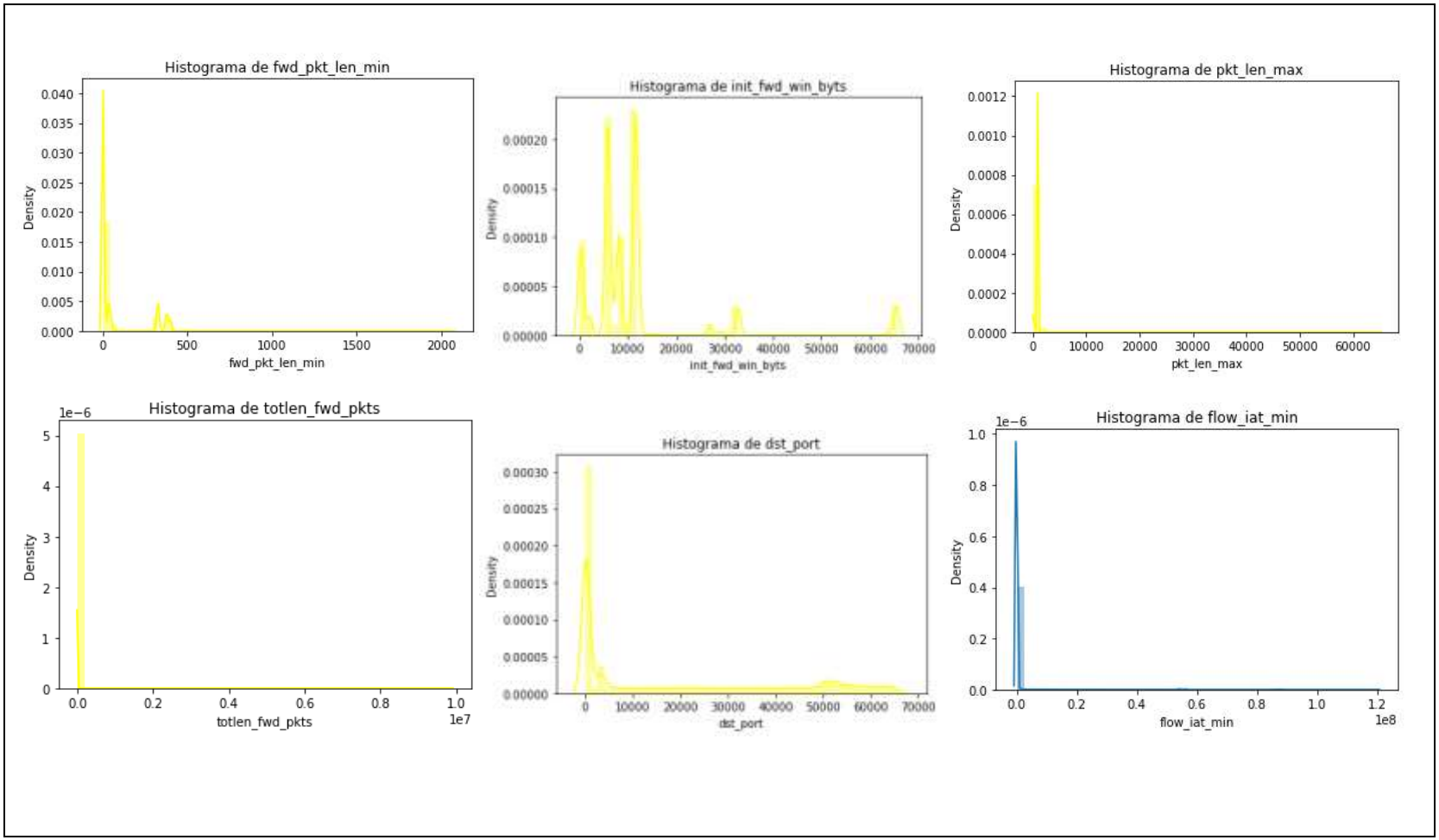
Para interpretar correctamente estos gráficos de densidad al ser una versión continua y suavizada de un histograma estimado a partir de los datos. Y cuya forma más común de estimación se conoce como estimación de la densidad del núcleo (kernel) El -eje x- de todas las figuras de la Tabla 5-1 es el valor de la variable o atributo, como en un histograma, y en el -eje y- es la función de densidad de probabilidad para la estimación de la densidad del núcleo. Sin embargo, debemos tener cuidado de especificar que se trata de una **densidad de probabilidad** y no de una probabilidad. La diferencia es que la densidad de probabilidad es la probabilidad por unidad en el -eje x-. Para convertirla en una probabilidad real, tenemos que encontrar el área bajo la curva para un intervalo específico en el -eje x-. Como se trata de una densidad de probabilidad y no de una probabilidad, el eje y puede tomar valores mayores

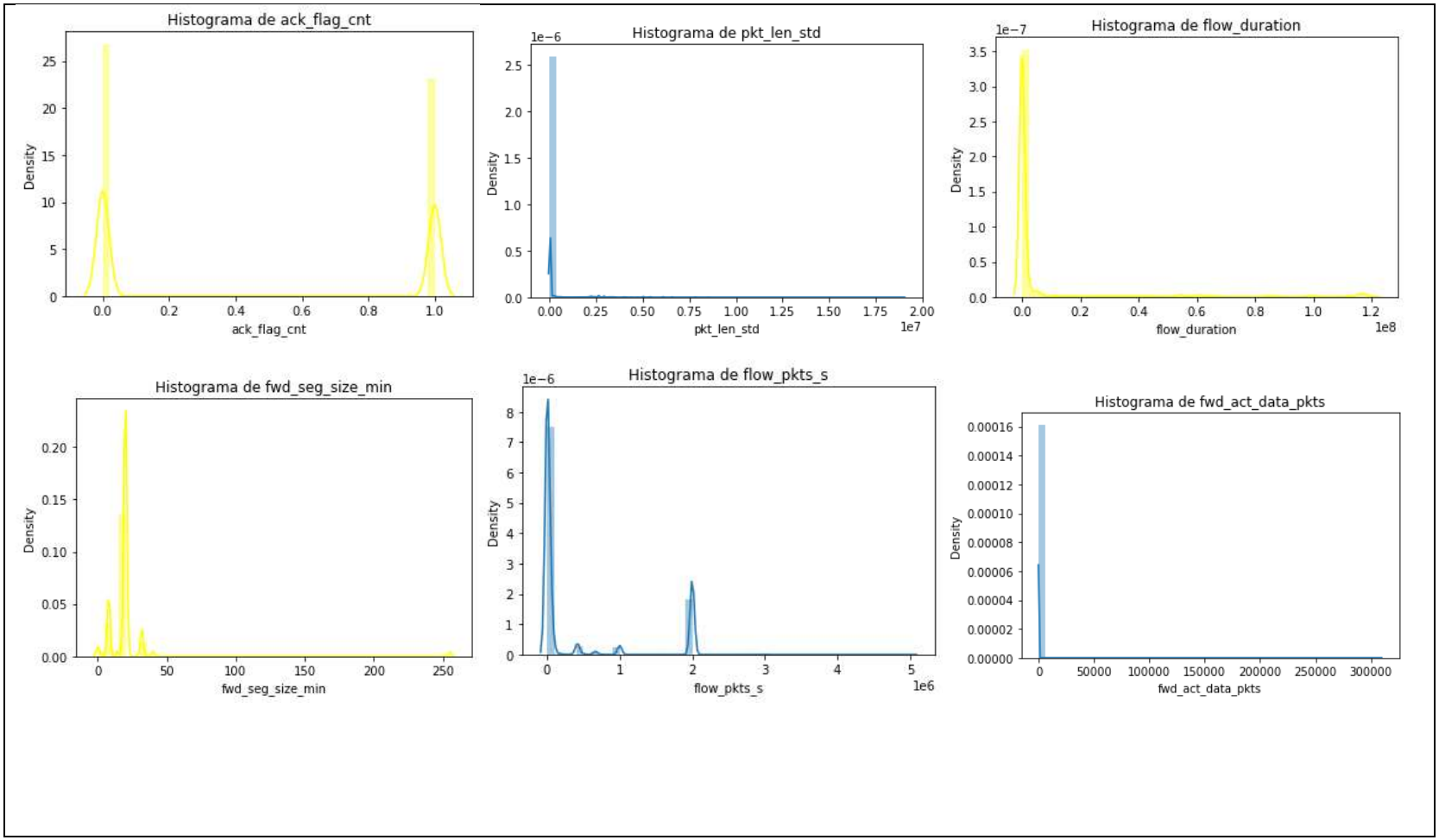
que uno. El único requisito del gráfico de densidad es que el área total bajo la curva se **integre en uno**. En conclusión, se debe tender a pensar en el -eje y- de un gráfico de densidad como un valor sólo para las **comparaciones relativas** entre diferentes categorías de familia de malware DoS y DDoS.

Para nuestro caso de análisis de atributos se ha obtenido cuatro atributos de forma de distribución bimodal: **ack_flag_cnt**, **protocol**, **psh_flag_cnt**, **rst_flag_cnt**; veintitrés atributos de forma de distribución desigual (*skewed*): **pkt_len_max**, **totlen_fwd_pkts**, **flow_iat_min**, **pkt_len_std**, **fwd_act_data_pkts**, **totlen_bwd_pkts**, **fwd_iat_min**, **tot_bwd_pkts**, **fwd_header_len**, **fwd_pkt_len_std**, **fwd_iat_tot**, **bwd_pkt_len_std**, **tot_fwd_pkts**, **flow_iat_std**, **bwd_iat_tot**, **urg_flag_cnt**, **bwd_pkt_len_min**, **active_mean**, **active_std**, **fwd_psh_flags**, **fin_flag_cnt**, **cwe_flag_count**, **fwd_pkt_len_mean**; cuatro atributos desiguales, más otro pico extra de frecuencia o conteo: **fwd_pkt_len_min**, **flow_byts_s**, **bwd_pkts_s**, **idle_mean**; cuatro atributos con muchas picos de frecuencia: **init_fwd_win_byts**, **fwd_seg_size_min**, **flow_pkts_s**, **init_bwd_win_byts**; y por ultimo dos que son atributos distribuidos de forma desigual, pero con una frecuencia mínima para las demás muestras de datos: **dst_port**, **flow_duration**.

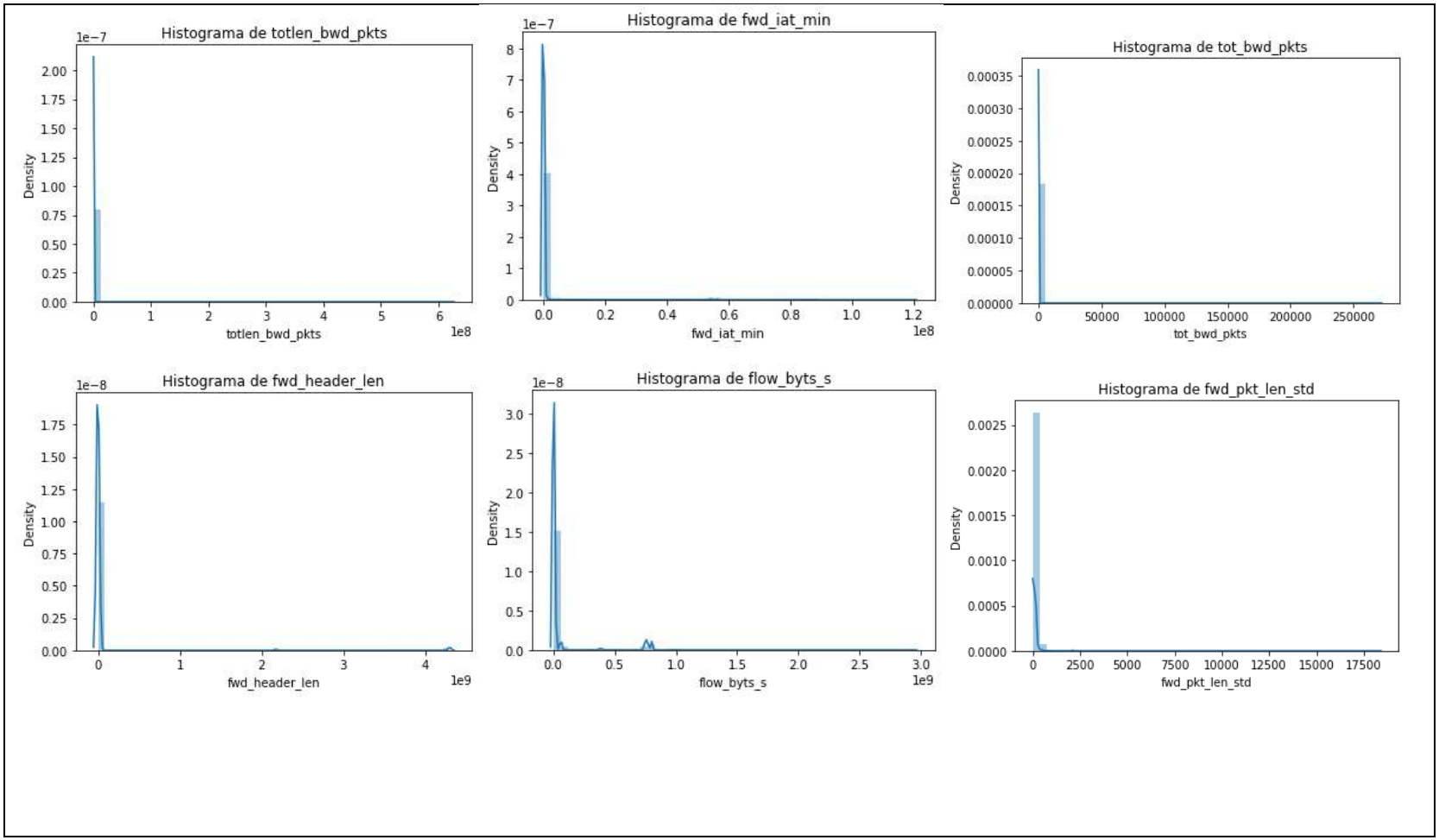
Tabla 5-1 Histograma y gráficos de Densidad de los 37 atributos finales

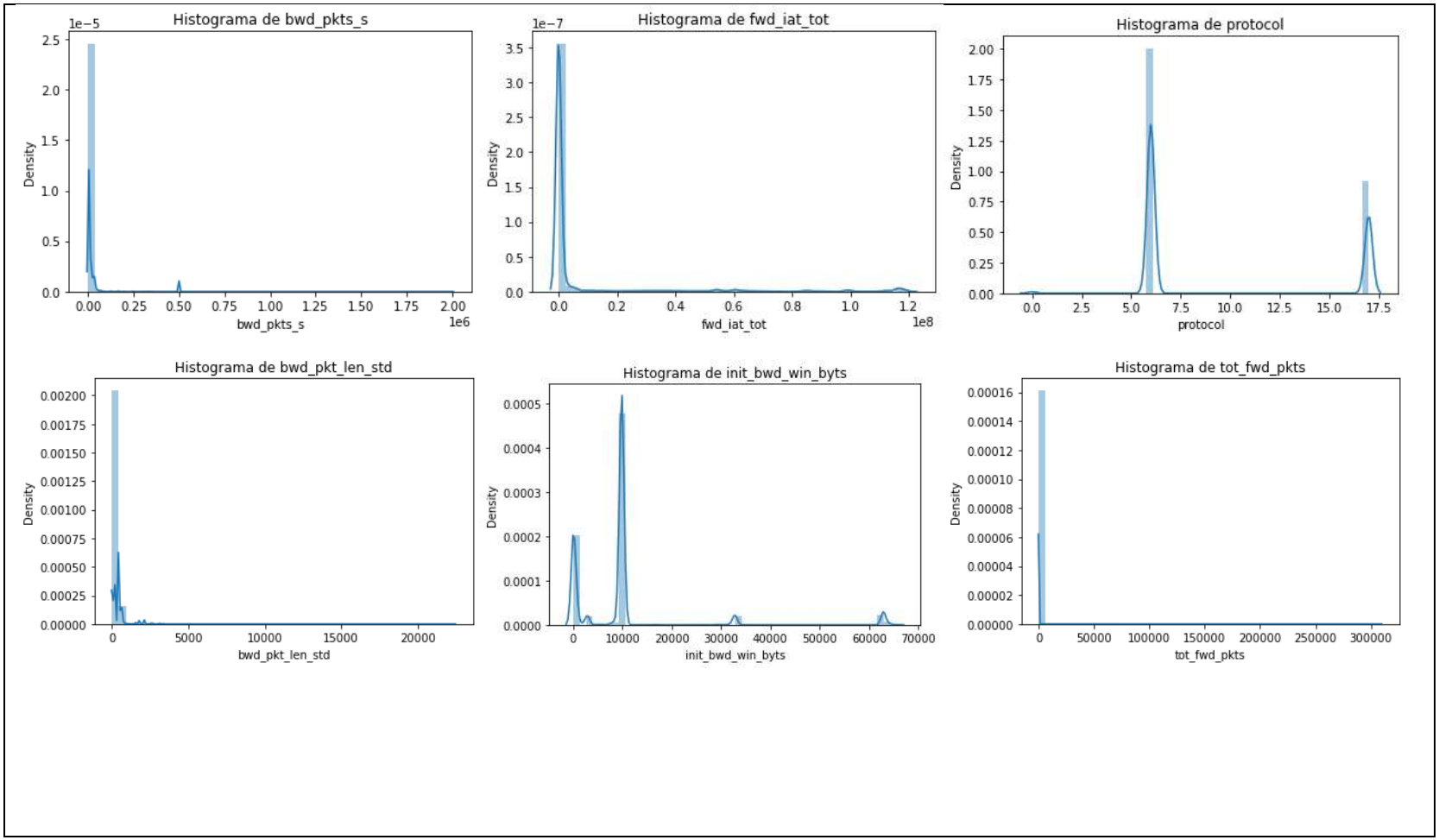



Histograma de fwd_pkt_len_min
Density
fwd_pkt_len_min
Histograma de init_fwd_win_byts
Density
init_fwd_win_byts
Histograma de pkt_len_max
Density
pkt_len_max
Histograma de totlen_fwd_pkts
1e−6
Density
totlen_fwd_pkts
1e7
Histograma de dst_port
Density
dst_port
Histograma de flow_iat_min
1e−6
Density
flow_iat_min
1e8

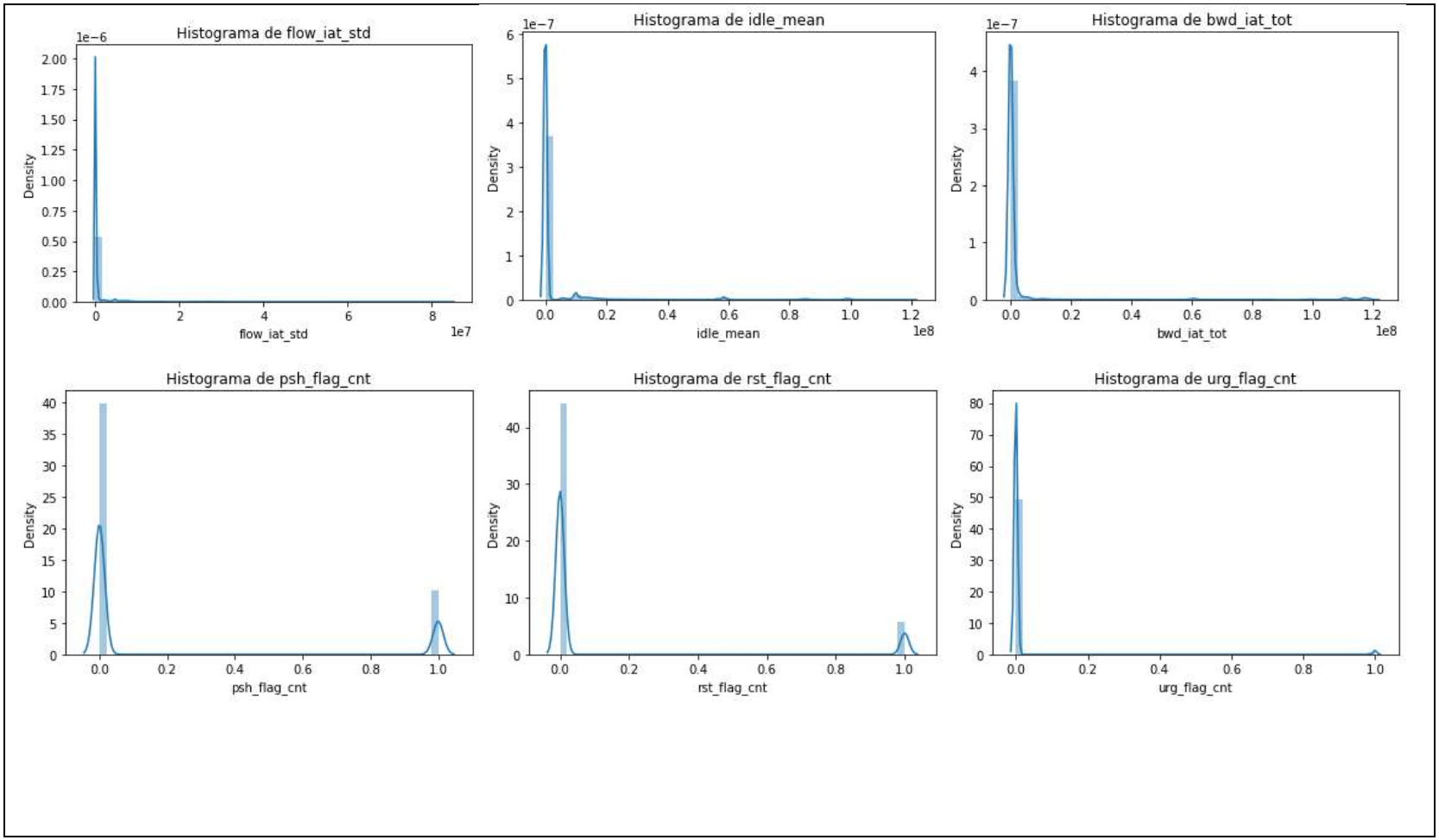

Histograma de ack_flag_cnt
Density
ack_flag_cnt
Histograma de pkt_len_std
1e−6
Density
pkt_len_std
1e7
Histograma de flow_duration
1e−7
Density
flow_duration
1e8
Histograma de fwd_seg_size_min
Density
fwd_seg_size_min
Histograma de flow_pkts_s
1e−6
Density
flow_pkts_s
1e6
Histograma de fwd_act_data_pkts
Density
fwd_act_data_pkts

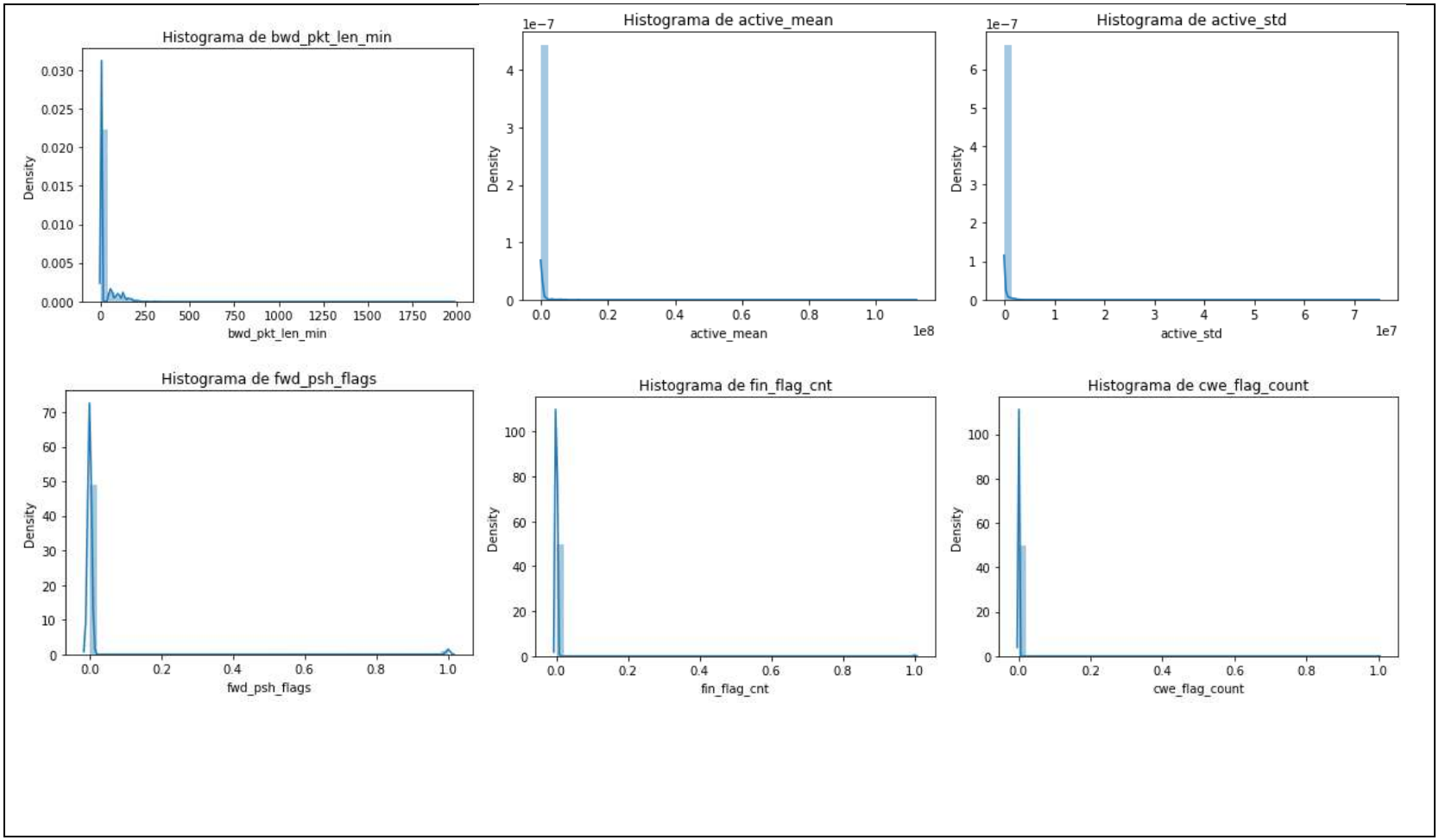
CAPÍTULO 5.2.3. HISTOGRAMA Y GRÁFICOS DE DENSIDAD DE LOS 37 ATRIBUTOS FINALES

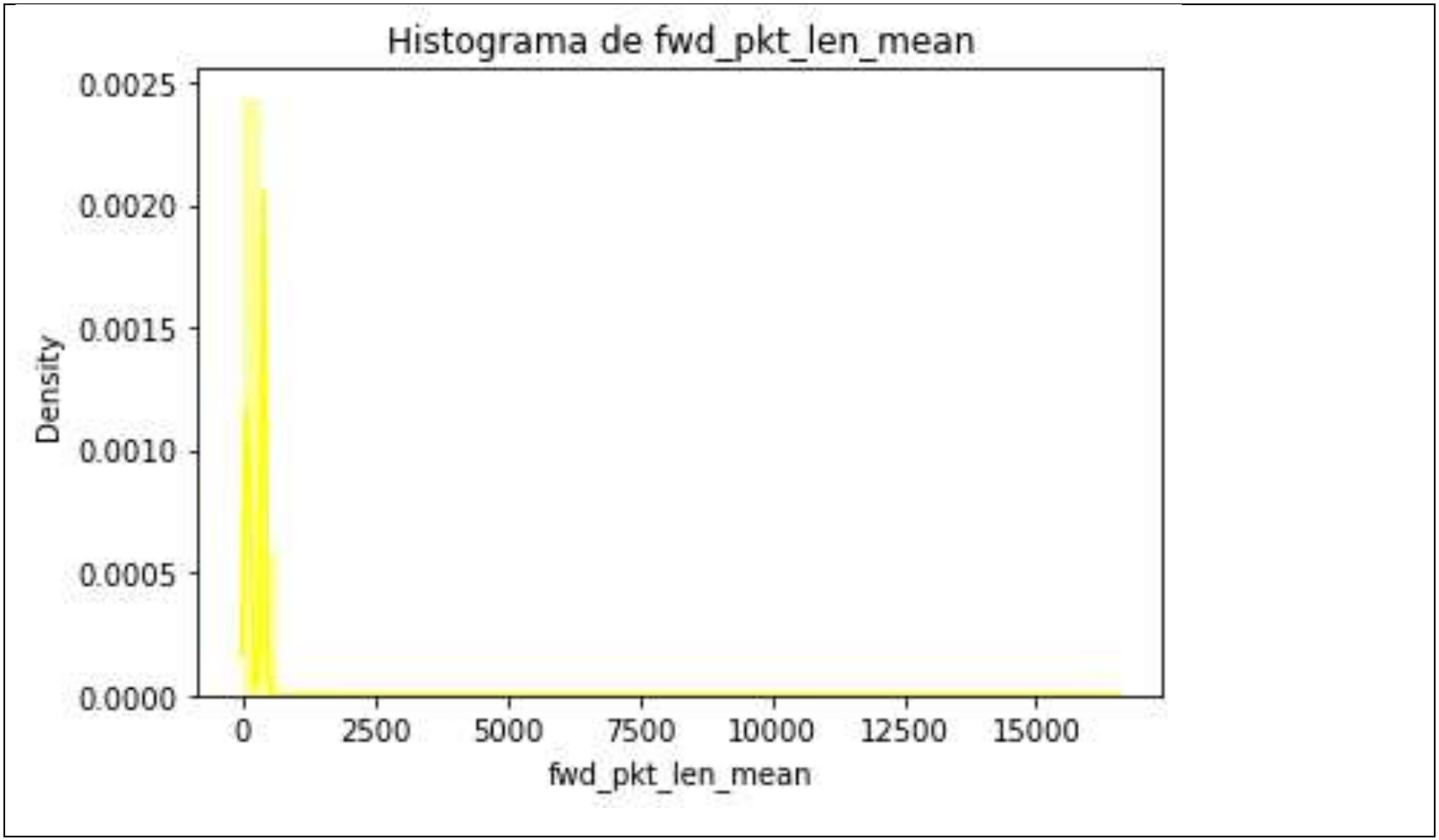



Histograma de bwd_pkts_s
1e−5
Density
bwd_pkts_s
1e6
Histograma de fwd_iat_tot
1e−7
Density
fwd_iat_tot
1e8
Histograma de protocol
Density
protocol
Histograma de bwd_pkt_len_std
Density
bwd_pkt_len_std
Histograma de init_bwd_win_byts
Density
init_bwd_win_byts
Histograma de tot_fwd_pkts
Density
tot_fwd_pkts

CAPÍTULO 5.2.3. HISTOGRAMA Y GRÁFICOS DE DENSIDAD DE LOS 37 ATRIBUTOS FINALES




Histograma de bwd_pkt_len_min
Density
bwd_pkt_len_min
Histograma de active_mean
1e−7
Density
active_mean
1e8
Histograma de active_std
1e−7
Density
active_std
1e7
Histograma de fwd_psh_flags
Density
fwd_psh_flags
Histograma de fin_flag_cnt
Density
fin_flag_cnt
Histograma de cwe_flag_count
Density
cwe_flag_count


Histograma de fwd_pkt_len_mean
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
2500
5000
7500
10000
12500
15000
fwd_pkt_len_mean

5.2.4. Gráficos de dispersión de los atributos importantes

El diagrama de dispersión es un gráfico en el que los valores de dos atributos ó variables se trazan a lo largo de los dos ejes cartesianos. Es un tipo de gráfico muy básico que ayuda a visualizar la relación entre dos variables.

Si el valor a lo largo del -eje y- parece aumentar a medida que el -eje x- aumenta (o disminuye), podría indicar una relación lineal positiva (o negativa). En cambio, si los puntos se distribuyen de forma aleatoria y sin un patrón evidente, podría indicar una falta de relación de dependencia.

Para nuestro caso, se ha decidido por la creación de tres gráficos de dispersión, comparando entre si los tres atributos que posteriormente sabremos que son los más importantes por el descernimiento de los 6 algoritmos de aprendizaje de clasificación del pódium: **`init_fwd_win_byts`**, **`fwd_pkt_len_min`** y **`fwd_pkt_len_mean`**.

En la Figura 5-15, se compara la relación entre los atributos **`init_fwd_win_byts`** versus **`fwd_pkt_len_min`**, y mediante la especificación por el parámetro ***hue***, se diferencia las clases de ataques DoS y DDoS.

En la Figura 5-16, se compara la relación entre los atributos **`init_fwd_win_byts`** versus **`fwd_pkt_len_min`**.

En la Figura 5-17, se compara la relación entre los atributos **`fwd_pkt_len_min`** versus **`fwd_pkt_len_mean.`**

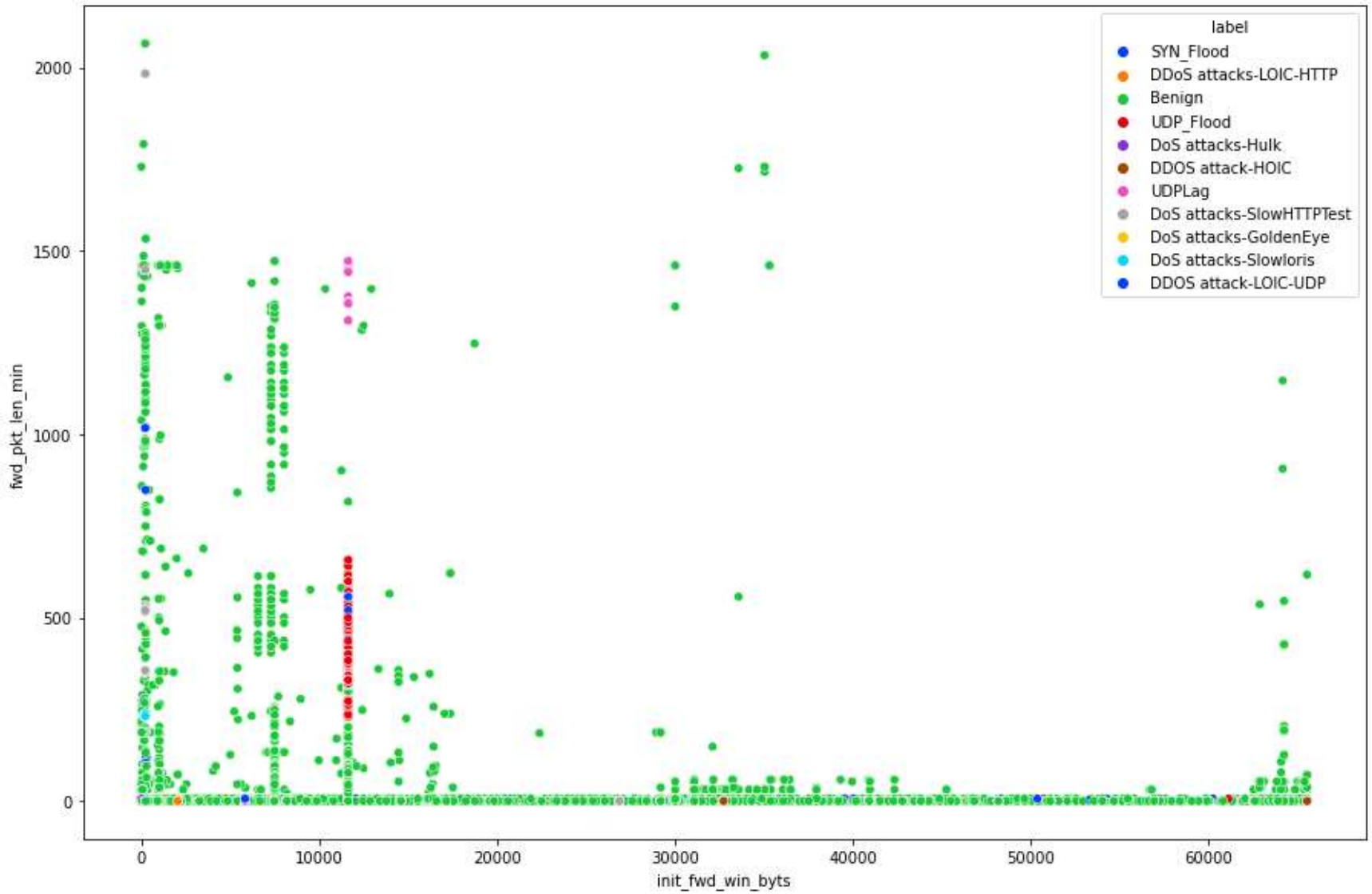


Figura 5-15 Gráfico Dispersión FWD_PKT_LEN_MIN vs INIT_FWD_WIN_BYTS por Clases ataques Maliciosos [Elaboración propia]

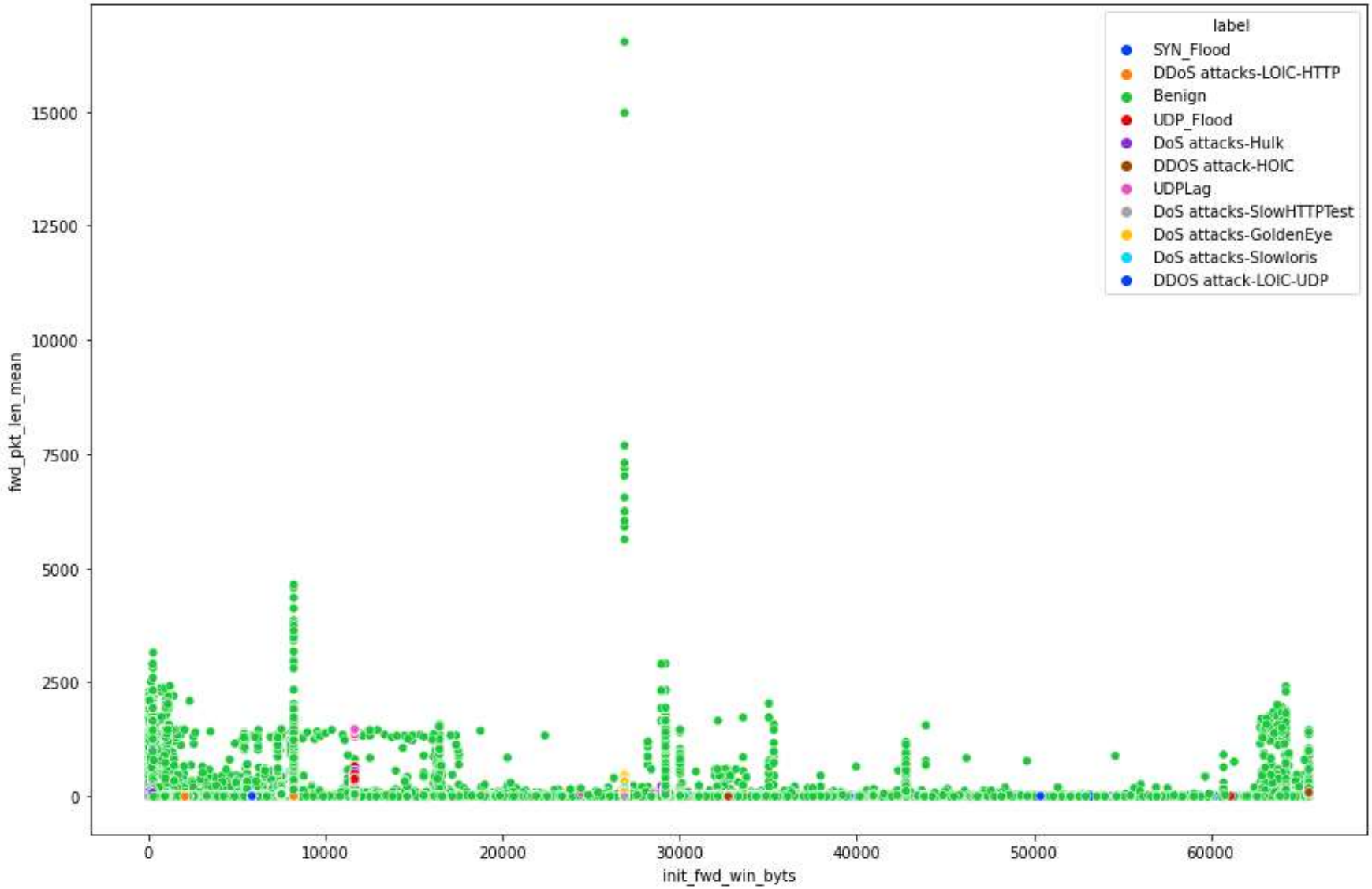


Figura 5-16 Gráfico Dispersión FWD_PKT_LEN_MEAN vs INIT_FWD_WIN_BYTS por Clases ataques Maliciosos [Elaboración propia]

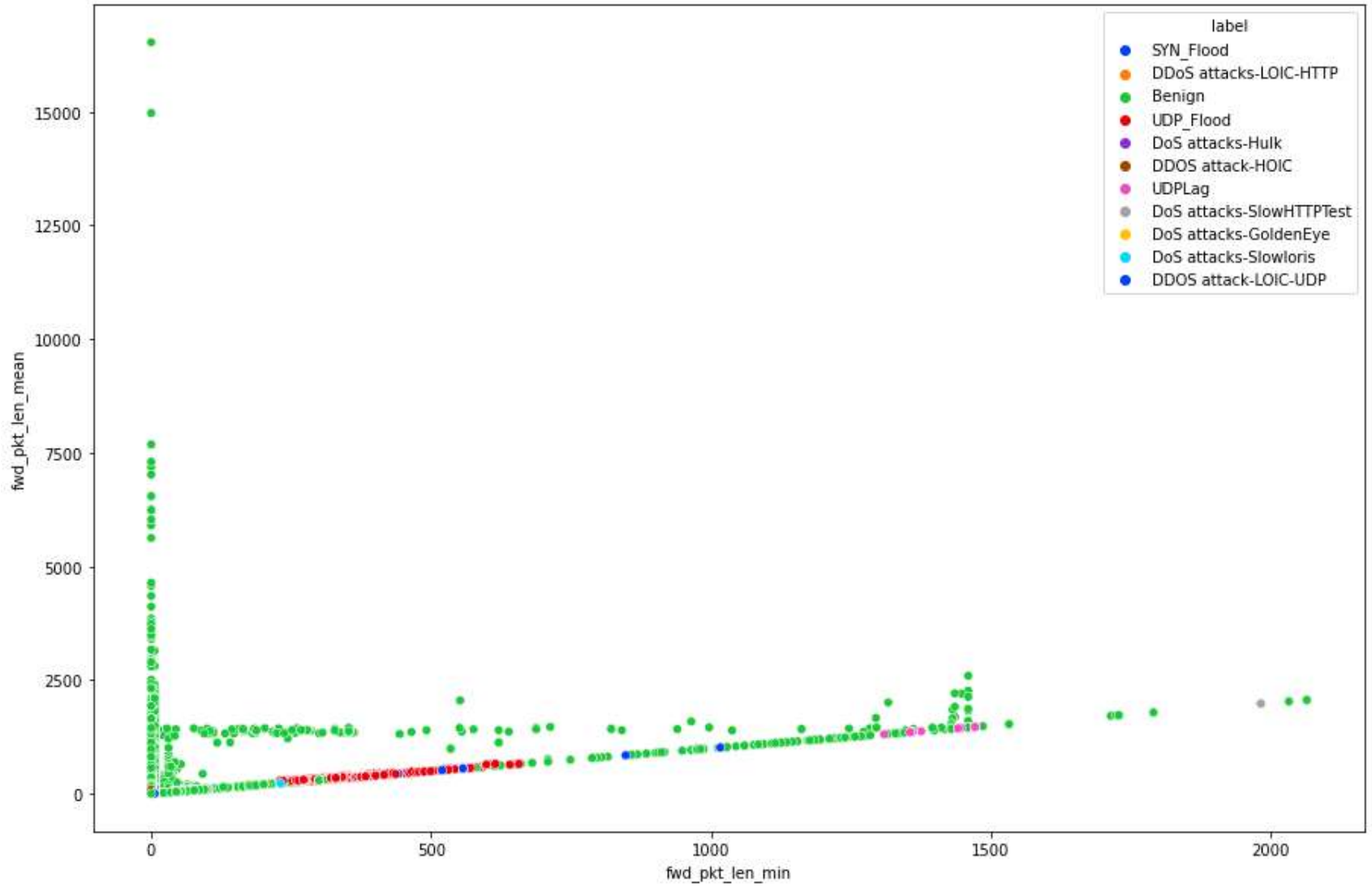


Figura 5-17 Gráfico Dispersión FWD_PKT_LEN_MEAN vs FWD_PKT_LEN_MIN por Clases ataques Maliciosos [Elaboración propia]

5.2.5. División en conjunto de entrenamiento y conjunto de prueba

Se obtiene los 4 subconjuntos: $\mathbf{X_{entrena}}$, $\mathbf{y_{entrena}}$ junto con $\mathbf{X_{prueba}}$, $\mathbf{y_{prueba}}$

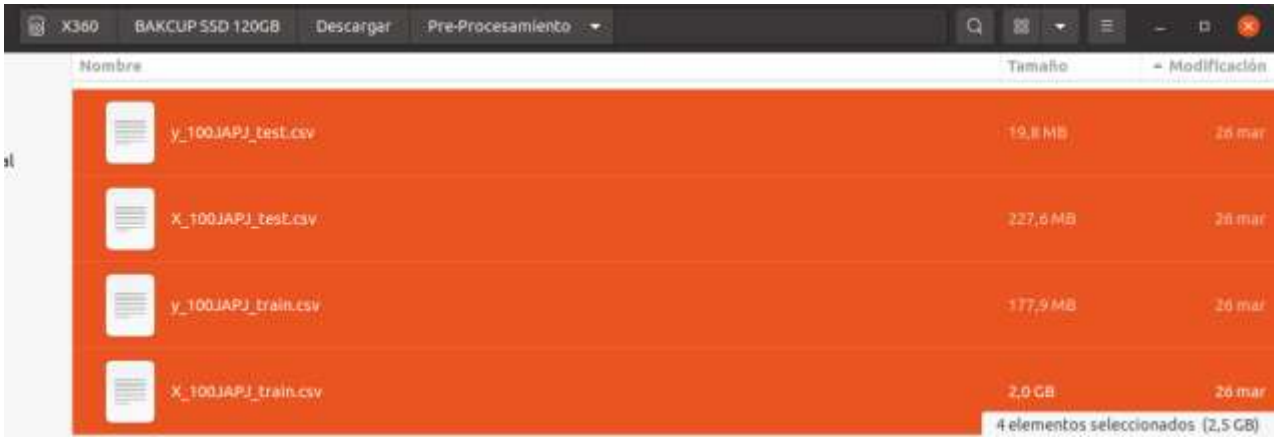


Figura 5-18 Cuatro subconjuntos de entrenamiento y prueba [Elaboración propia]

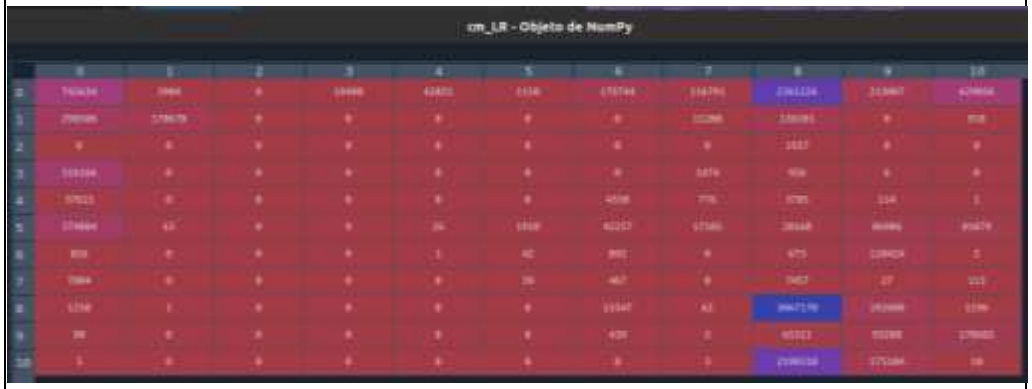
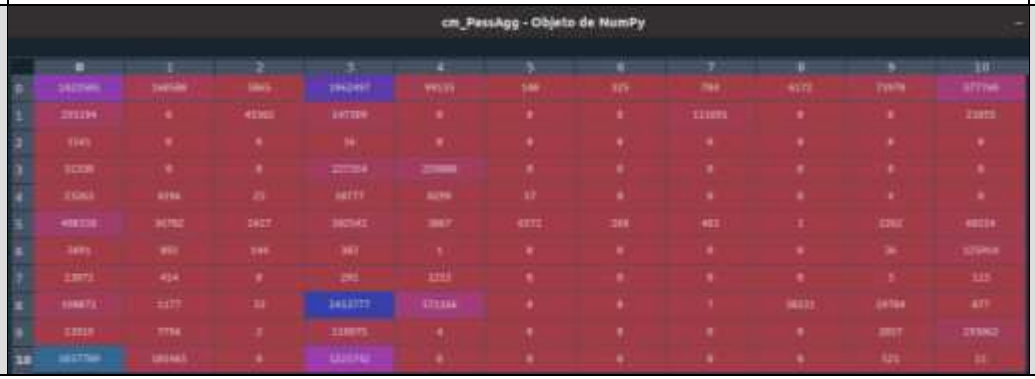
La división se ha decidido por ser **90%** de entrenamiento, y solo el **10%** de prueba. Mantenido la mayor cantidad posible para entrenamiento como se aprecia en la Figura 5-18.

Así los **13'596.273** de flujos de tráficos totales, se dividen en **12'236.645 flujos** para conjunto de entrenamiento y para **1'359.628 flujos** estratificados con los 10 ataques maliciosos de denegación para conjunto de prueba.

5.3. Etapa Aprendizaje: Entrenamiento

5.3.1. Selección del estimador: resultados entrenamiento

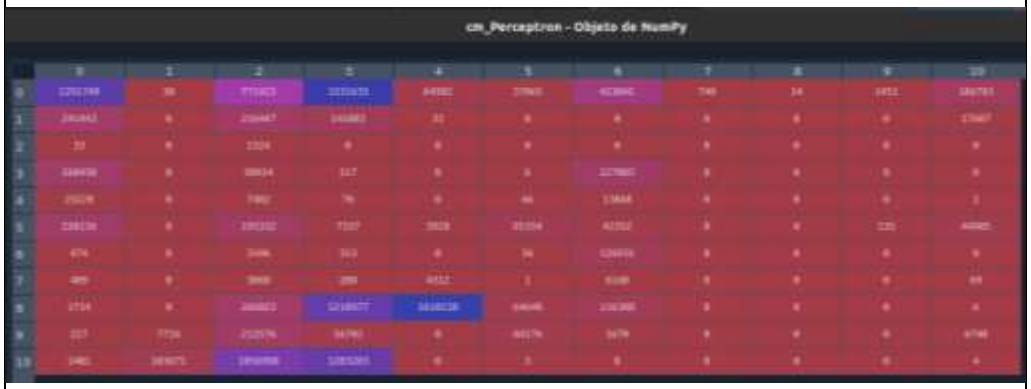
En la siguiente Tabla 5-2 se presentan los resultados resumidos de los siguientes 21 algoritmos de clasificación multiclase para aprendizaje supervisado. La idea ha sido encontrar los mejores cinco estimadores con el conjunto de entrenamiento y descartar el resto para el resto de análisis posterior, elegidos como pódium.

Algoritmo Aplicado	Reporte de clasificación	Matriz de confusión	Tiempo entrenamiento
Regresión Logística	Reporte de Clasificación ===> (Entrenamiento Regresión Logística) precision recall f1-score support 0 0.98 0.98 0.98 4309733 1 0.97 1.00 0.98 617411 2 0.78 0.86 0.82 1557 3 1.00 0.94 0.97 518572 4 0.71 0.83 0.77 46621 5 0.96 0.97 0.97 623687 6 0.99 0.99 0.99 130860 7 0.87 0.70 0.78 15107 8 0.91 1.00 0.95 3277834 9 0.60 0.04 0.08 329815 10 0.98 1.00 0.99 2365448 accuracy 0.96 12236645 macro avg 0.89 0.85 0.84 12236645 weighted avg 0.95 0.96 0.95 12236645		1681,0927 s 28,01min
Pasivo-Agresivos	precision recall f1-score support 0 0.89 0.97 0.93 4309733 1 0.98 1.00 0.99 617411 2 0.00 0.01 0.00 1557 3 1.00 0.86 0.92 518572 4 0.94 0.62 0.75 46621 5 1.00 0.91 0.95 623687 6 0.91 0.98 0.95 130860 7 0.55 0.22 0.31 15107 8 0.93 0.91 0.92 3277834 9 0.69 0.13 0.22 329815 10 0.98 1.00 0.99 2365448 accuracy 0.93 12236645 macro avg 0.81 0.69 0.72 12236645 weighted avg 0.93 0.93 0.92 12236645 El tiempo del algoritmo de la PassiveAggressive es de: 260.5305438041687		260,5305 s

Perceptrón

Reporte de Clasificación ===> (Entrenamiento Perceptron):

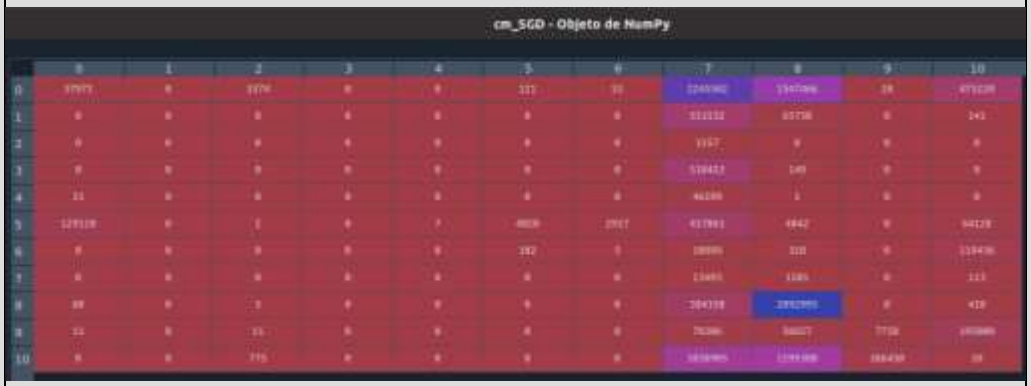
	precision	recall	f1-score	support
0	0.98	0.97	0.97	4309733
1	0.99	1.00	1.00	617411
2	0.66	0.99	0.79	1557
3	0.98	0.94	0.96	518572
4	0.76	0.64	0.69	46621
5	0.98	0.97	0.97	623687
6	0.96	0.99	0.97	130860
7	0.58	0.10	0.17	15107
8	0.92	1.00	0.96	3277834
9	0.66	0.17	0.27	329815
10	0.98	1.00	0.99	2365448
accuracy			0.96	12236645
macro avg	0.86	0.80	0.80	12236645
weighted avg	0.95	0.96	0.95	12236645



Descenso de Gradiente Estocástico SGD

	precision	recall	f1-score	support
0	0.98	0.89	0.93	4309733
1	0.96	1.00	0.98	617411
2	0.70	0.75	0.72	1557
3	0.96	0.94	0.95	518572
4	0.69	0.45	0.55	46621
5	0.92	0.96	0.94	623687
6	0.96	0.96	0.96	130860
7	0.72	0.22	0.33	15107
8	0.83	1.00	0.90	3277834
9	0.01	0.00	0.00	329815
10	0.98	1.00	0.99	2365448
accuracy			0.92	12236645
macro avg	0.79	0.74	0.75	12236645
weighted avg	0.91	0.92	0.91	12236645

El tiempo del algoritmo de la SGD es de: 290.28904938697815

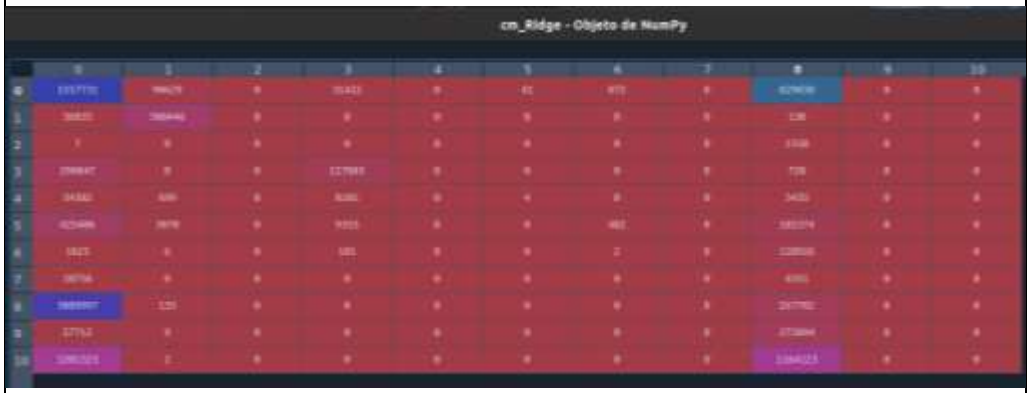


290,2890 s

<table>
<tr>
<td>Rigde (Cresta)</td>
<td>
<pre>
              precision   recall f1-score   support

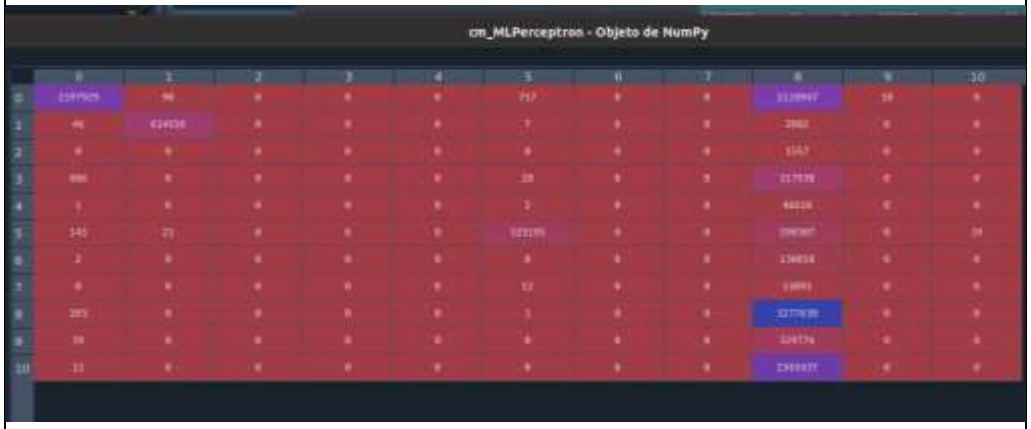
           0   0.95   0.85   0.90   4309733
           1   0.92   1.00   0.96   617411
           2   0.73   0.96   0.83   1557
           3   0.81   0.72   0.76   518572
           4   0.00   0.00   0.00   46621
           5   0.95   0.22   0.36   623687
           6   0.97   0.83   0.90   130860
           7   0.00   0.00   0.00   15107
           8   0.72   1.00   0.84   3277834
           9   0.00   0.00   0.00   329815
          10   0.98   1.00   0.99   2365448

    accuracy   0.86 12236645
   macro avg   0.64   0.60   0.59 12236645
weighted avg   0.86   0.86   0.84 12236645

El tiempo del algoritmo de la Ridge es de: 60.75147318840027
</pre>
</td>
<td> 
</td>
<td>60,7514 s</td>
</tr>
<tr>
<td>MLP</td>
<td>
<pre>
  _warn_prf(average, modifier, msg_start, len(result))
              precision   recall f1-score   support

           0   1.00   0.51   0.68   4309733
           1   1.00   1.00   1.00   617411
           2   0.00   0.00   0.00   1557
           3   0.00   0.00   0.00   518572
           4   0.00   0.00   0.00   46621
           5   1.00   0.52   0.68   623687
           6   0.00   0.00   0.00   130860
           7   0.00   0.00   0.00   15107
           8   0.36   1.00   0.53   3277834
           9   0.00   0.00   0.00   329815
          10   0.00   0.00   0.00   2365448

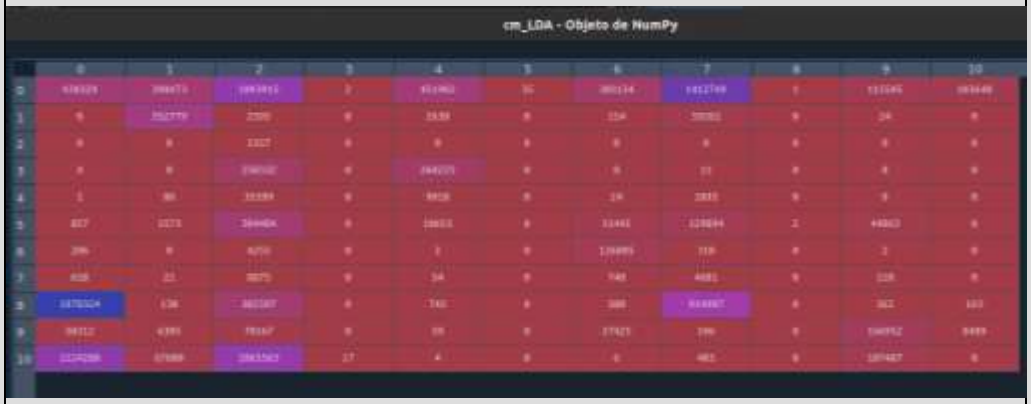
    accuracy   0.52 12236645
   macro avg   0.31   0.28   0.26 12236645
weighted avg   0.55   0.52   0.47 12236645

/home/andres/anaconda3/lib/python3.7/site-packages/sklearn/
</pre>
</td>
<td> 
</td>
<td>935,1825 s</td>
</tr>
</table>

LDA

Reporte de Clasificaciòn ===> (Entrenamiento LDA):

	precision	recall	f1-score	support
0	0.97	0.77	0.86	4309733
1	0.89	1.00	0.94	617411
2	0.73	0.98	0.84	1557
3	0.60	0.86	0.71	518572
4	0.21	0.64	0.31	46621
5	0.93	0.22	0.35	623687
6	0.95	0.84	0.90	130860
7	0.39	0.60	0.47	15107
8	0.73	1.00	0.84	3277834
9	0.26	0.01	0.02	329815
10	0.98	1.00	0.99	2365448
accuracy			0.84	12236645
macro avg	0.69	0.72	0.66	12236645
weighted avg	0.86	0.84	0.83	12236645

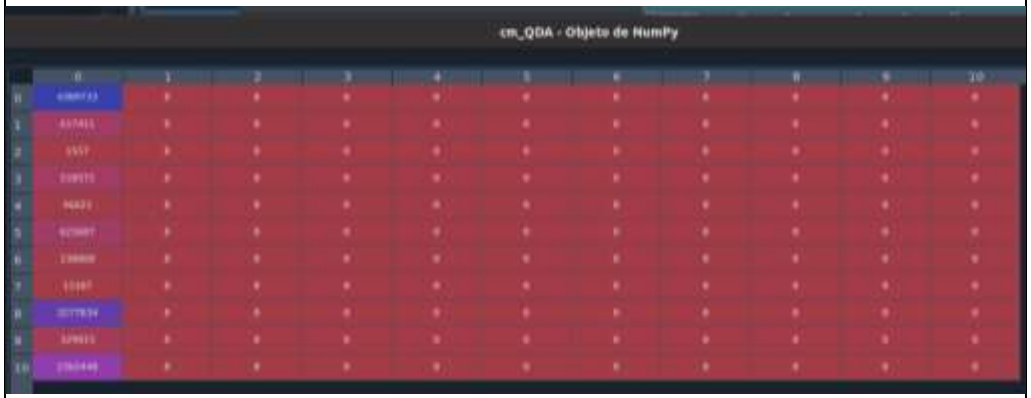


700,6967 s

QDA

Reporte de Clasificaciòn ===> (Entrenamiento QDA):

	precision	recall	f1-score	support
0	1.00	0.92	0.96	4309733
1	1.00	1.00	1.00	617411
2	0.71	1.00	0.83	1557
3	0.99	1.00	0.99	518572
4	0.18	1.00	0.30	46621
5	0.74	0.65	0.69	623687
6	1.00	0.99	1.00	130860
7	0.98	1.00	0.99	15107
8	0.98	1.00	0.99	3277834
9	0.67	0.88	0.76	329815
10	0.98	1.00	0.99	2365448
accuracy			0.95	12236645
macro avg	0.84	0.95	0.86	12236645
weighted avg	0.96	0.95	0.95	12236645



381,6747 s

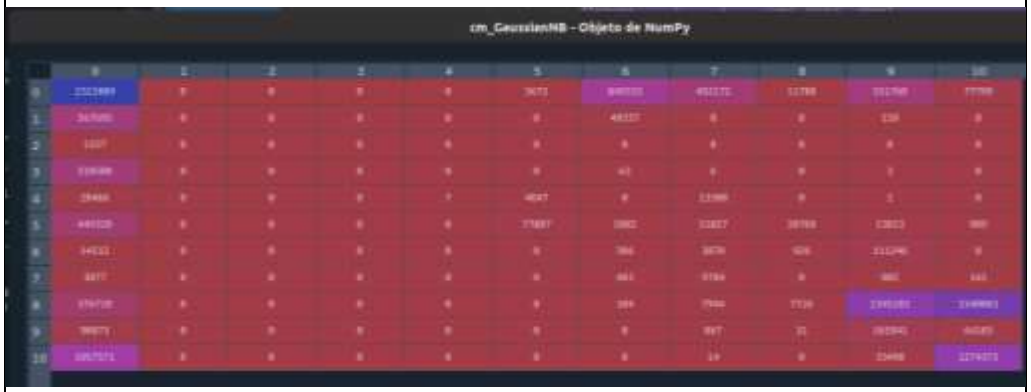
Multinomial NB

```
  _warn_prf(average, modifier, msg_start, len(result))
/home/andres/anaconda3/lib/python3.7/site-packages/sklearn/m
are ill-defined and being set to 0.0 in labels with no predi
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           0       0.80      0.88      0.84   4309733
           1       0.00      0.00      0.00    617411
           2       0.73      0.98      0.84      1557
           3       0.41      0.02      0.03    518572
           4       0.00      0.00      0.00     46621
           5       0.92      0.20      0.34    623687
           6       0.98      0.01      0.01    130860
           7       0.55      0.21      0.31     15107
           8       0.66      0.99      0.79   3277834
           9       0.00      0.00      0.00    329815
          10       0.97      1.00      0.99   2365448

    accuracy                           0.78  12236645
   macro avg       0.55      0.39      0.38  12236645
weighted avg       0.72      0.78      0.72  12236645
```

682,3919 s

Gaussian NB

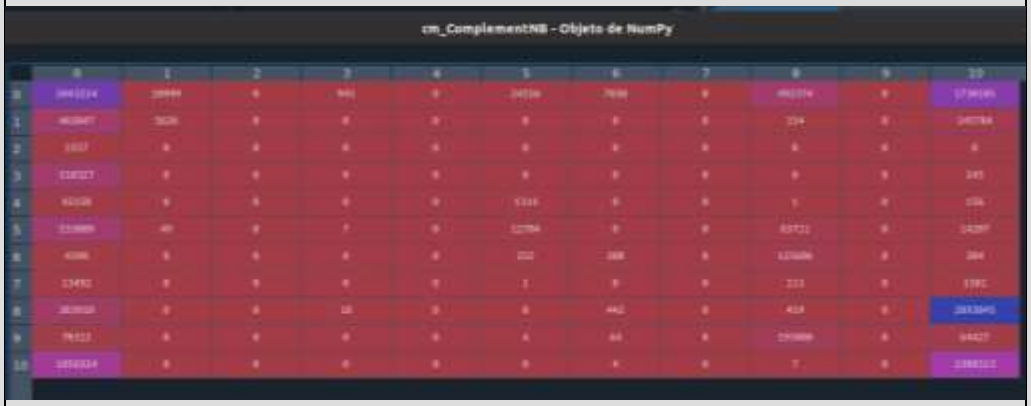
```
Reporte de Clasificación ===> (Entrenamiento GaussianNB)
              precision    recall  f1-score   support

           0       0.99      0.65      0.78   4309733
           1       0.70      1.00      0.83    617411
           2       0.69      1.00      0.82      1557
           3       0.87      1.00      0.93    518572
           4       0.58      0.98      0.73     46621
           5       0.85      0.94      0.90    623687
           6       0.99      0.96      0.98    130860
           7       0.40      0.96      0.56     15107
           8       0.91      0.99      0.95   3277834
           9       0.01      0.02      0.01    329815
          10       0.98      1.00      0.99   2365448

    accuracy                           0.84  12236645
   macro avg       0.72      0.86      0.77  12236645
weighted avg       0.91      0.84      0.86  12236645
```



334,1200 s

Complement NB

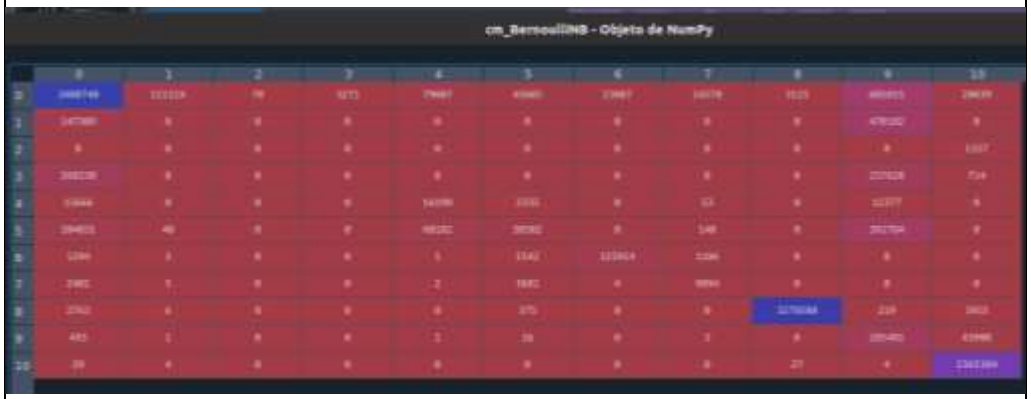
```
  _warn_prf(average, modifier, msg_start, len(result))
/home/andres/anaconda3/lib/python3.7/site-packages/sklearn/
and being set to 0.0 in labels with no predicted samples. U
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

           0       0.74      0.56      0.64   4309733
           1       0.00      0.00      0.00    617411
           2       0.71      0.98      0.83      1557
           3       0.57      0.14      0.23    518572
           4       0.00      0.00      0.00     46621
           5       0.86      0.02      0.04    623687
           6       0.00      0.00      0.00    130860
           7       0.00      0.00      0.00     15107
           8       0.64      0.99      0.78   3277834
           9       0.00      0.00      0.00    329815
          10       0.63      1.00      0.77   2365448

    accuracy                           0.66  12236645
   macro avg       0.38      0.34      0.30  12236645
weighted avg       0.62      0.66      0.59  12236645
```



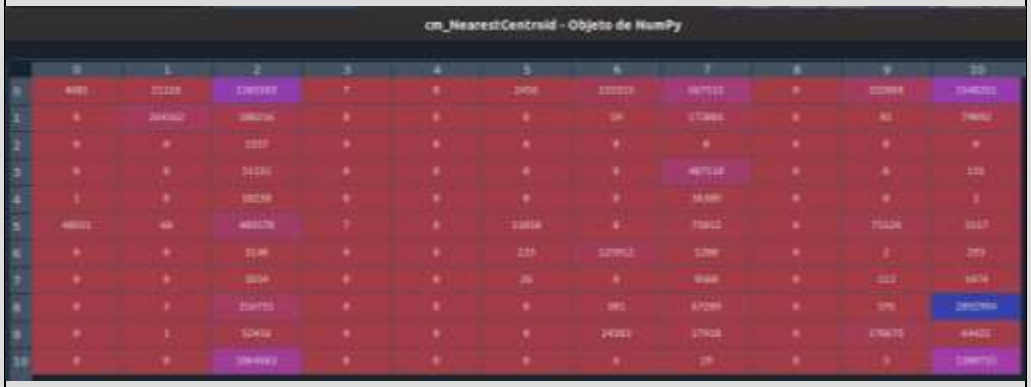
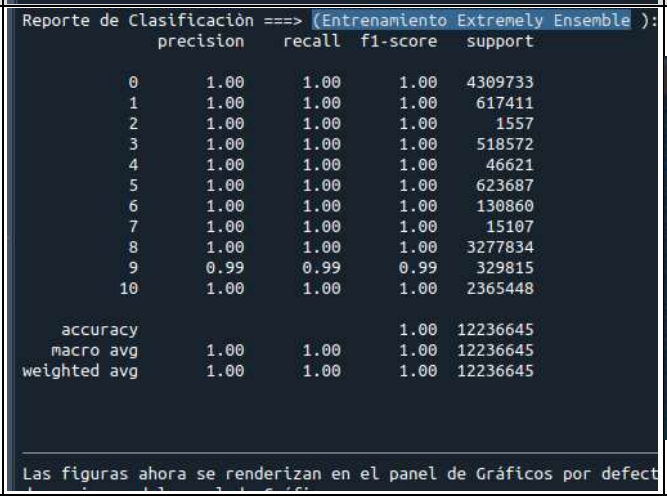
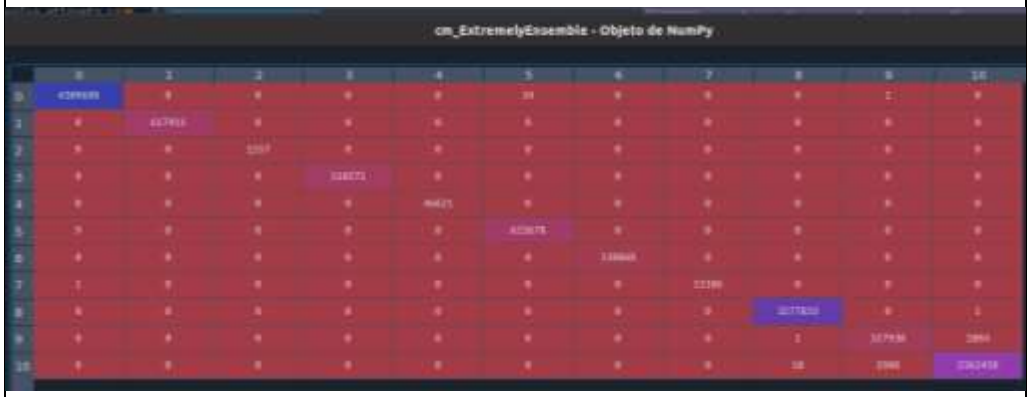
669,1558 s

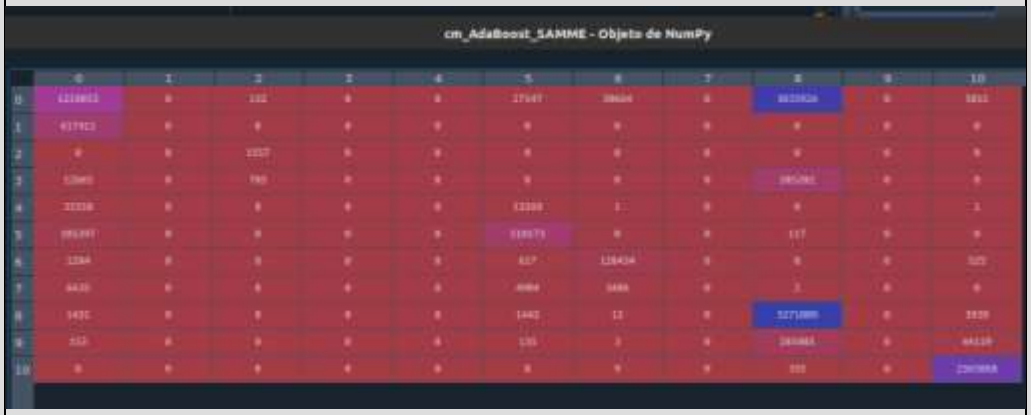
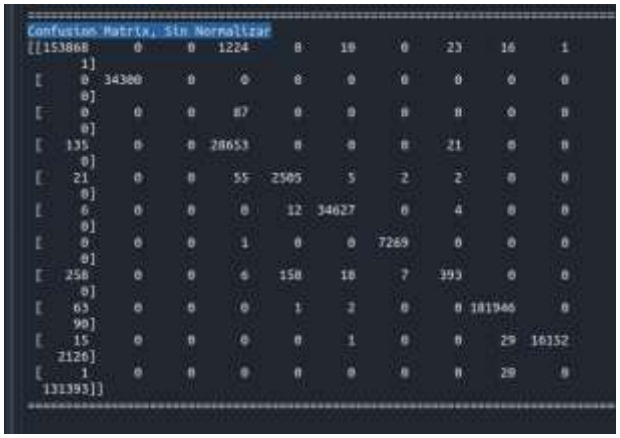
Bernoulli NB

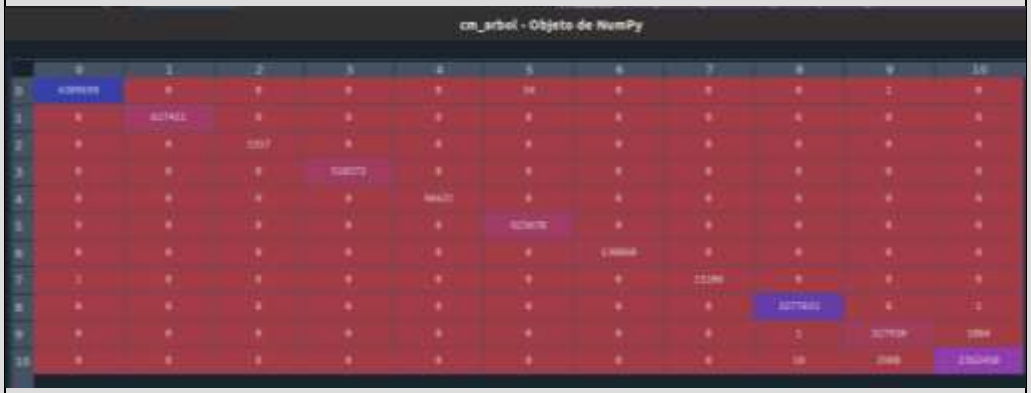
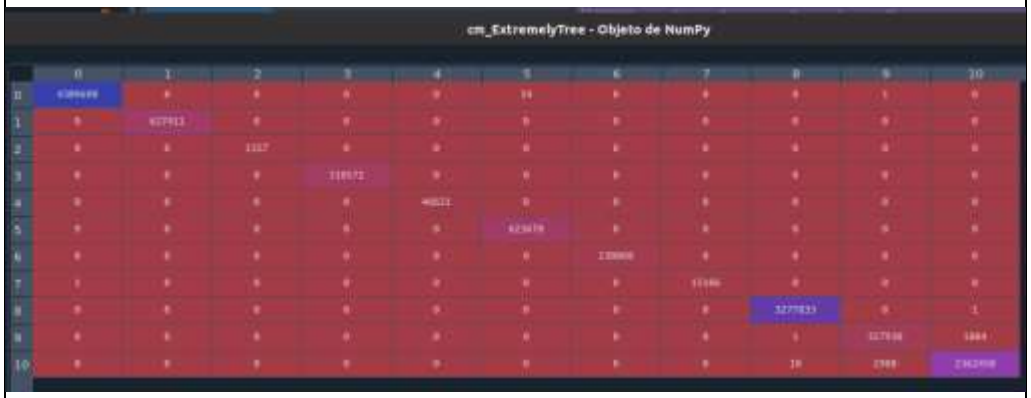
```
Reporte de Clasificaciòn ===> (Entrenamiento Bernoulli NB)
              precision    recall  f1-score   support

           0       0.87      0.82      0.85   4309733
           1       0.00      0.00      0.00    617411
           2       0.00      0.00      0.00      1557
           3       0.00      0.00      0.00    518572
           4       0.10      0.35      0.15     46621
           5       0.52      0.09      0.16    623687
           6       0.80      0.96      0.87    130860
           7       0.36      0.60      0.45     15107
           8       1.00      1.00      1.00   3277834
           9       0.15      0.87      0.26    329815
          10       0.97      1.00      0.99   2365448

    accuracy                           0.79  12236645
   macro avg       0.43      0.52      0.43  12236645
weighted avg       0.80      0.79      0.78  12236645
```



251,9191 s

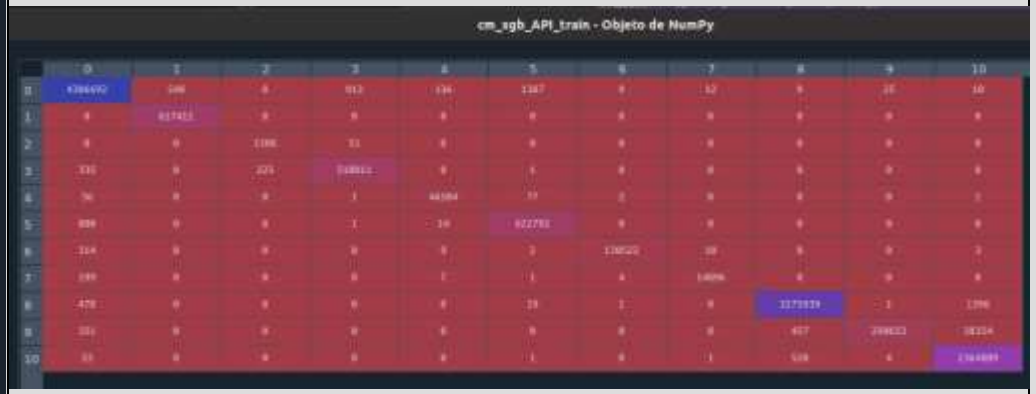
Nearest Centroid			**40,7289 s**
Extra Trees Emsamble			**1147,7456 s 19,12 min**

<table>
<tr>
<td>AdaBoost-SAMME</td>
<td>
set to 0.0 in labels with no predicted samples. Use `zero_divi
_warn_prf(average, modifier, msg_start, len(result))
/home/andres/anaconda3/lib/python3.7/site-packages/sklearn/met
set to 0.0 in labels with no predicted samples. Use `zero_divi
_warn_prf(average, modifier, msg_start, len(result))
<table>
<tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr>
<tr><td>0</td><td>0.61</td><td>0.28</td><td>0.38</td><td>4309733</td></tr>
<tr><td>1</td><td>0.00</td><td>0.00</td><td>0.00</td><td>617411</td></tr>
<tr><td>2</td><td>0.65</td><td>1.00</td><td>0.79</td><td>1557</td></tr>
<tr><td>3</td><td>0.00</td><td>0.00</td><td>0.00</td><td>518572</td></tr>
<tr><td>4</td><td>0.00</td><td>0.00</td><td>0.00</td><td>46621</td></tr>
<tr><td>5</td><td>0.92</td><td>0.83</td><td>0.87</td><td>623687</td></tr>
<tr><td>6</td><td>0.79</td><td>0.98</td><td>0.88</td><td>130860</td></tr>
<tr><td>7</td><td>0.00</td><td>0.00</td><td>0.00</td><td>15107</td></tr>
<tr><td>8</td><td>0.46</td><td>1.00</td><td>0.63</td><td>3277834</td></tr>
<tr><td>9</td><td>0.00</td><td>0.00</td><td>0.00</td><td>329815</td></tr>
<tr><td>10</td><td>0.98</td><td>1.00</td><td>0.99</td><td>2365448</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.61</td><td>12236645</td></tr>
<tr><td>macro avg</td><td>0.40</td><td>0.46</td><td>0.41</td><td>12236645</td></tr>
<tr><td>weighted avg</td><td>0.58</td><td>0.61</td><td>0.55</td><td>12236645</td></tr>
</table>
</td>
<td> 
</td>
<td>34,46 min</td>
</tr>
<tr>
<td>GradientBoosting</td>
<td>0,99 0,86 0,99</td>
<td> 
</td>
<td>13,68 H</td>
</tr>
</table>

Árbol Decisión DT	Reporte de Clasificación ===> (Entrenamiento Árbol de Decisión): precision recall f1-score support 0 1.00 1.00 1.00 4309733 1 1.00 1.00 1.00 617411 2 1.00 1.00 1.00 1557 3 1.00 1.00 1.00 518572 4 1.00 1.00 1.00 46621 5 1.00 1.00 1.00 623687 6 1.00 1.00 1.00 130860 7 1.00 1.00 1.00 15107 8 1.00 1.00 1.00 3277834 9 0.99 0.99 0.99 329815 10 1.00 1.00 1.00 2365448 accuracy 1.00 12236645 macro avg 1.00 1.00 1.00 12236645 weighted avg 1.00 1.00 1.00 12236645 El tiempo del algoritmo del Árbol de Decisión es de: 629.4979996		629,4979 s
Árbol Extra ET	Reporte de Clasificación ===> (Entrenamiento Extremely Tree): precision recall f1-score support 0 1.00 1.00 1.00 4309733 1 1.00 1.00 1.00 617411 2 1.00 1.00 1.00 1557 3 1.00 1.00 1.00 518572 4 1.00 1.00 1.00 46621 5 1.00 1.00 1.00 623687 6 1.00 1.00 1.00 130860 7 1.00 1.00 1.00 15107 8 1.00 1.00 1.00 3277834 9 0.99 0.99 0.99 329815 10 1.00 1.00 1.00 2365448 accuracy 1.00 12236645 macro avg 1.00 1.00 1.00 12236645 weighted avg 1.00 1.00 1.00 12236645 El tiempo del algoritmo de Extremely Tree es de: 149.389144182		149,3891s

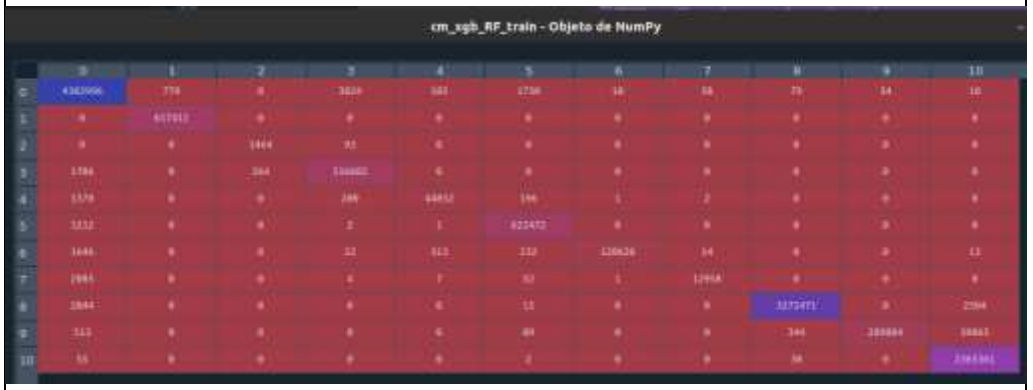
xgboost.Dmatrix xgboost.train

```
   ...: imprimir_reporte('Entrenamiento XGBoost API', bst,
Reporte de Clasificaciòn ===> (Entrenamiento XGBoost API):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   4309733
           1       1.00      1.00      1.00    617411
           2       0.87      0.97      0.92      1557
           3       1.00      1.00      1.00    518572
           4       1.00      1.00      1.00     46621
           5       1.00      1.00      1.00    623687
           6       1.00      1.00      1.00    130860
           7       1.00      0.99      0.99     15107
           8       1.00      1.00      1.00   3277834
           9       1.00      0.88      0.94    329815
          10       0.98      1.00      0.99   2365448

    accuracy                           1.00  12236645
   macro avg       0.99      0.98      0.98  12236645
weighted avg       1.00      1.00      1.00  12236645
```



xgboost.XGBClassifier

```
In [5]: imprimir_reporte('Entrenamiento XGB xgb_RF', xgb_
Reporte de Clasificaciòn ===> (Entrenamiento XGB xgb_RF):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00   4309733
           1       1.00      1.00      1.00    617411
           2       0.85      0.94      0.89      1557
           3       0.99      1.00      0.99    518572
           4       0.99      0.96      0.98     46621
           5       1.00      1.00      1.00    623687
           6       1.00      0.98      0.99    130860
           7       0.99      0.86      0.92     15107
           8       1.00      1.00      1.00   3277834
           9       1.00      0.88      0.94    329815
          10       0.98      1.00      0.99   2365448

    accuracy                           0.99  12236645
   macro avg       0.98      0.96      0.97  12236645
weighted avg       1.00      0.99      0.99  12236645
```



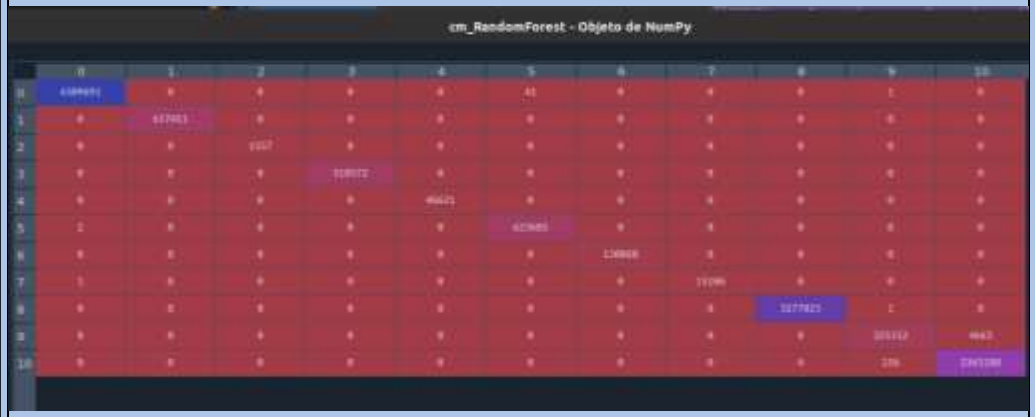
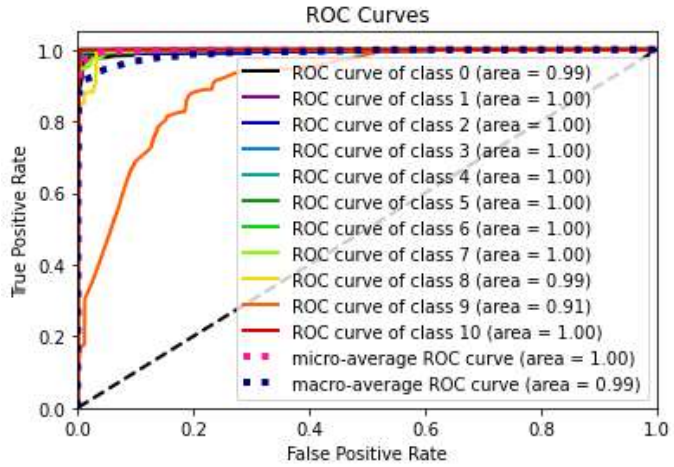
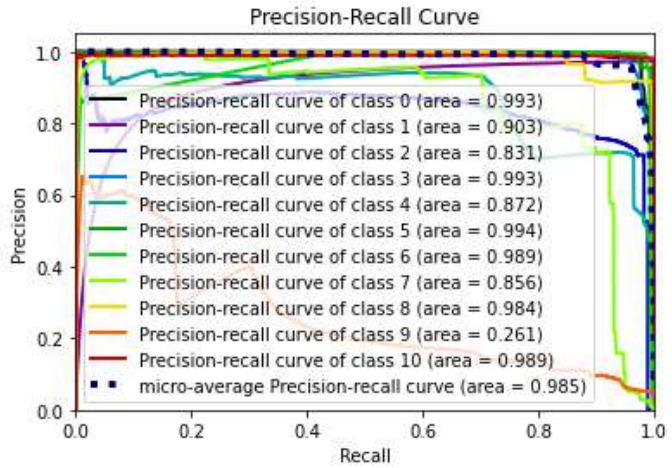
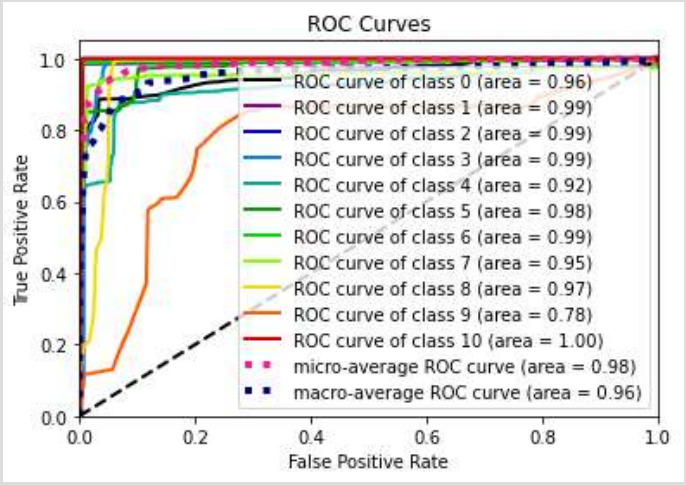
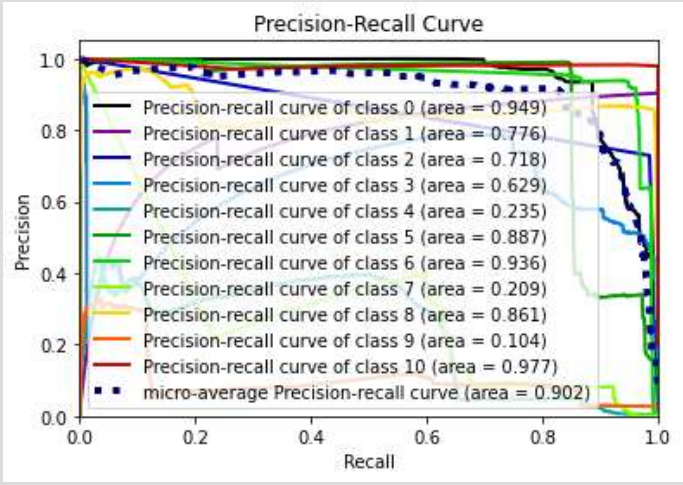
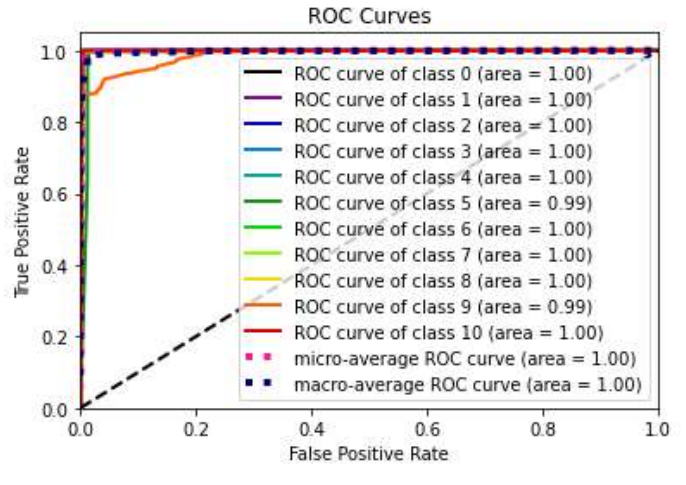
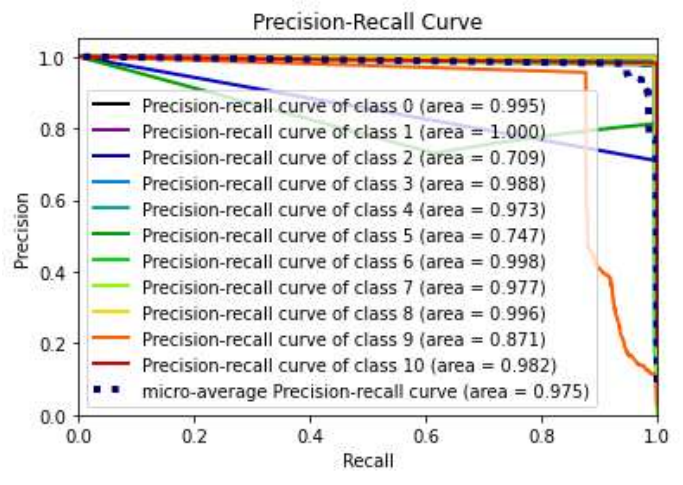
Randon Forest	Reporte de Clasificación ===> (Entrenamiento Random Forest): precision recall f1-score support 0 1.00 1.00 1.00 4309733 1 1.00 1.00 1.00 617411 2 1.00 1.00 1.00 1557 3 1.00 1.00 1.00 518572 4 1.00 1.00 1.00 46621 5 1.00 1.00 1.00 623687 6 1.00 1.00 1.00 130860 7 1.00 1.00 1.00 15107 8 1.00 1.00 1.00 3277834 9 1.00 0.99 0.99 329815 10 1.00 1.00 1.00 2365448 accuracy 1.00 12236645 macro avg 1.00 1.00 1.00 12236645 weighted avg 1.00 1.00 1.00 12236645		1266.091765 s 21.10min

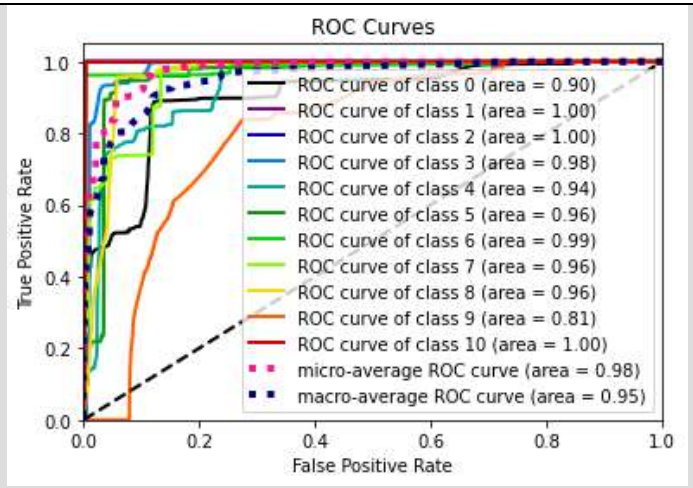
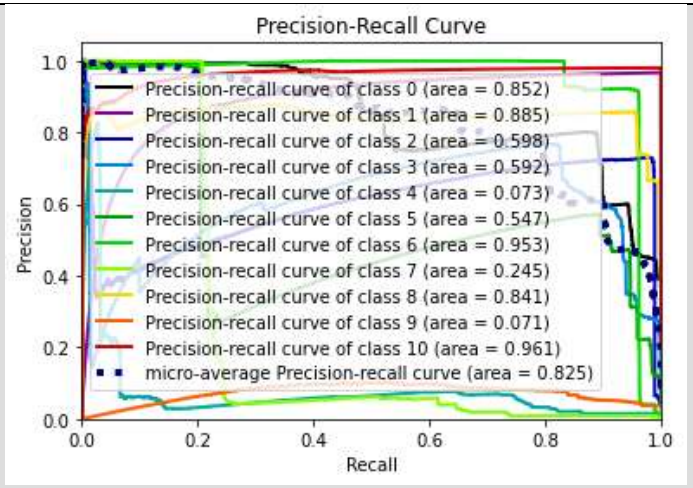
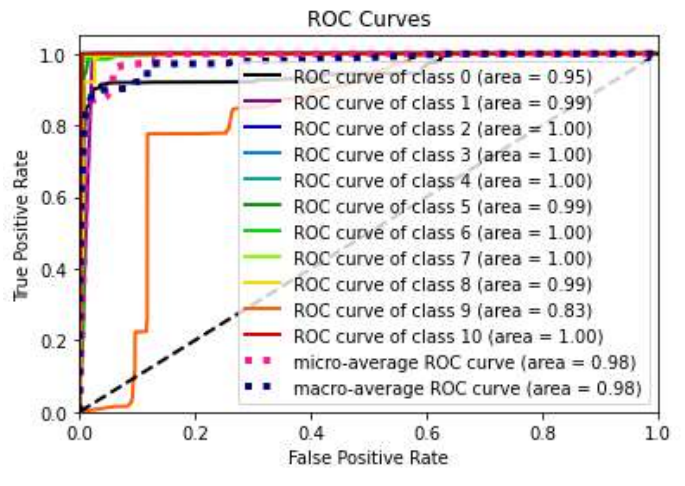
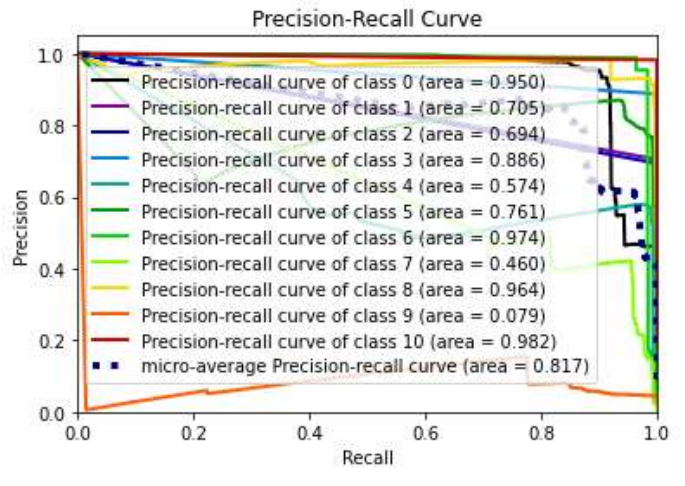
Tabla 5-2 Reportes clasificación y Matrices de Confusión de 21 algoritmos de aprendizaje

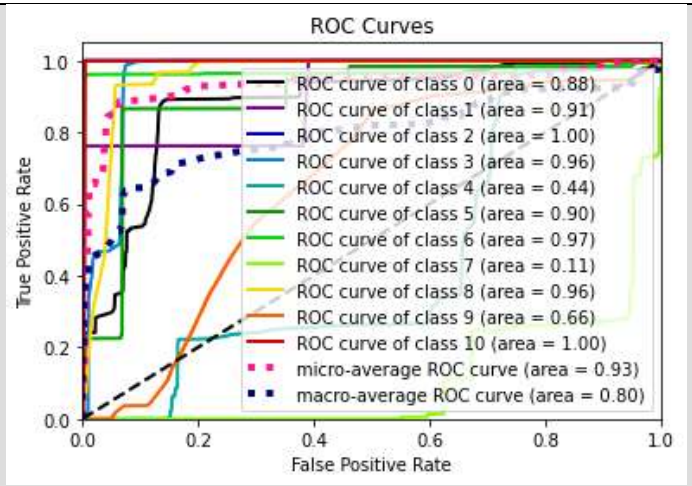
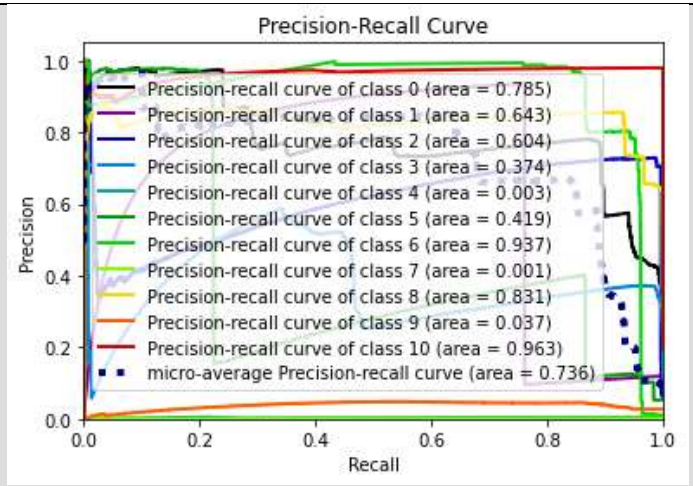
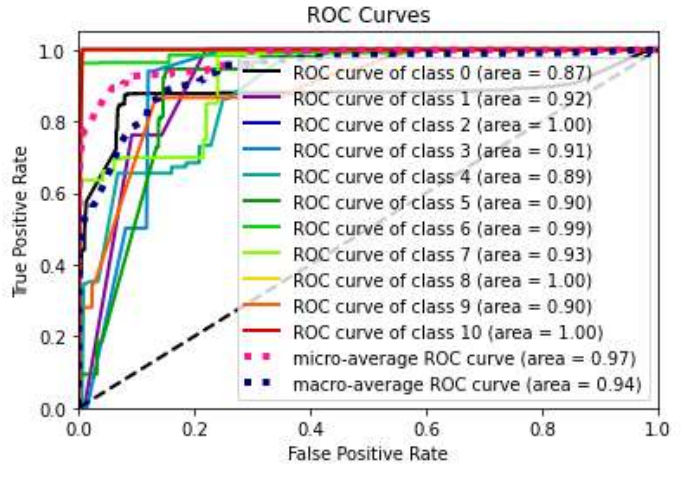
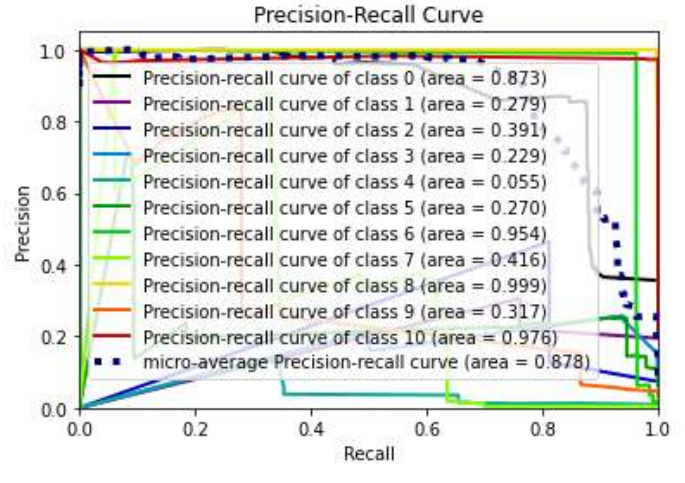
Con solo observar las matrices de confusiones de la Tabla 5-2 se nota la superioridad de los algoritmos de **basado en conjuntos** (*ensemble-based*) y sus **árboles de decisión** (*tree-based*), se eligen respaldados en su deducción experimental, y cabe mencionar algo muy curioso; tanto **ET**, **DT** y el **ExtraTrees** dieron el exacto e idéntico resultado en el conjunto de entrenamiento, pero de menor a mayor tiempo gastado respectivamente. También tienen diferentes capacidades al ser guardado el modelo, y como se verá más adelante presentan diferentes valores de evaluación, porque son diferentes, pero con idéntico resultado en su entrenamiento.

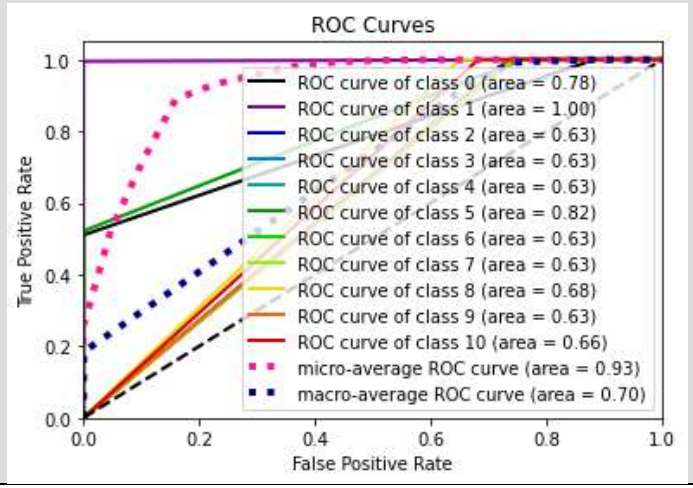
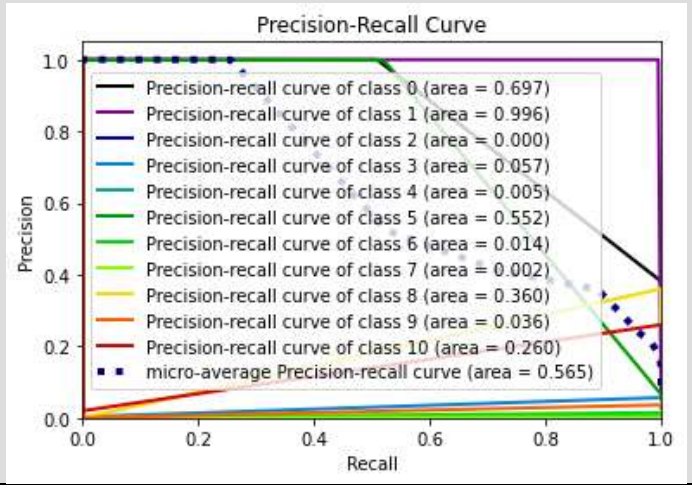
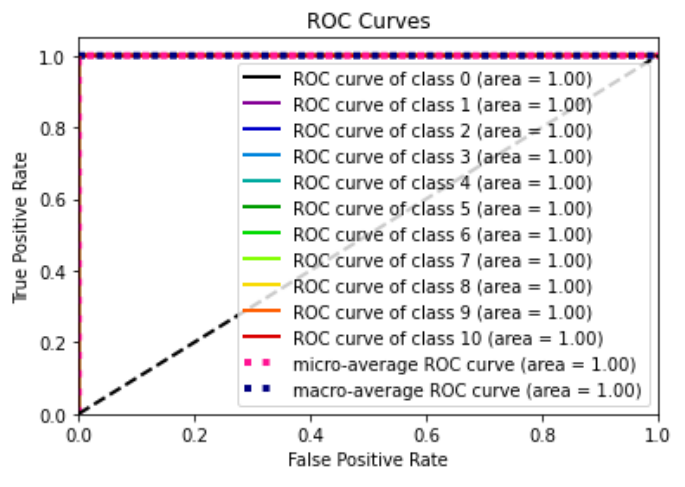
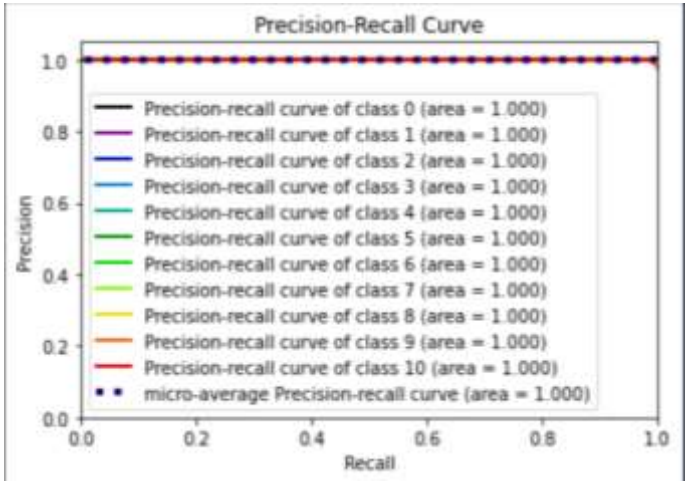
Importante entre más se es explícito en los parámetros, mejores resultados presentan **XGBoost**.

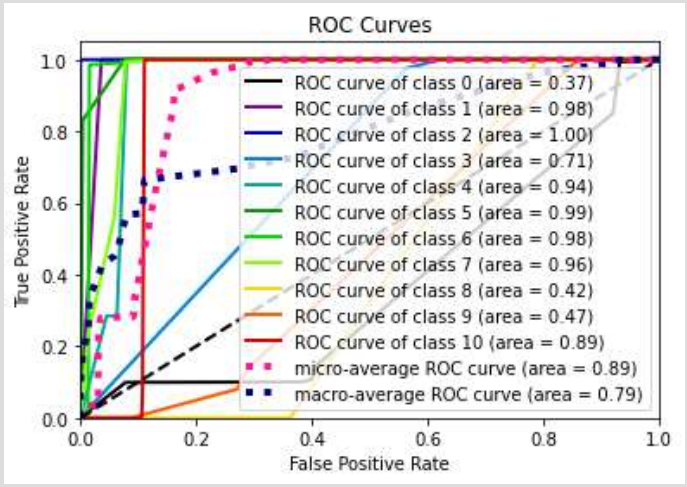
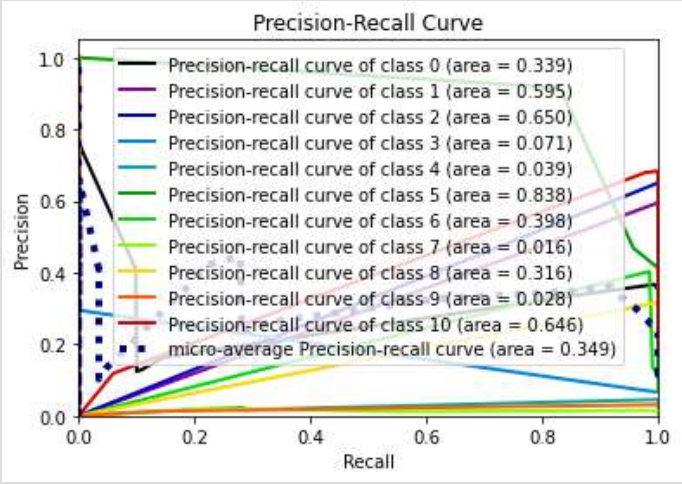
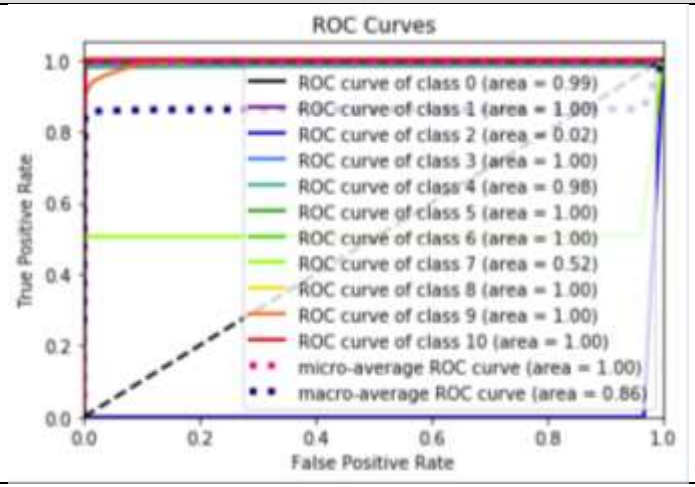
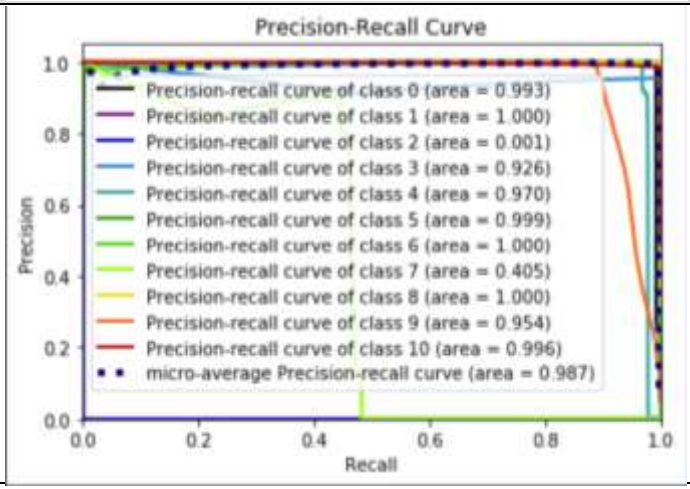
Algoritmo Aplicado	Reporte de clasificación	Matriz de confusión	Tiempo entrenamiento
Regresión Logística			**1348,61 s** **22,74min**
P-A	`No predict_proba(X)`	`No predict_proba(X)`	
Percep trón	`No predict_proba(X)`	`No predict_proba(X)`	
SGD	`No predict_proba(X)`	`No predict_proba(X)`	
Rigde	`No predict_proba(X)`	`No predict_proba(X)`	

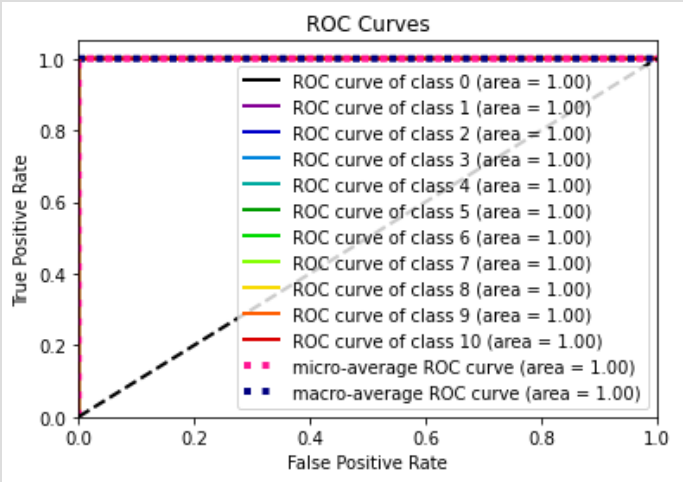
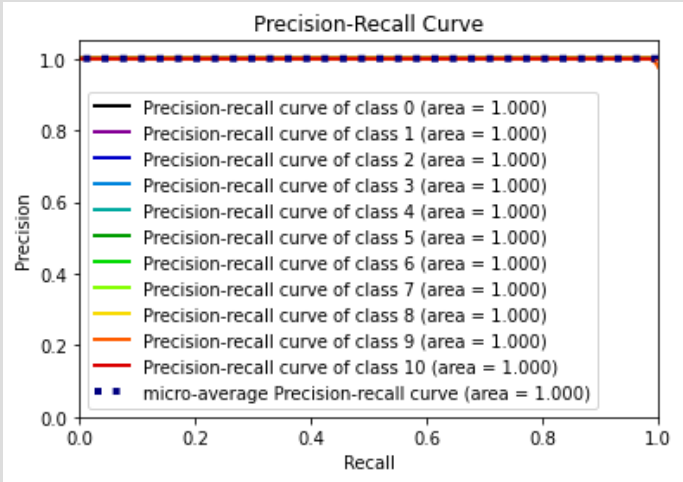
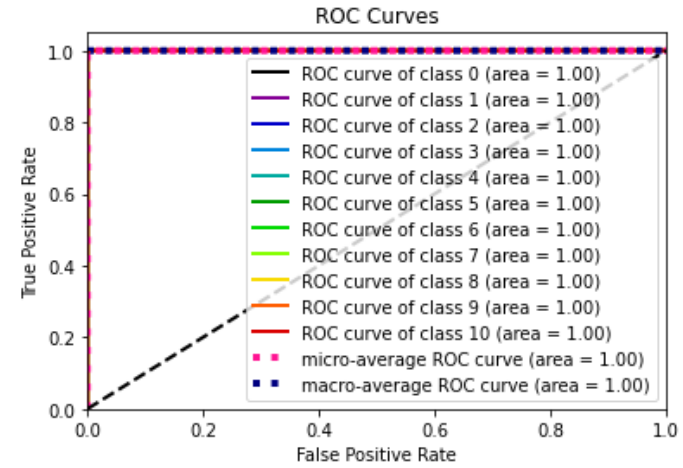
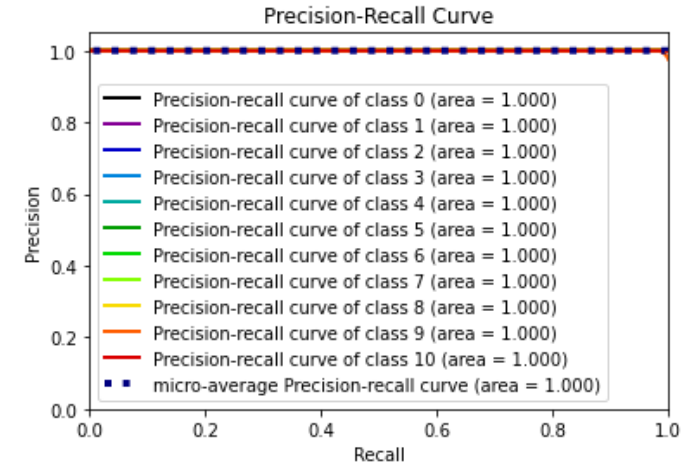
LDA			**61,94 s**
QDA			**74,29 s**

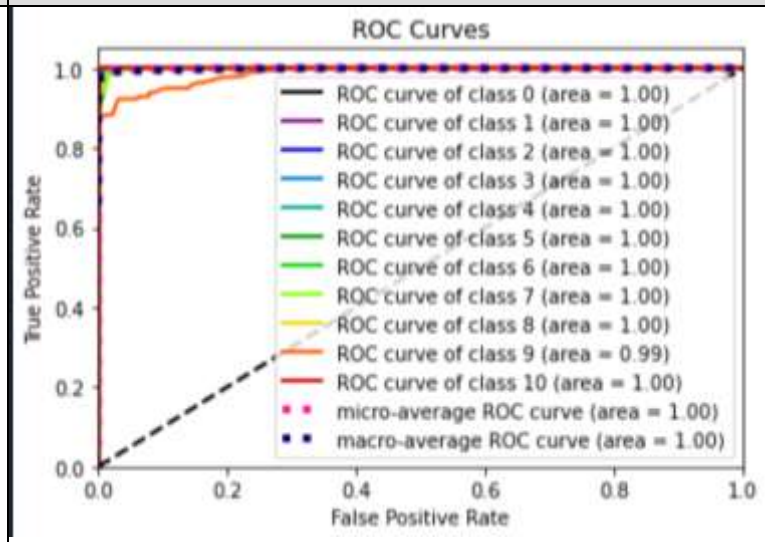
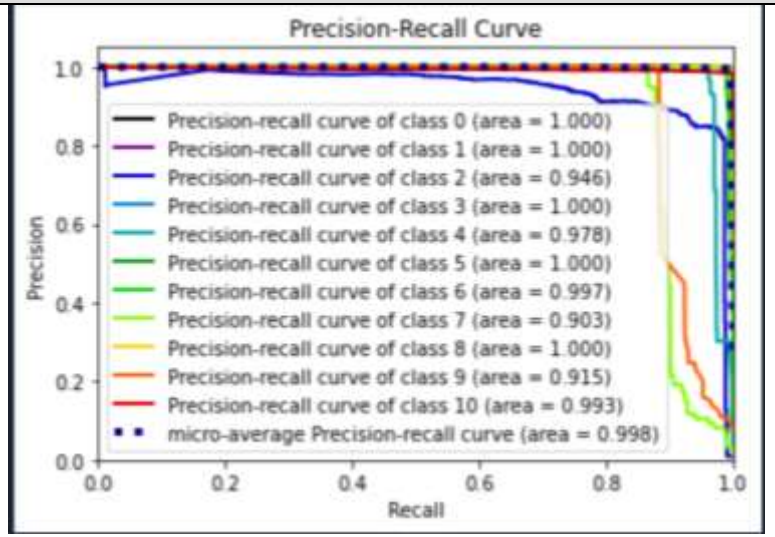
Multinomial NB		
Gaussian NB		

Complement NB		
Bernoulli NB		

MLPClassifier		
Extra Trees Emsamble		

AdaBoost-SAMME		
GradientBoosting		

Árbol Decisión DT		
Árbol Extra ET		

xgboost.D	`No predict_proba(X)`	`No predict_proba(X)`	
xgboost.XGBClassifier			
Nearest Crentroid	`No predict_proba(X)`	`No predict_proba(X)`	

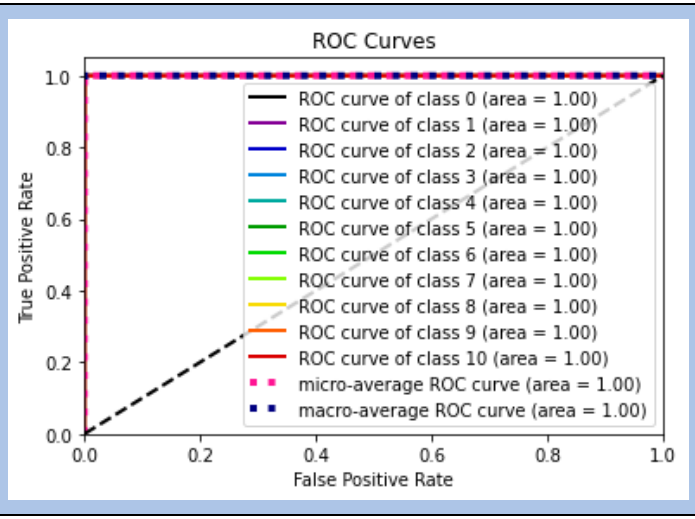
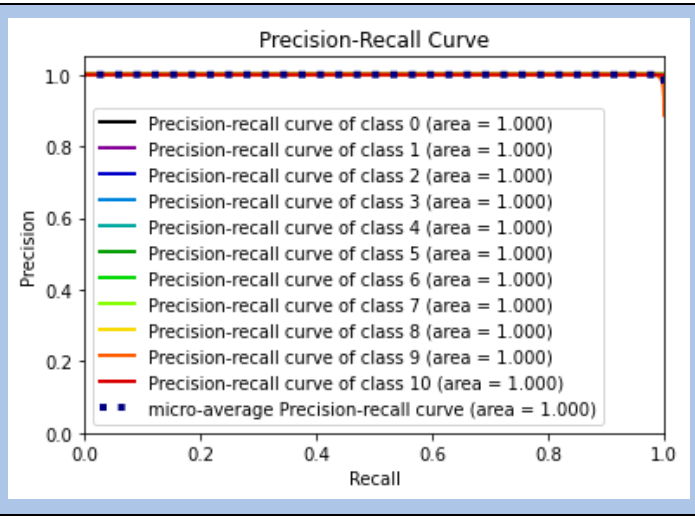
Randon Forest		

Tabla 5-3 Curvas ROC y Curvas Precision-Recall de 21 algoritmos de aprendizaje

Primero explicar en la Tabla 5-3 donde se comparan las curvas ***ROC*** y ***Precision-Recall*** y se evidencia celdas sin ellas, es porque es necesario para su elaboración partir del método del estimador llamado ***.predict_proba***, cuyo método de objeto no todos los tipos de estimadores lo implementan, para ejemplo el caso: Ridge, SGD, Perceptron, Passive-Agressive y Nearest Crentroid. Las Curvas ROC compara las variables de tasa de verdaderos positivos *TPR* versus tasa de falsos positivos *FPR*. Las Curvas P-R compara las variables de Exactitud versus Exhaustividad. Ambas graficas entre más cercas a 1.00 es lo ideal.

En la relación de las Tabla 5-2 y Tabla 5-3, se ha descartado en etapa de entrenamiento los algoritmos de aprendizaje basados en modelos lineales (cinco), basados en análisis discriminante (dos), basados en Naive Bayes (cuatro), basado en Vecinos Cercanos (uno), Redes Neuronal (uno) y dos de *Ensemble:* AdaBoost Classifier & Gradient Boosting Classifier. Descartados en total 15 algoritmos de aprendizaje.

Como ya se ha mencionado, es evidente la superioridad de los algoritmos de basado en conjuntos (***ensemble-based***) y sus Árboles de decisión (***tree-based***) su relación, su rendimiento, su desempeño. Es tan impresionante que más adelante se verá resultados de validación cruzada solo en estos, para confirmar que no están incurriendo en una mala generalización como lo es el sobreajuste (***overfitting***) donde aprende digamos de memoria; los demás se descartan totalmente, y se llega hasta este punto con ellos dieciséis (16) algoritmos de aprendizaje restantes.

Cabe mencionar que dichos algoritmos descartados corresponden a estimadores que han sufrido problemas de infraajuste (***underfitting***) pues no son suficientes para ajustarse a las muestras de entrenamiento; algunos mejoran en su resultado, se observa que hay alguna tendencia en los datos, pero no son lo suficientemente específicos como para captar la información relevante, como si lo realizan el pódium seleccionado.

5.3.2. Medición del Rendimiento: Conjunto Entrenamiento

En un modelo simple se tiene alto de sesgo, un modelo complejo en varios casos tendrá una varianza alta. Las curvas de Aprendizaje pueden ayudan a lograr un mejor diagnóstico del modelo seleccionado y se ha alcanzado a identificar la situación que se tiene para intentar maximizar.

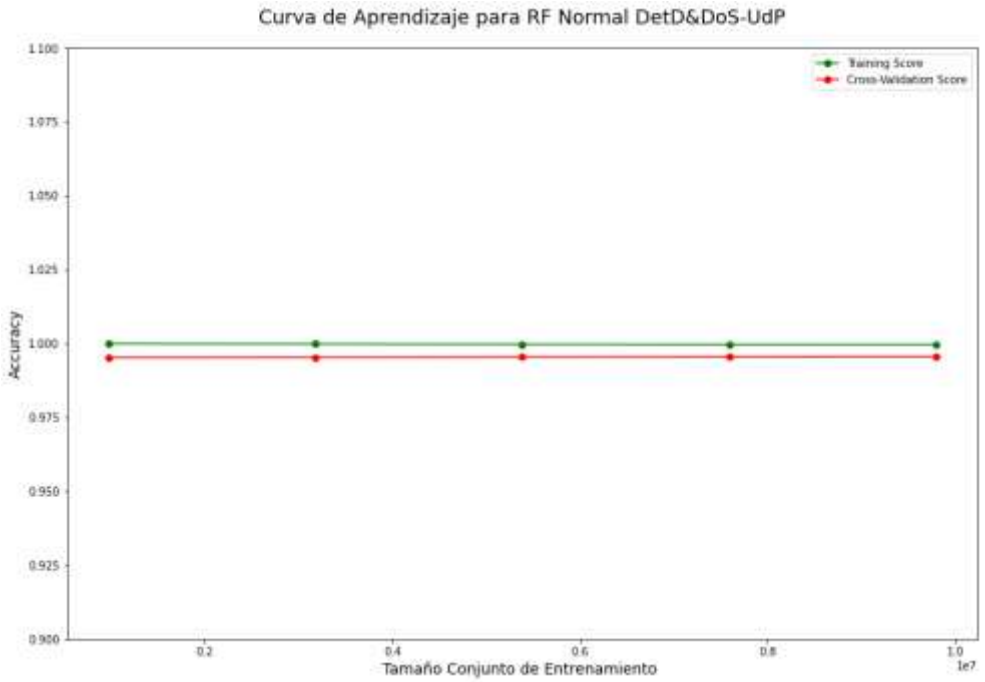


Figura 5-19 Curva de Aprendizaje Bosques Aleatorios con 5 divisiones y 5-KFold [Elaboración propia]

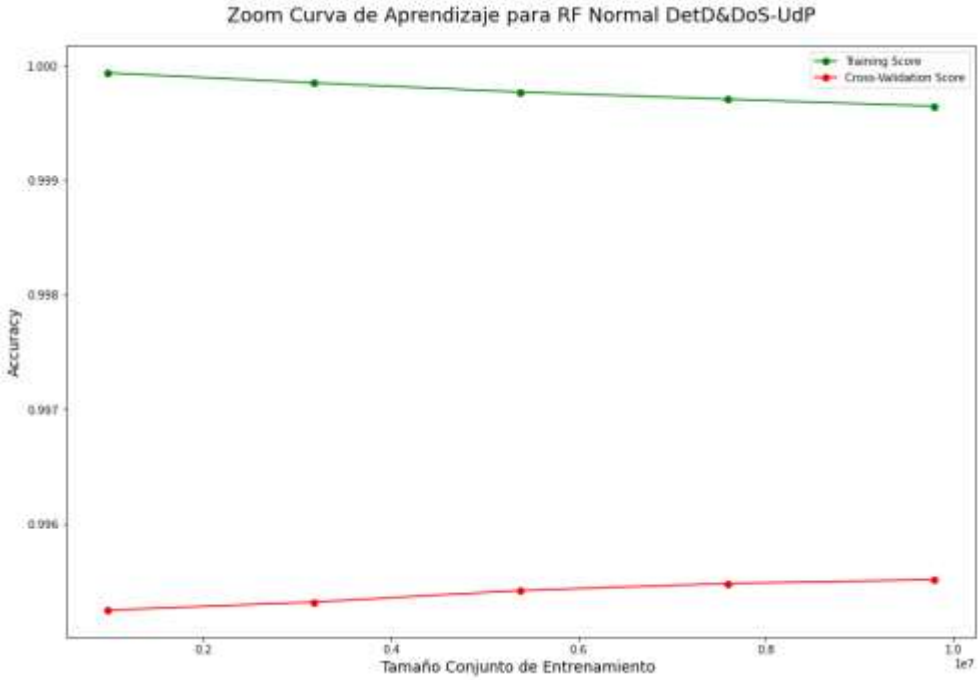


Figura 5-20 Zoom Curva de Aprendizaje Bosques Aleatorios con 5 divisiones y 5-KFold [Elaboración propia]

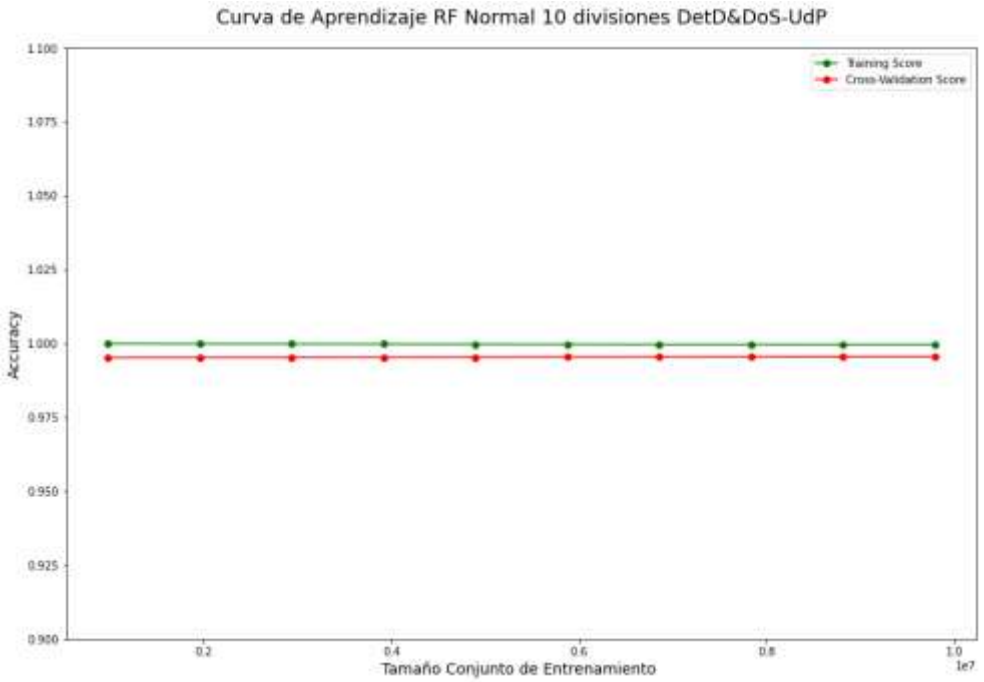


Figura 5-22 Curva de Aprendizaje Bosques Aleatorios con 10 divisiones y 3-KFold [Elaboración propia]

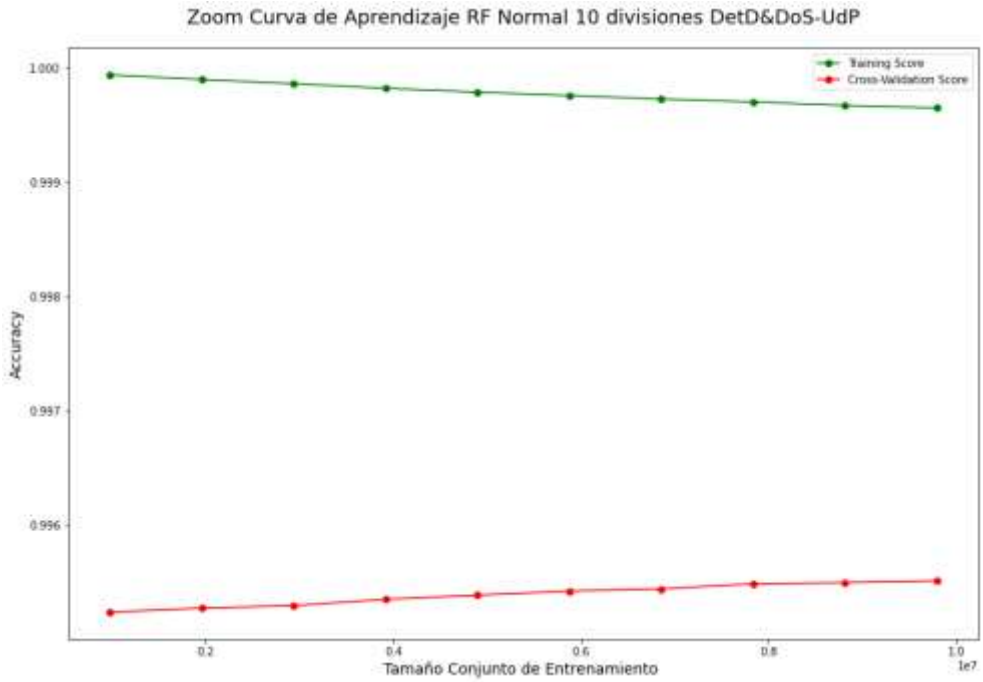


Figura 5-21 Zoom Curva de Aprendizaje Bosques Aleatorios con 10 divisiones y 3-KFold [Elaboración propia]

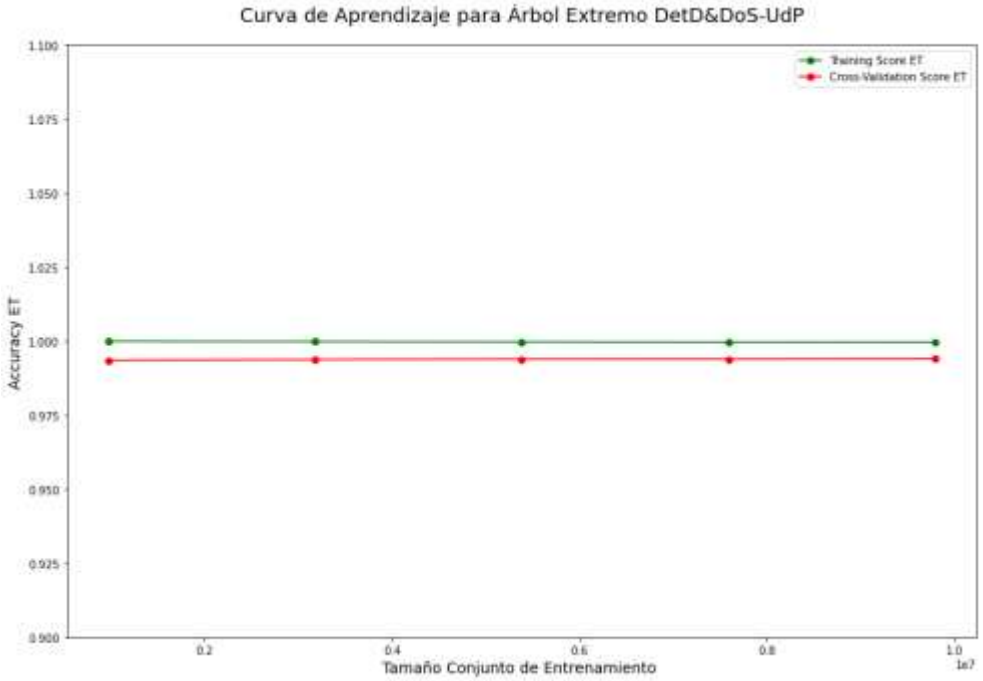


Figura 5-24 Curva de Aprendizaje Árbol Extremo con 5 divisiones y 5-KFold [Elaboración propia]

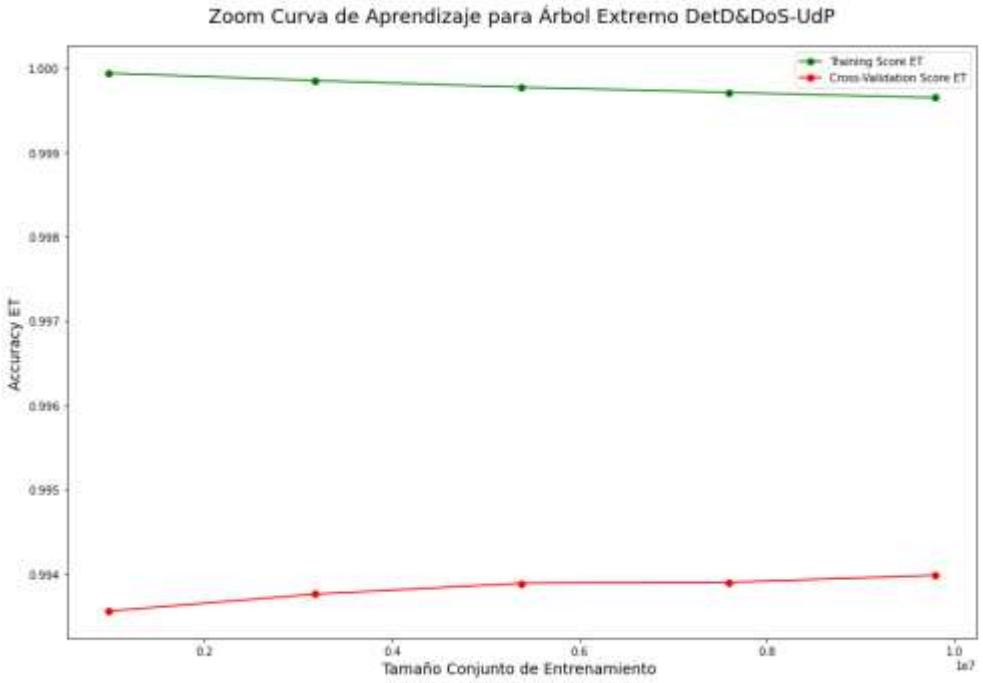


Figura 5-23 Zoom Curva de Aprendizaje Árbol Extremo con 5 divisiones y 5-KFold [Elaboración propia]

En caso de varianza alta: conseguir más ejemplos, reducir la cantidad de features y aumentar coeficiente de regularización. Si el bias es alto: más atributos (*features*) o modelo más complejo; no siempre el modelo de clasificación escogido será el funcional para el dataset, por esta razón es importante tener en cuenta escoger otro modelo.

En la Figura 5-19, se representa el aprendizaje del modelo de Bosques Aleatorios para 5 particiones del conjunto de datos, con 5 resultados de validación cruzada, que se toma como la estadística de la mediana (*mean*). En la Figura 5-20 es el zoom o a una escala más detallada.

En la Figura 5-22, se representa el aprendizaje del modelo de Bosques Aleatorios para 10 particiones del conjunto de datos, con 3 resultados de validación cruzada, mejorando la apreciación grafica de los datos de la figura anterior. Su Figura 5-21 para el zoom.

Y finalmente en la Figura 5-24, se representa el aprendizaje del modelo de Árbol Extremo para 5 particiones del conjunto de datos, con 5 resultados de validación cruzada. Y en la Figura 5-23 su ampliación o zoom en la escala de los ejes.

5.3.3. Validación cruzada: tres algoritmos de aprendizaje

La validación cruzada es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba. Consiste en repetir y calcular la media aritmética obtenida de las medidas de evaluación, para el caso del aplicado **cross_validate** se obtiene múltiples métricas, sobre diferentes particiones. Se utiliza en entornos donde el objetivo principal es la predicción y se quiere estimar la precisión de un modelo que se llevará a cabo a la práctica.

Para nuestro caso se ha decidido elegir 10 particiones, y lo de una forma estratificada, para que queden muestras de todas las categorías, y contrarrestar el desequilibrio de clases que presenta nuestro conjunto de datos DoS y DDoS.

Por cuestiones de tiempo, y gasto computacional, se ha decido hacer solo sobre los tres mejores algoritmos de clasificación. Pues en conclusión seria entrenar 10 modelos de aprendizaje, o 30 entrenamientos para los 3 algoritmos.

Se presencia en los resultados, que los datos de validación son consistentes y la división de las particiones de conjunto de entrenamiento y prueba, es indiferente.

En la Figura 5-25, se presenta el algoritmo de albo de decisión DT; en la Figura 5-26 los resultados para el Árbol Extremo ET y en la Figura 5-27 la validación cruzada para los Bosques Aleatorios RF.

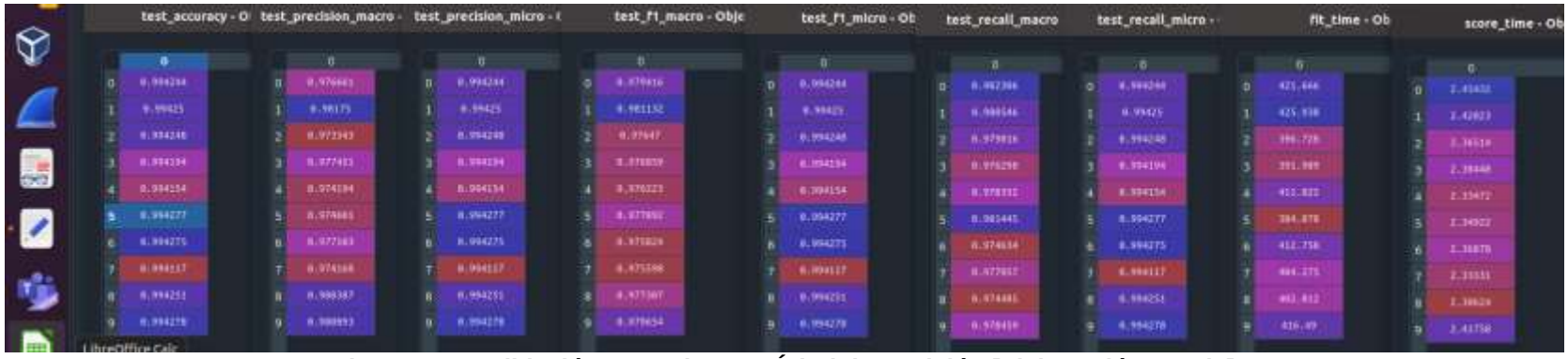


Figura 5-25 Validación Cruzada para Árbol de Decisión [Elaboración propia]

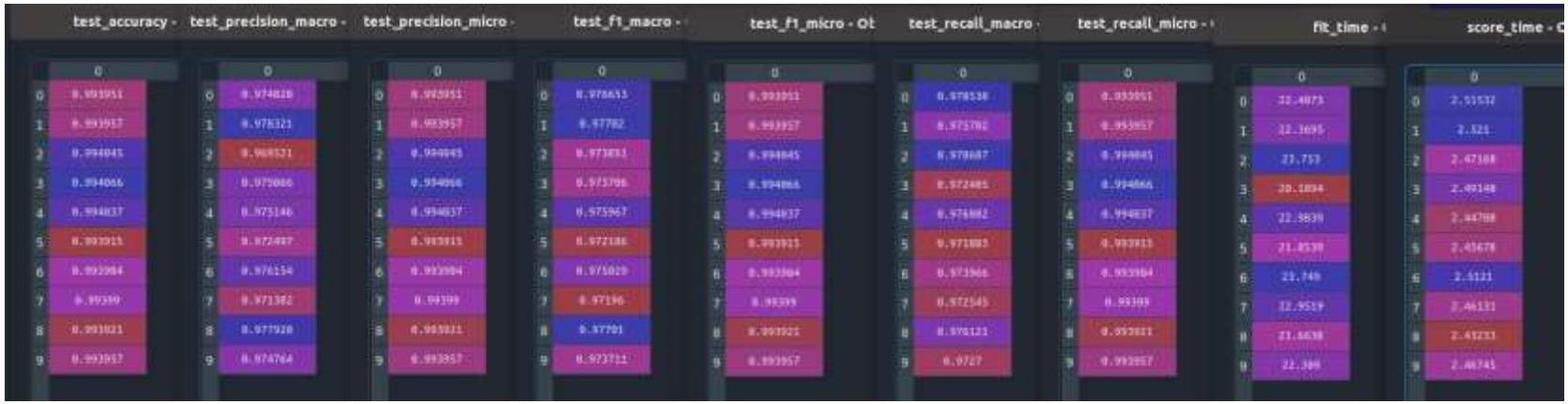


Figura 5-26 Validación Cruzada para Árbol Extremo [Elaboración propia]

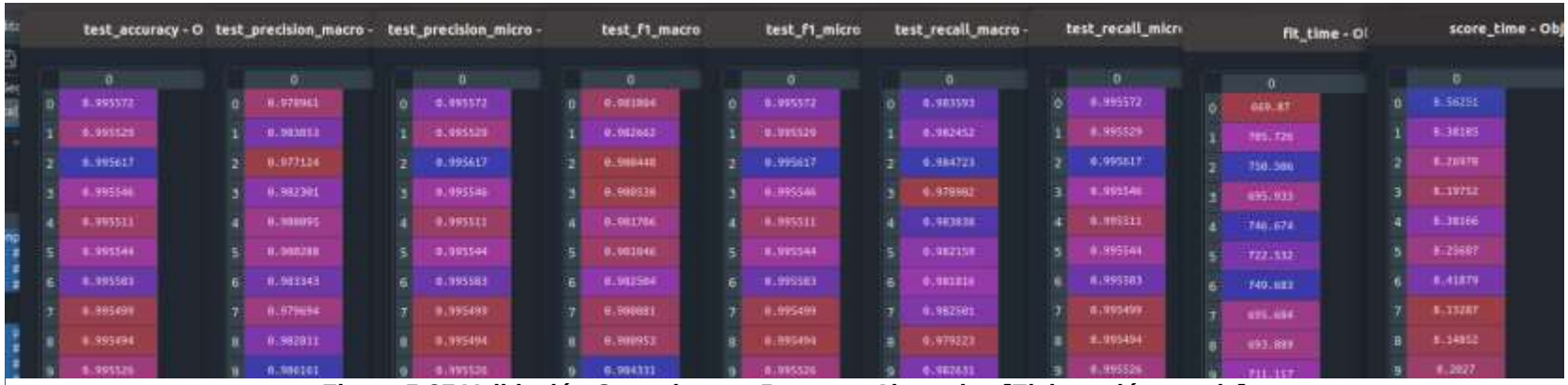


Figura 5-27 Validación Cruzada para Bosques Aleatorios [Elaboración propia]

5.3.4. Importancia de atributos: algoritmos de aprendizaje

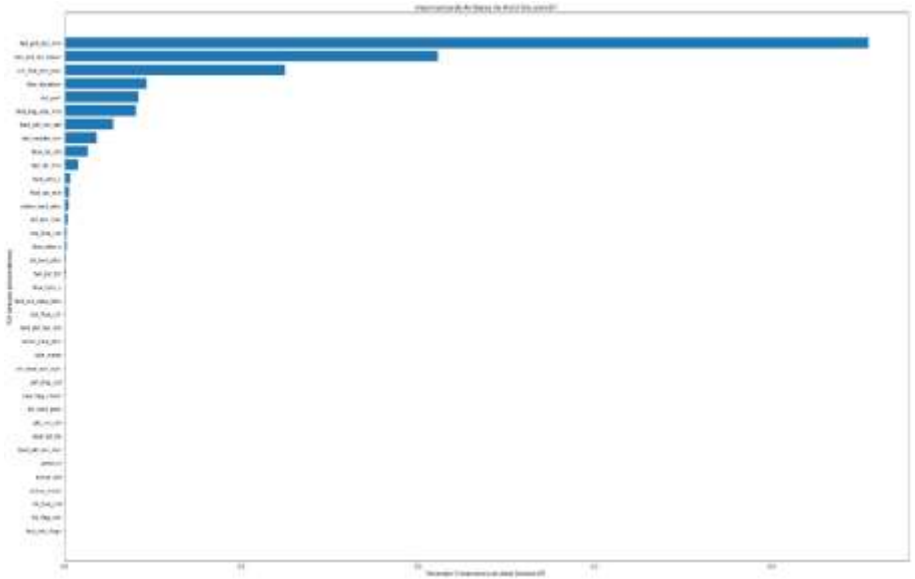


Figura 5-28 Importancia Atributos Árbol de Decisión [Elaboración propia]

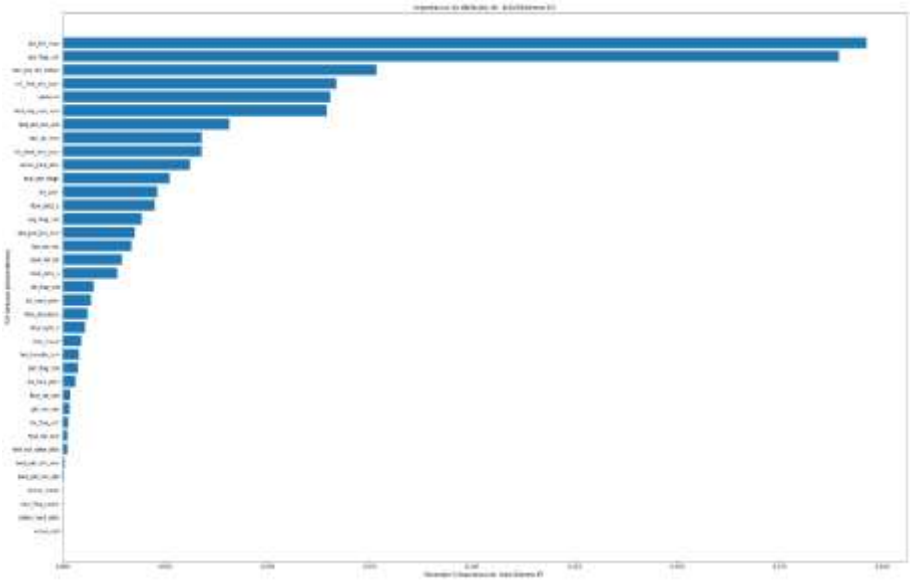


Figura 5-29 Importancia Atributos Árbol Extremo [Elaboración propia]

La interpretación de la importancia de atributos radica en la vital información de obtener una estimación de cuán significativo es cada atributo para realizar la predicción de la clasificación. Al comparar entre 6 algoritmos de aprendizaje se ha permitido entender mejor el problema de cómo detecta y clasifica, cada uno de los árboles y conjuntos, y ver cada una de sus deducciones y descifrar el modelo de caja negra, del aprendizaje automático.

Así tenemos, para los arboles unitarios se observa y se entiende el por qué el Árbol de Decisión en la Figura 5-28 es el algoritmo más flexible, y que ocupa menos tamaño de almacenamiento, pues de los 37 atributos, para el **DT** solo tiene peso representativo 10 a 13, siendo los otros 24 atributos casi nulos o cercanos al 0%. Mientras para el Árbol Extremo en la Figura 5-29 solo 5 atributos son muy pocos importantes y cercanos a la nulidad, mientras todos los otros 32 atributos son decisorios, al punto que, de ellos, 3 atributos superan el peso del 7,5% hasta casi el 20% de máximo en el **ET**.

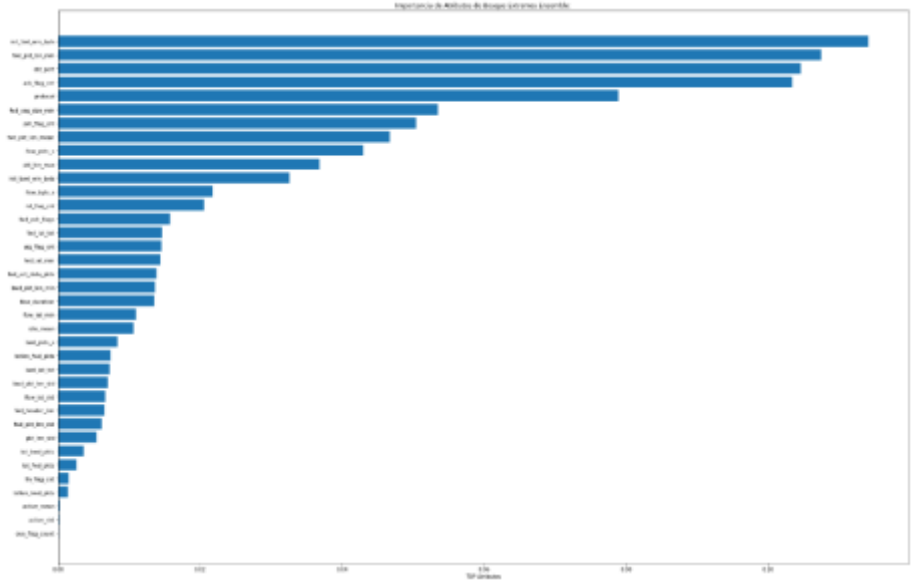


Figura 5-31 Importancia Atributos Bosques Extremos [Elaboración propia]

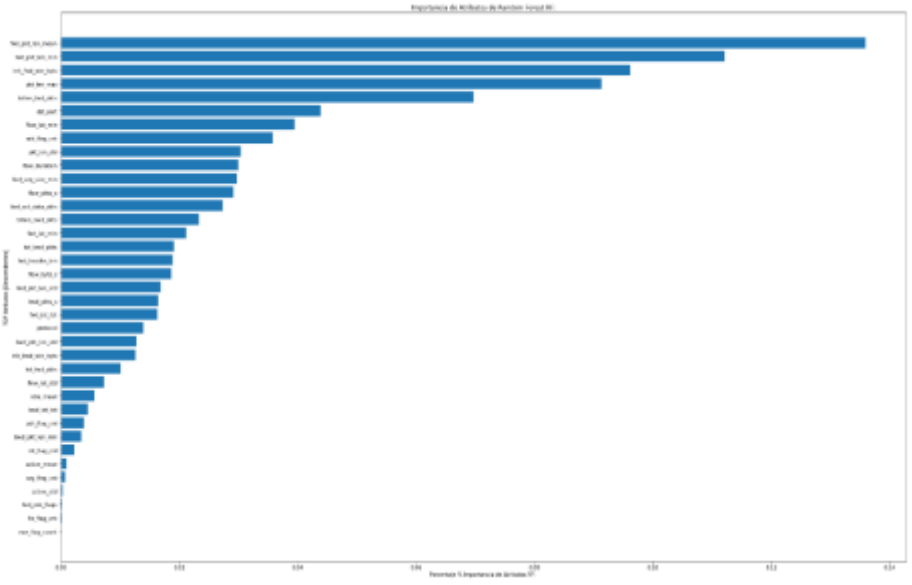


Figura 5-30 Importancia Atributos Bosques Aleatorios [Elaboración propia]

Para la importancia de atributos de los conjuntos de árboles, en la gráfica de Bosques Extremos Figura 5-31 se observa que 35 atributos tienen un peso, solo 2 atributos son cercanos al peso de 0%, mientras 4 atributos se llevan el protagonismo al superar el 10% y llegar hasta casi el 15%. Si se comparan con los otros bosques, se puede ver q suelen ser parecidos en su forma peso de importancias, en el sentido de usar también 34 atributos, es decir la mayoría; los Bosques Aleatorios Figura 5-30 en contraste usan solo 2 atributos superan el 10% y llegan casi al 14% los **RF.**

Hay que ser muy puntales que aún se pueden parecer estar formas, en su organización, el orden de los atributos en el eje y es totalmente diferente, disímil. Y en todos, como se verá en las siguientes graficas de los Gradientes de Refuerzo Extremo, de las 2 formas diferentes de implementar, ver figuras Figura 5-32 y Figura 5-33.

Justamente para finalizar el quinto y sexto resultados de atributos importantes, se aprecia de los XGB ambos tienen 2 atributos esenciales para ellos, y de forma similar 6 atributos de la cola, son de menor peso. Se puede indicar, que el valor de importancia es una puntuación de F1 y no son porcentajes.

Para resumir, y poder tener un *ranking* comparando la importancia de atributos de los resultados de los 6 algoritmos de aprendizaje, y hacer un resultado general. Para ello se realizó un archivo en Excel y se presenta como en la tabla como ya se ha indicado en la sección 5.2.1 y resume la relación ordenada de los atributos importantes en sus seis resultados, uno por cada uno de los seis algoritmos de aprendizaje.

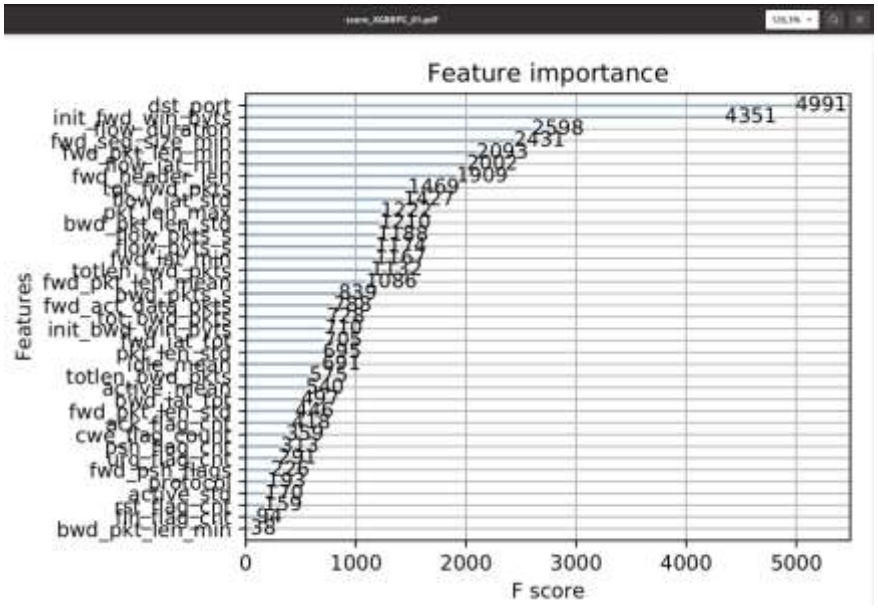


Figura 5-33 Importancia Atributos XGB API Scikit-L [Elaboración propia]

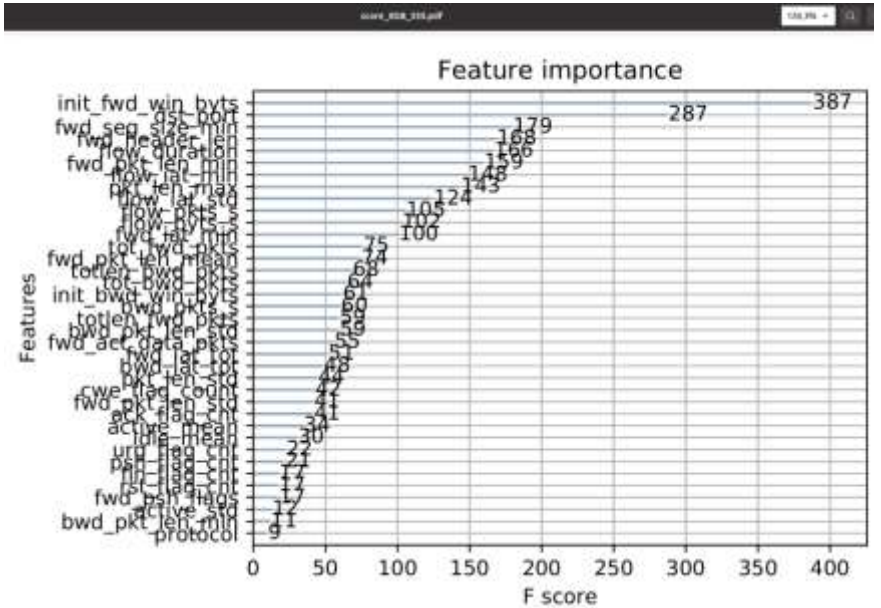


Figura 5-32 Importancia Atributos XGB API Learning [Elaboración propia]

Las indicaciones que se han de tener en cuenta en la tabla es, las filas de color en verde son los 5 atributos de la identificación del flujo, que solo sirven para reconocerlos y ser etiquetados, pero para atributos de entrenamiento no deben ser tenidos en cuenta. Se prosigue a las filas de color rojo, las cuales son los 9 atributos que se han excluido de la desviación estándar al ser nulos. Por último, el color azul indica los 32 atributos que han sido quitados por la correlación jerárquica de Spearman, para así quedar con los 37 atributos o características finales. Se ha creado el top de todos los 8 atributos finales, siendo en su orden descendente: **`init_fwd_win_byts`**, **`fwd_pkt_len_min`**, **`fwd_pkt_len_mean`**, **`dst_port`**, **`flow_duration`**, **`fwd_seg_size_min`**, **`pkt_len_max`**, **`ack_flag_cnt.`**

Nombre Normal	Nombre II	Selección Atributos STD	Selección Atributos Jerarquía - Spermanr	RF	ET	DT	EE	XGBRFC	XGB	PROM
Flow ID	Flow ID									
Src IP	Src IP									
Src Port	Src Port									
Dst IP	Dst IP									
Dst Port	**Dst Port**	**dst_port**	**dst_port**	**6**	**12**	**5**	**3**	**1**	**2**	**4,833**
Protocol	Protocol	protocol	protocol	22	5	32	5	33	37	**22,33**
Timestamp	Timestamp									**0**
Flow Duration	fl_dur	flow_duration	flow_duration	10	21	4	20	3	5	**10,5**
Tot Fwd Pkts	tot_fw_pk	tot_fwd_pkts	tot_fwd_pkts	25	26	17	32	8	13	**20,17**
Tot Bwd Pkts	tot_bw_pk	tot_bwd_pkts	tot_bwd_pkts	16	20	28	31	19	16	**21,67**
TotLen Fwd Pkts	tot_bw_pk	totlen_fwd_pkts	totlen_fwd_pkts	5	10	23	24	15	19	**16**
TotLen Bwd Pkts	tot_l_fw_pkt	totlen_bwd_pkts	totlen_bwd_pkts	14	36	13	34	24	15	**22,67**
Fwd Pkt Len Max	fw_pkt_l_max	fwd_pkt_len_max								**0**
Fwd Pkt Len Min	**fw_pkt_l_min**	**fwd_pkt_len_min**	**fwd_pkt_len_min**	**2**	**15**	**1**	**2**	**5**	**6**	**5,167**
Fwd Pkt Len Mean	**fw_pkt_l_avg**	**fwd_pkt_len_mean**	**fwd_pkt_len_mean**	**1**	**3**	**2**	**8**	**16**	**14**	**7,333**
Fwd Pkt Len Std	fw_pkt_l_std	fwd_pkt_len_std	fwd_pkt_len_std	19	7	22	29	27	26	**21,67**
Bwd Pkt Len Max	Bw_pkt_l_max	bwd_pkt_len_max								**0**
Bwd Pkt Len Min	Bw_pkt_l_min	bwd_pkt_len_min	bwd_pkt_len_min	30	32	31	19	37	36	**30,83**
Bwd Pkt Len Mean	Bw_pkt_l_avg	bwd_pkt_len_mean								**0**
Bwd Pkt Len Std	Bw_pkt_l_std	bwd_pkt_len_std	bwd_pkt_len_std	23	33	7	26	11	20	**20**
Flow Byts/s	fl_byt_s	flow_byts_s	flow_byts_s	18	22	19	12	13	11	**15,83**
Flow Pkts/s	fl_pkt_s	flow_pkts_s	flow_pkts_s	12	13	16	9	12	10	**12**
Flow IAT Mean	fl_iat_avg	flow_iat_mean								**0**
Flow IAT Std	fl_iat_std	flow_iat_std	flow_iat_std	26	27	9	27	9	9	**17,83**
Flow IAT Max	fl_iat_max	flow_iat_max								**0**
Flow IAT Min	fl_iat_min	flow_iat_min	flow_iat_min	7	30	12	21	6	7	**13,83**

Fwd IAT Tot	fw_iat_tot	fwd_iat_tot	fwd_iat_tot	21	16	18	15	21	22	**18,83**
Fwd IAT Mean	fw_iat_avg	fwd_iat_mean								**0**
Fwd IAT Std	fw_iat_std	fwd_iat_std								**0**
Fwd IAT Max	fw_iat_max	fwd_iat_max								**0**
Fwd IAT Min	fw_iat_min	fwd_iat_min	fwd_iat_min	15	8	10	17	14	12	**12,67**
Bwd IAT Tot	bw_iat_tot	bwd_iat_tot	bwd_iat_tot	28	17	30	25	26	23	**24,83**
Bwd IAT Mean	bw_iat_avg	bwd_iat_mean								**0**
Bwd IAT Std	bw_iat_std	bwd_iat_std								**0**
Bwd IAT Max	bw_iat_max	bwd_iat_max								**0**
Bwd IAT Min	bw_iat_min	bwd_iat_min								**0**
Fwd PSH Flags	fw_psh_flag	fwd_psh_flags	fwd_psh_flags	35	11	37	14	32	34	**27,17**
Bwd PSH Flags	bw_psh_flag									**0**
Fwd URG Flags	fw_urg_flag									**0**
Bwd URG Flags	bw_urg_flag									**0**
Fwd Header Len	fw_hdr_len	fwd_header_len	fwd_header_len	17	24	8	28	7	4	**14,67**
Bwd Header Len	bw_hdr_len	bwd_header_len								**0**
Fwd Pkts/s	fw_pkt_s	fwd_pkts_s								**0**
Bwd Pkts/s	bw_pkt_s	bwd_pkts_s	bwd_pkts_s	20	18	11	23	17	18	**17,83**
Pkt Len Min	pkt_len_min	pkt_len_min								**0**
Pkt Len Max	**pkt_len_max**	**pkt_len_max**	**pkt_len_max**	**4**	**1**	**14**	**10**	**10**	**8**	**7,833**
Pkt Len Mean	pkt_len_avg	pkt_len_mean								**0**
Pkt Len Std	pkt_len_std	pkt_len_std	pkt_len_std	9	28	29	30	22	24	**23,67**
Pkt Len Var	pkt_len_va	pkt_len_var								**0**
FIN Flag Cnt	fin_cnt	fin_flag_cnt	fin_flag_cnt	36	29	36	33	36	32	**33,67**
SYN Flag Cnt	syn_cnt	syn_flag_cnt								**0**
RST Flag Cnt	rst_cnt	rst_flag_cnt	rst_flag_cnt	31	19	35	13	35	33	**27,67**
PSH Flag Cnt	pst_cnt	psh_flag_cnt	psh_flag_cnt	29	25	26	7	30	31	**24,67**
ACK Flag Cnt	ack_cnt	ack_flag_cnt	ack_flag_cnt	8	2	21	4	28	27	**15**
URG Flag Cnt	urg_cnt	urg_flag_cnt	urg_flag_cnt	33	14	15	16	31	30	**23,17**
CWE Flag Count	cwe_cnt	cwe_flag_count	cwe_flag_count	37	35	27	37	29	25	**31,67**

ECE Flag Cnt	ece_cnt	ece_flag_cnt								0
Down/Up Ratio	down_up_ratio	down_up_ratio								0
Pkt Size Avg	pkt_size_avg	pkt_size_avg								0
Fwd Seg Size Avg	fw_seg_avg	fwd_seg_size_avg								0
Bwd Seg Size Avg	bw_seg_avg	bwd_seg_size_avg								0
Fwd Byts/b Avg	fw_byt_blk_avg									0
Fwd Pkts/b Avg	fw_pkt_blk_avg									0
Fwd Blk Rate Avg	fw_blk_rate_avg									0
Bwd Byts/b Avg	bw_byt_blk_avg									0
Bwd Pkts/b Avg	bw_pkt_blk_avg									0
Bwd Blk Rate Avg	bw_blk_rate_avg									0
Subflow Fwd Pkts	subfl_fw_pk	subflow_fwd_pkts								0
Subflow Fwd Byts	subfl_fw_byt	subflow_fwd_byts								0
Subflow Bwd Pkts	subfl_bw_pkt	subflow_bwd_pkts								0
Subflow Bwd Byts	subfl_bw_byt	subflow_bwd_byts								0
Init Fwd Win Byts	**fw_win_byt**	**init_fwd_win_byts**	**init_fwd_win_byts**	**3**	**4**	**3**	**1**	**2**	**1**	**2,333**
Init Bwd Win Byts	bw_win_byt	init_bwd_win_byts	init_bwd_win_byts	24	9	25	11	20	17	**17,67**
Fwd Act Data Pkts	Fw_act_pkt	fwd_act_data_pkts	fwd_act_data_pkts	13	31	20	18	18	21	**20,17**
Fwd Seg Size Min	**fw_seg_min**	**fwd_seg_size_min**	**fwd_seg_size_min**	**11**	**6**	**6**	**6**	**4**	**3**	**6**
Active Mean	atv_avg	active_mean	active_mean	32	34	34	35	25	28	**31,33**
Active Std	atv_std	active_std	active_std	34	37	33	36	34	35	**34,83**
Active Max	atv_max	active_max								0
Active Min	atv_min	active_min								0
Idle Mean	idl_avg	idle_mean	idle_mean	27	23	24	22	23	29	**24,67**
Idle Std	idl_std	idle_std								0
Idle Max	idl_max	idle_max								0
Idle Min	idl_min	idle_min								0
Label	label									

Tabla 5-4 Atributos Importantes para los 6 algoritmos de Pódium [Elaboración propia]

5.3.5. Diagrama de árbol unitarios: DT y ET

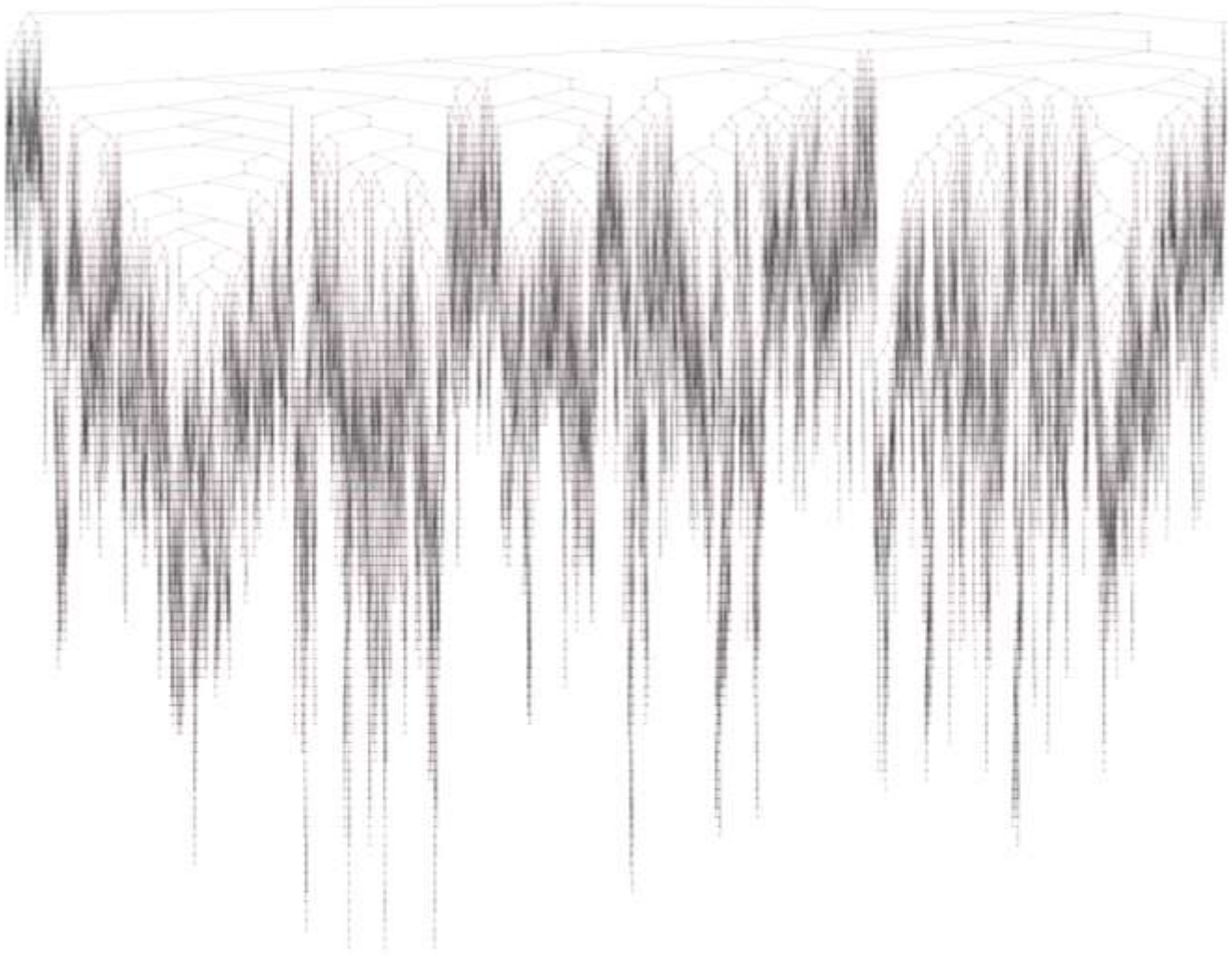


Figura 5-34 Diagrama de Árbol para DT [Elaboración propia]

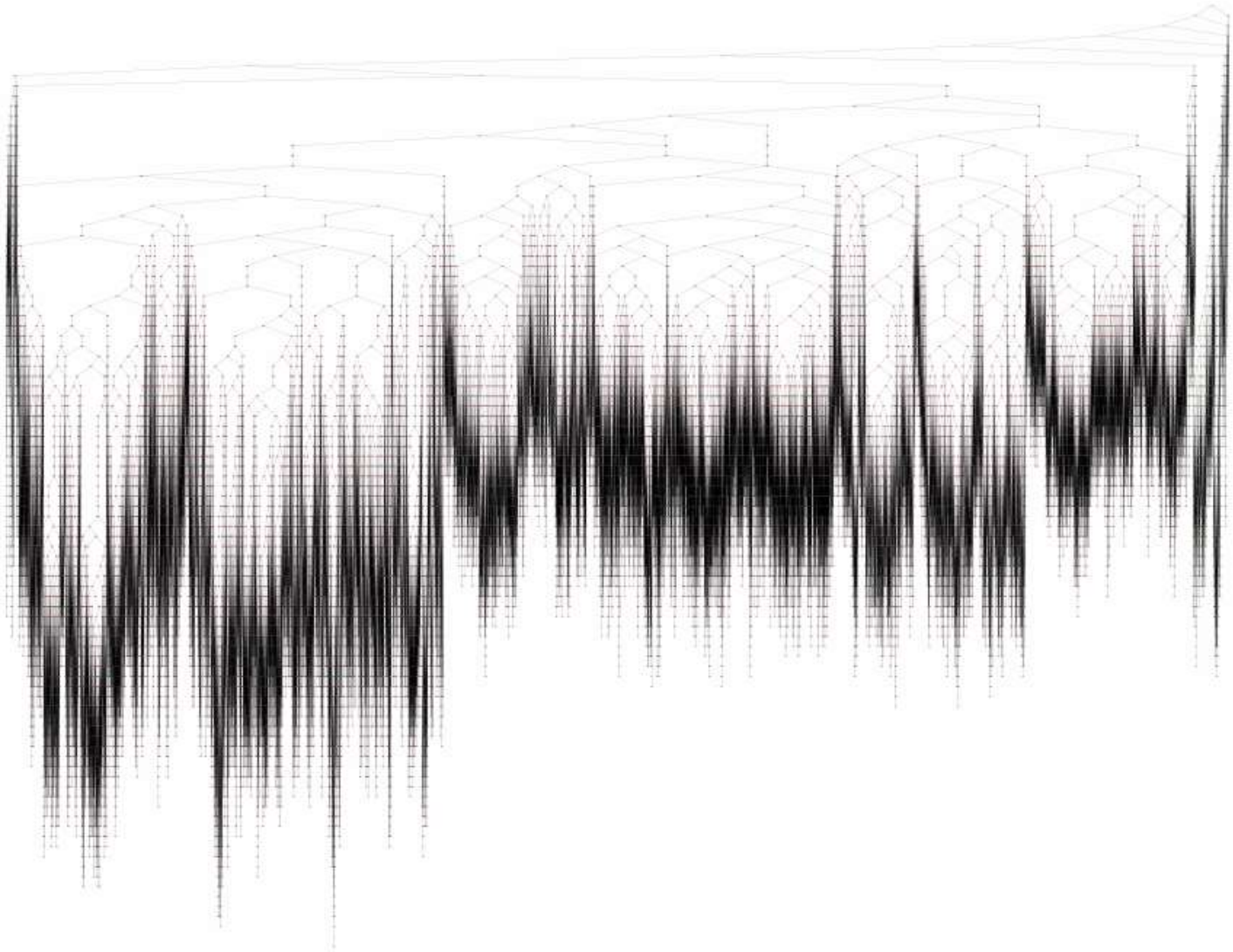


Figura 5-35 Diagrama de Árbol para ET [Elaboración propia]

En las figuras Figura 5-34 y Figura 5-35 se representa el objeto de árbol, obtenidos cómo atributos de la función tanto de DT, como de ET, pues son los únicos que la implementan, respecto a la no funcionalidad en la clase de conjunto de árboles (*ensemble*).

Se desea poner en manifiesto el gran poder que conllevan los modelos de aprendizaje automático al realizar ellos automáticamente el discernimiento y crear ellos mismos unas complejas estructuras de árbol, tan complicadas y tediosos, que un humano debería acercarse a 139.168 instancias de Hojas para el DT (7.735 págs.). Mientras para el ET, se debería acercar a 427.752 instancias de Hojas para unas 23.802 páginas. Es por eso la dificultad de los 2 anteriores gráficos de además de ser representados por el camino más sencillo de representación de texto **tree.export_text** y el objeto del diagrama de árbol **tree.plot_tree** que se ha mostrado; no se ha podido realizar ni el método ni **export_graphviz** ni el **dtreeviz package.**

5.3.6. Diagrama de árbol XGBoost

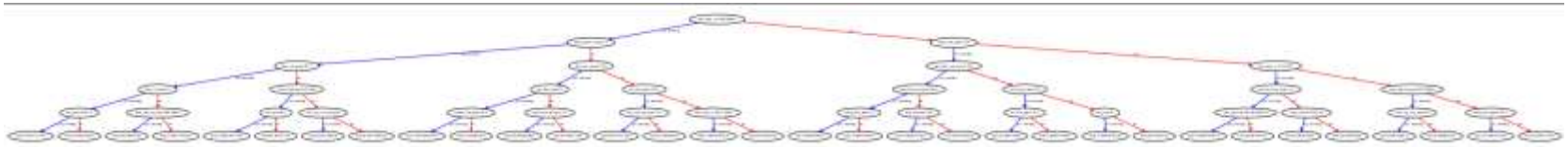


Figura 5-36 Diagrama de Árbol para XGBRFC [Elaboración propia]

Ahora bien, en estos diagramas de árbol de los dos algoritmos de XGB de entrenamiento, tanto la forma de obtener, como los mismos objetos generados son mucho más complejos, pero realizados en una forma sencilla y óptima. Tan sencilla su ejecución que son efectuados en una API de Graficación (*Plotting API*) gracias al poder de su

librería **xgboost.plot_importance** al ser trazado la importancia basada en árboles ajustados, como **xgboost.plot_tree** para sea trazado el árbol determinado por el algoritmo de XGB.

La Figura 5-36 interpreta dicho diagrama para el clasificador XGBRFC contiene 32 instancias de hoja (*leaf*), para un total de 63 instancias totales.

Ahora bien, la Figura 5-37 señala el objeto de árbol para el clasificador XGB por el API Aprendizaje, realizar la *dMatrix*, y el método *train*, cuyo diagrama tiene 29 instancias de hoja, y 57 instancias totales, siendo esta figura de árbol, más sencilla, pero con resultados un poco por debajo en comparación de XGBRFC.

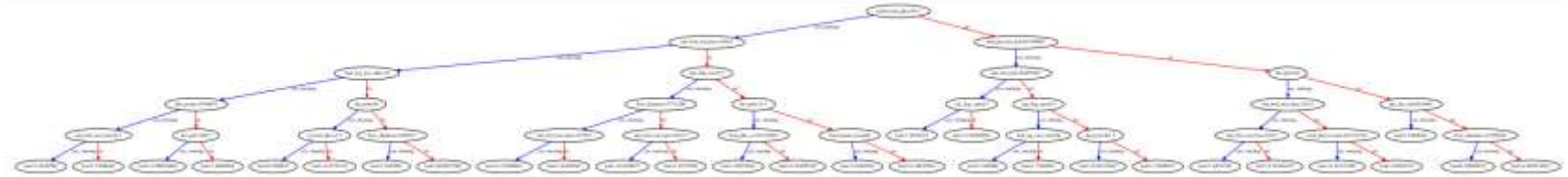


Figura 5-37 Diagrama de Árbol para XGB [Elaboración propia]

5.3.7. Guardar el modelo: conservar el entrenamiento

A continuación, se muestra en la Figura 5-38 siete (7) clasificadores almacenados, el pódium elegido como más importante y representativo. Resaltar que el binario almacenado de bosques aleatorios tiene un peso de 1,6GB para la versión normal (más de 12 millones) y de 1,8GB para la versión total (más de 15 millones). El estimador que mayor complejidad requiere son los Bosques Extremos, y el estimador más flexible es el árbol de Decisión, para un tamaño de almacenamiento de 6,1GB y 20,0MB respectivamente.

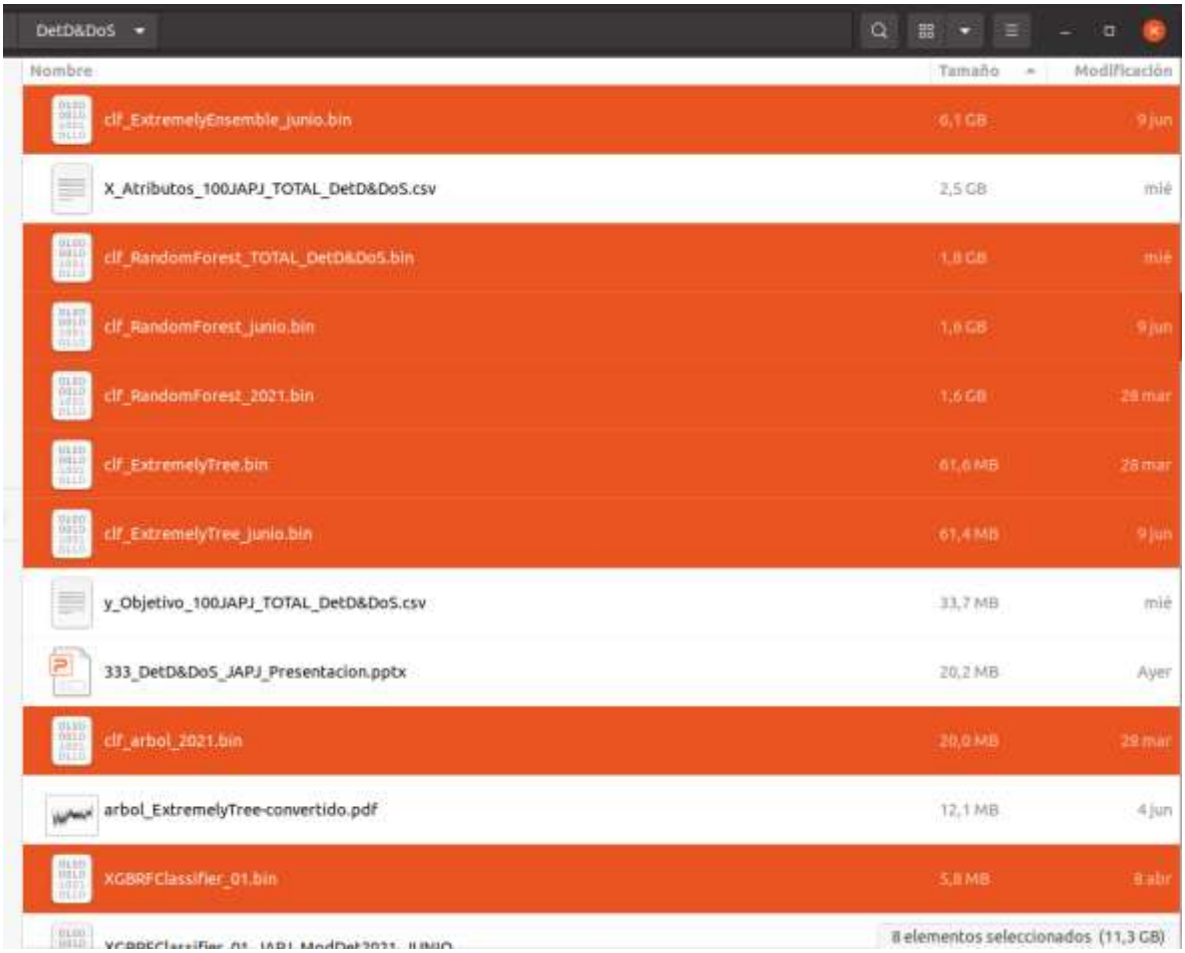


Figura 5-38 Clasificadores entrenados almacenados [Elaboración propia]

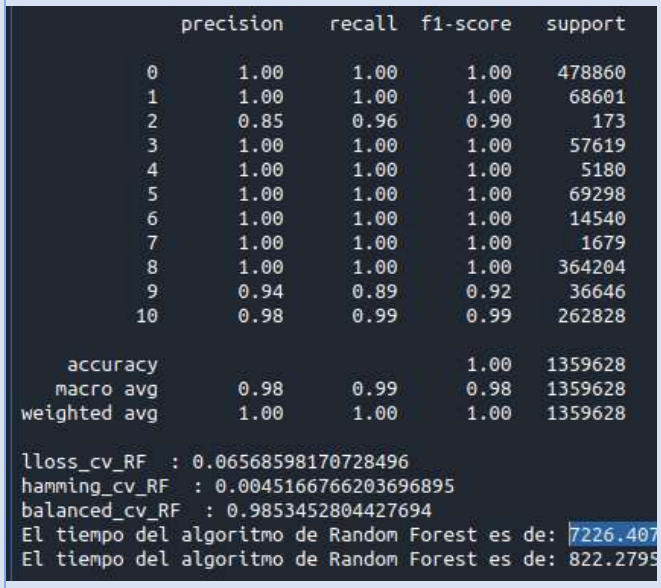
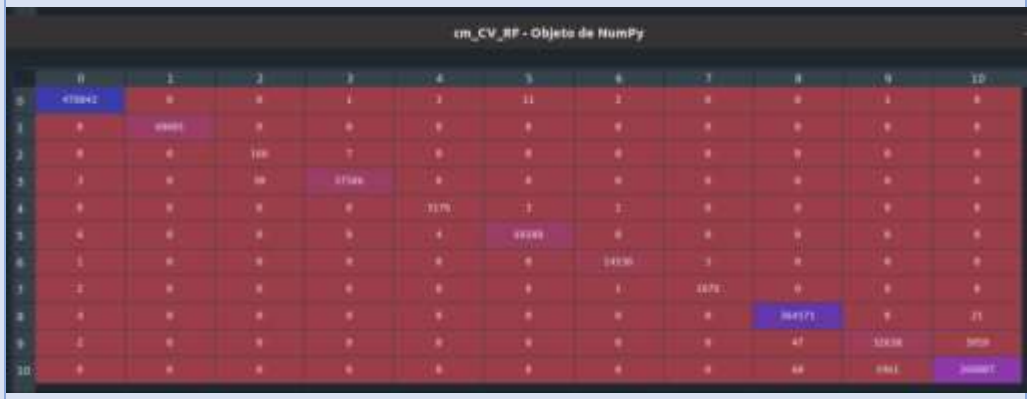
5.4. Evaluación de Prueba: Métricas de rendimiento

A continuación, en la Tabla 5-5, se visualizan los resultados de la evaluación de la prueba, perteneciente al conjunto de prueba siendo el 10% estratificado del conjunto total, los mejores cinco (5) algoritmos del pódium mencionados muestran el siguiente desempeño.

Si apreciamos los 5 reportes de clasificación multiclase, mostradas a la izquierda de la tabla podemos ver los similares que son los porcentajes desde el **97%** al **100%** en las diferentes métricas, siendo una constante la dificultad de clasificar la clase dos, **2** el cual es el ataque **LOIC** sobre **UDP** y cuya dificultad radica en el escaso y mínimo número de muestras, apenas **173 flujos** sobre los 13 millones casi 600 mil flujos, obteniendo porcentajes mínimos de métricas desde el **80%** en el XGB del API de aprendizaje, hasta el **85%** en el elegido como medalla de oro RF, para nuestro caso.

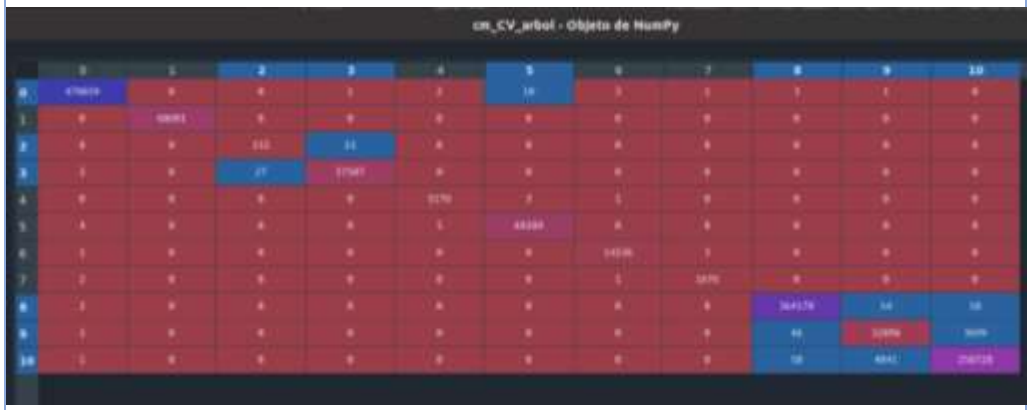
De forma personal, me parece mejor, los resultados brindados en las matrices de confusión multiclase, expuestas a la derecha de la tabla, ya que se aprecia su comportamiento tanto de la eficacia de clasificación contenida en el número de registros en su diagonal principal; como todas las clasificaciones incorrectas cometidas que se salen de dicha diagonal principal. Asimismo, se contrasta los errores entre clasificaciones entre clases de ataques DoS y DDoS, como el número de Falsos Positivos y Falsos Negativos que procede cada algoritmo de Árboles y Bosques, con los cuales nos casamos en nuestra investigación. Para el caso del mejor, el RF presenta errores de tan solo **16** Flujos de **FP** y **18** de **FN**, para el caso de la clasificación de los **flujos Benignos**.

Tabla 5-5 Evaluación de Conjunto de Prueba 10%: Reportes y Matrices [Elaboracion propia]

Algoritmo Aplicado	Reporte de Clasificación Evaluación (Conjunto Prueba 10%)	Matriz de Confusión Evaluación (Conjunto Prueba 10%)
Randon Forest -RF		

Árbol Decisión - DT

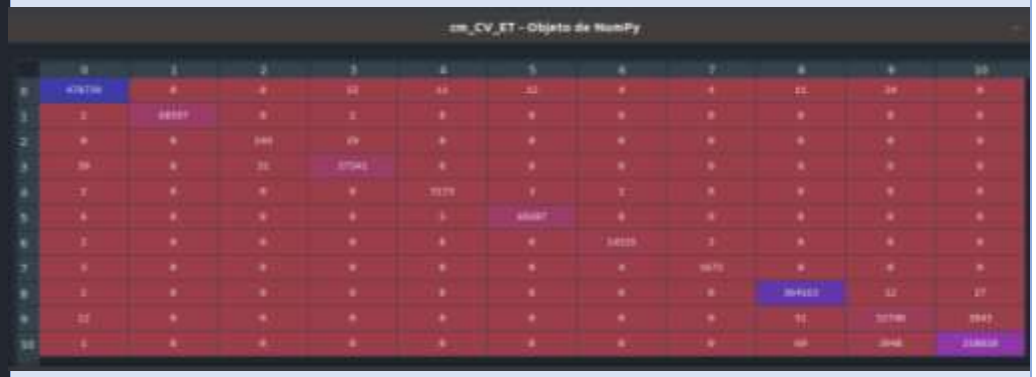
	precision	recall	f1-score	support
0	1.00	1.00	1.00	478860
1	1.00	1.00	1.00	68601
2	0.85	0.88	0.86	173
3	1.00	1.00	1.00	57619
4	1.00	1.00	1.00	5180
5	1.00	1.00	1.00	69298
6	1.00	1.00	1.00	14540
7	1.00	1.00	1.00	1679
8	1.00	1.00	1.00	364204
9	0.89	0.90	0.89	36646
10	0.99	0.98	0.99	262828
accuracy			0.99	1359628
macro avg	0.97	0.98	0.98	1359628
weighted avg	0.99	0.99	0.99	1359628



Árbol Extra - ET

	precision	recall	f1-score	support
0	1.00	1.00	1.00	478860
1	1.00	1.00	1.00	68601
2	0.82	0.83	0.83	173
3	1.00	1.00	1.00	57619
4	1.00	1.00	1.00	5180
5	1.00	1.00	1.00	69298
6	1.00	1.00	1.00	14540
7	1.00	1.00	1.00	1679
8	1.00	1.00	1.00	364204
9	0.89	0.89	0.89	36646
10	0.99	0.98	0.98	262828
accuracy			0.99	1359628
macro avg	0.97	0.97	0.97	1359628
weighted avg	0.99	0.99	0.99	1359628

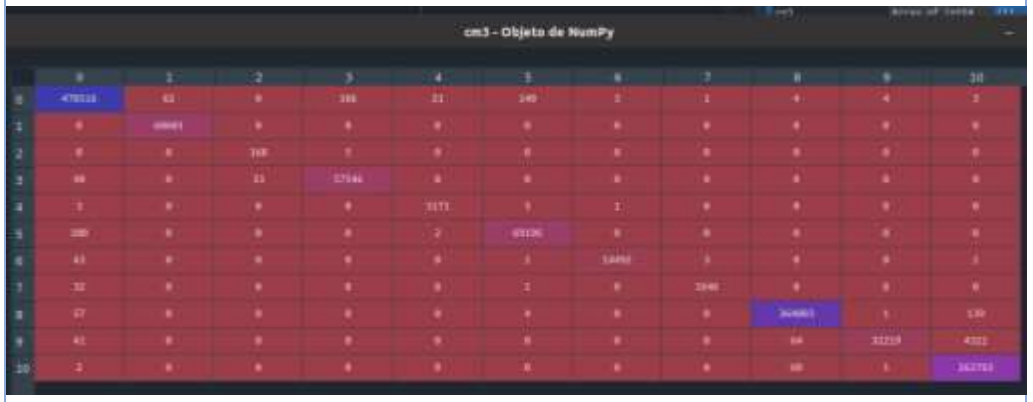
```
lloss_cv_ET  : 0.19683198565674698
hanming_cv_ET  : 0.006050919818418732
balanced_cv_ET  : 0.97297223900737
El tiempo del algoritmo de  ExtremelyTree es de: 250.79512095451355
El tiempo del algoritmo de ExtremelyTree es de: 27.492543697357178
```



xgboost.Dmatrix xgboost.train

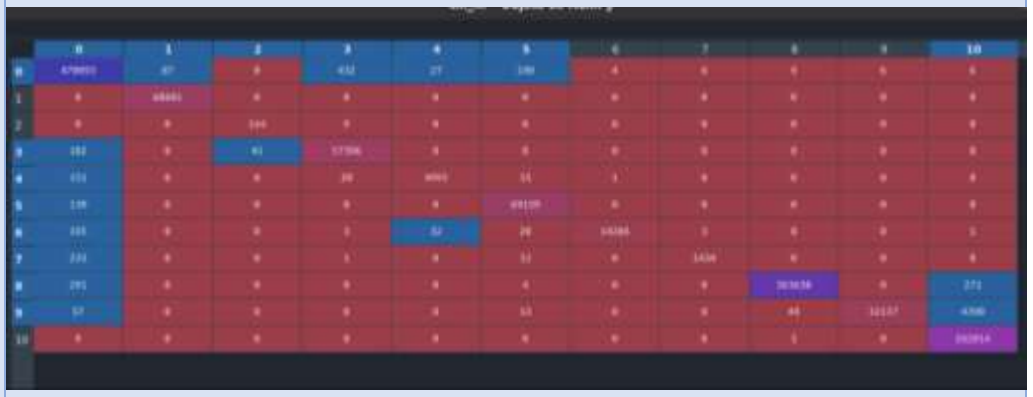
```
El tiempo del ENTRENO XGB xgboost es de: 1180.0635228157043
El tiempo de PREDECIR XGB xgboost es de: 0.8269808292388916
                              precision    recall  f1-score   support

                  Benigno [0]       1.00      1.00      1.00    478860
           DDOS attack-HOIC [1]       1.00      1.00      1.00     68601
     DDoS attacks-LOIC-UDP [2]       0.84      0.97      0.90       173
   DDoS attacks-LOIC-HTTP [3]        1.00      1.00      1.00     57619
    DoS attacks-GoldenEye [4]        1.00      1.00      1.00      5180
          DoS attacks-Hulk [5]       1.00      1.00      1.00     69298
DoS attacks-SlowHTTPTest [6]        1.00      1.00      1.00     14540
   DoS attacks-Slowloris [7]        1.00      0.98      0.99      1679
               SYN_Flood [8]        1.00      1.00      1.00    364204
                  UDPLag [9]        1.00      0.88      0.94     36646
              UDP_Flood [10]        0.98      1.00      0.99    262828

                  accuracy                              1.00   1359628
                 macro avg        0.98      0.98      0.98   1359628
              weighted avg        1.00      1.00      1.00   1359628


In [10]:
```

Terminal de IPython | Historial



xgboost.XGBClassifier

```
In [4]: y_pred_RF = xgb_RF.predict(X_test)
   ...: cm_RF = confusion_matrix(y_test.etiqueta_cat, y_pred_RF)
   ...: print(classification_report(y_test.etiqueta_cat, y_pred_RF,
target_names=cn))
                              precision    recall  f1-score   support

                  Benigno [0]       1.00      1.00      1.00    478860
           DDOS attack-HOIC [1]       1.00      1.00      1.00     68681
     DDoS attacks-LOIC-UDP [2]       0.80      0.95      0.87       173
   DDoS attacks-LOIC-HTTP [3]        0.99      1.00      0.99     57619
    DoS attacks-GoldenEye [4]        0.99      0.96      0.98      5180
          DoS attacks-Hulk [5]       1.00      1.00      1.00     69298
DoS attacks-SlowHTTPTest [6]        1.00      0.98      0.99     14540
   DoS attacks-Slowloris [7]        0.99      0.85      0.92      1679
               SYN_Flood [8]        1.00      1.00      1.00    364204
                  UDPLag [9]        1.00      0.88      0.93     36646
              UDP_Flood [10]        0.98      1.00      0.99    262828

                  accuracy                              0.99   1359628
                 macro avg        0.98      0.97      0.97   1359628
              weighted avg        0.99      0.99      0.99   1359628


In [5]:
```



Continuando analizando la matriz de confusión de RF su mayor debilidad es presentar errores en la clasificación de los flujos maliciosos de **DDoS UDPLag**, esto es para la clase categorizada para **9**. En ella **3959 flujos** siendo flujos reales de UDPLag, son mal clasificados como flujos DDoS UDP y a su vez, **1961 flujos** fueron calificados como UDPLag, pero en realidad son flujos de ataques UDP. Esta situación se presenta por el desequilibrio de muestras de clases minoritarias en el aprendizaje, como es el caso de dicho ataque DDoS UDPLag.

Ahora, pasamos a la Tabla 5-6, donde se aprecian las gráficas de las curvas tanto ROC a la derecha, como de P-R a la izquierda. Estas curvas nos indican de manera visual la relación entre la **precisión** y la **sensibilidad** de nuestro modelo, a la vez que sirven para **comparar el rendimiento** de los cinco distintos algoritmos de clasificación. En la **curva P-R**, la **sensibilidad** (***recall***) de nuestro modelo es el *ratio* de positivos detectado en el conjunto de datos por nuestro clasificador es el *eje x,* y **precisión** (***precision***) es el *eje y*. *Recall* tiene el mismo significado que **TPR**, y **precisión** se refiere a la proporción de muestras positivas clasificadas correctamente con respecto al total de muestras positivas.

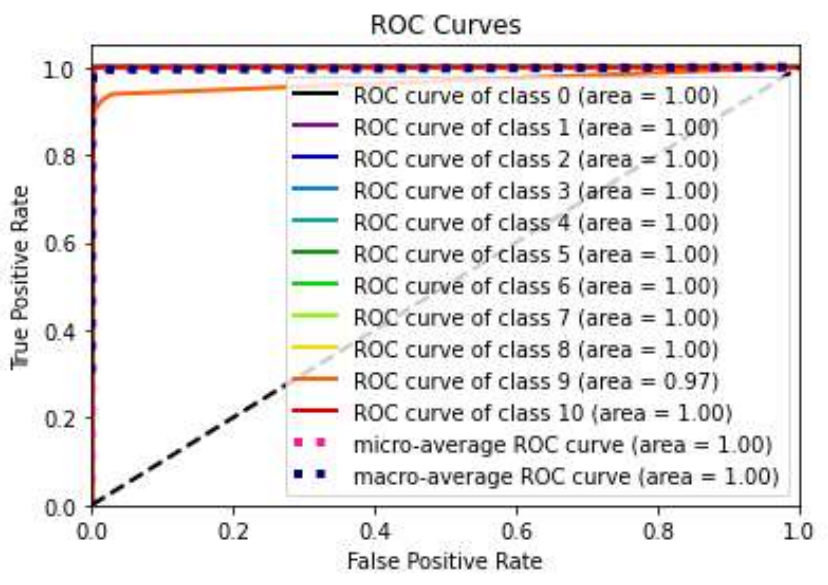
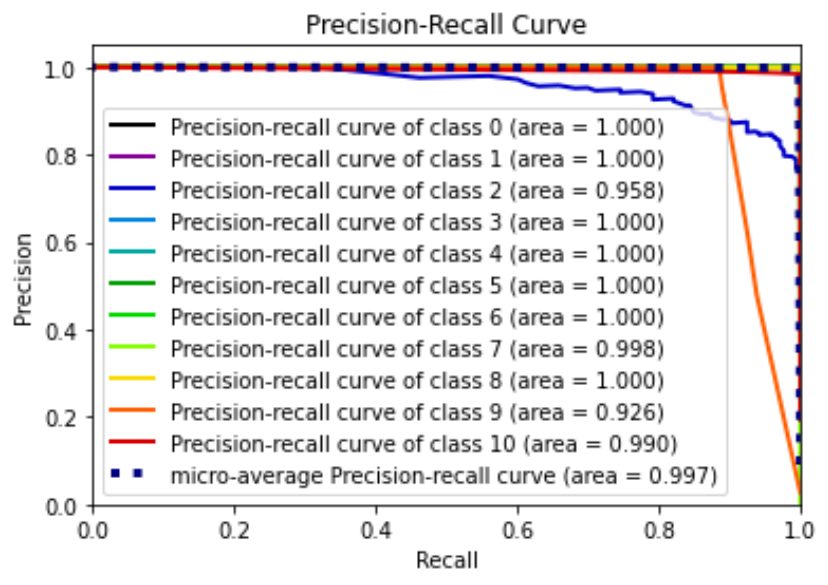
Con observarla la curva P-R del algoritmo de RF nos permite ver para el ataque de **DDoS LOIC-UDP** a partir de 0,4 de *recall* tenemos una degradación de la precisión y para el ataque de **DDoS UDP-Lag** a partir de 0,9 de recall. Lo ideal sería una curva que se acerque lo máximo posible a la **esquina superior derecha** (alta precisión y alto *recall*) como presentan mayor similitud los resultados del entrenamiento vistos en la Tabla 5-3. Por lo general, se debe usar la curva P-R cuando tengamos problemas de conjunto de datos no balanceados, es decir, cuando la clase de ataques maliciosos de denegación ocurre pocas veces.

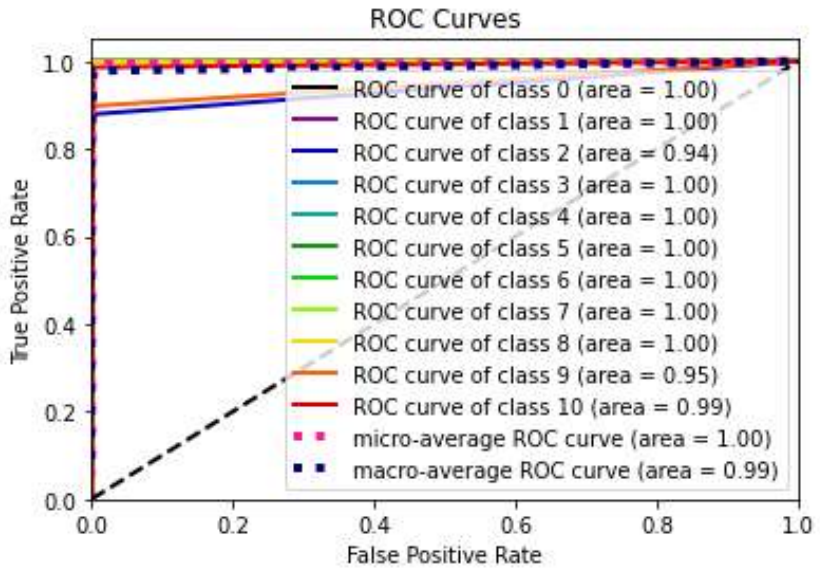
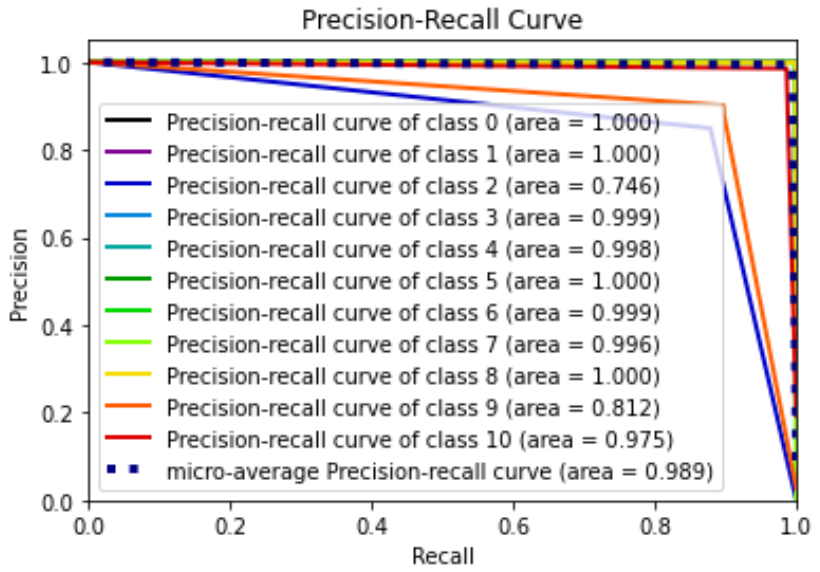
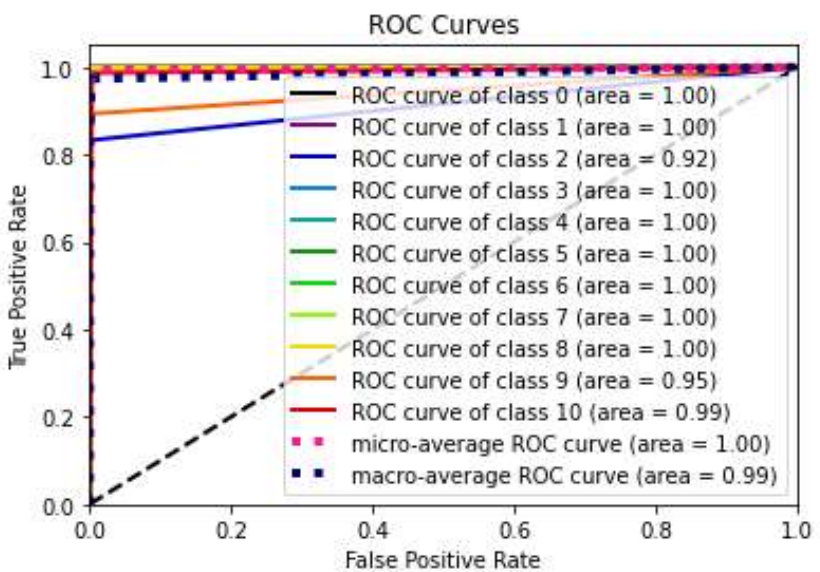
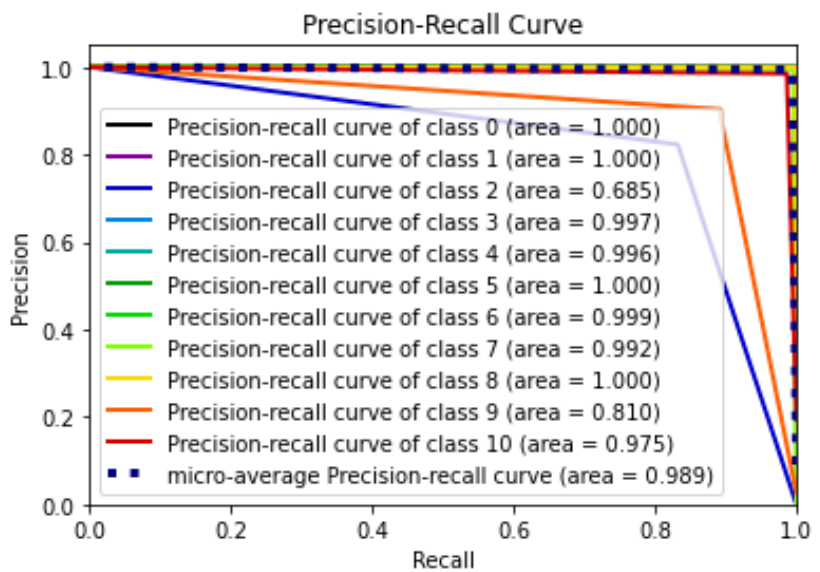
Ahora bien, **curva ROC** es parecida a la curva P-R pero cambiando algunos valores. Relaciona el *recall* con el *ratio* de falsos positivos. O, en otras palabras, la tasa de falsos positivos **FPR** es el *eje x* versus la tasa de verdadero positivos **TPR**

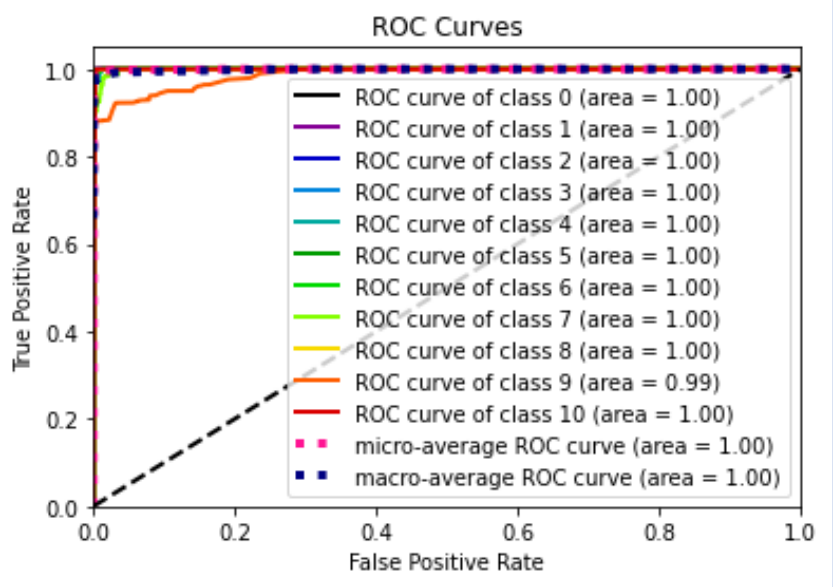
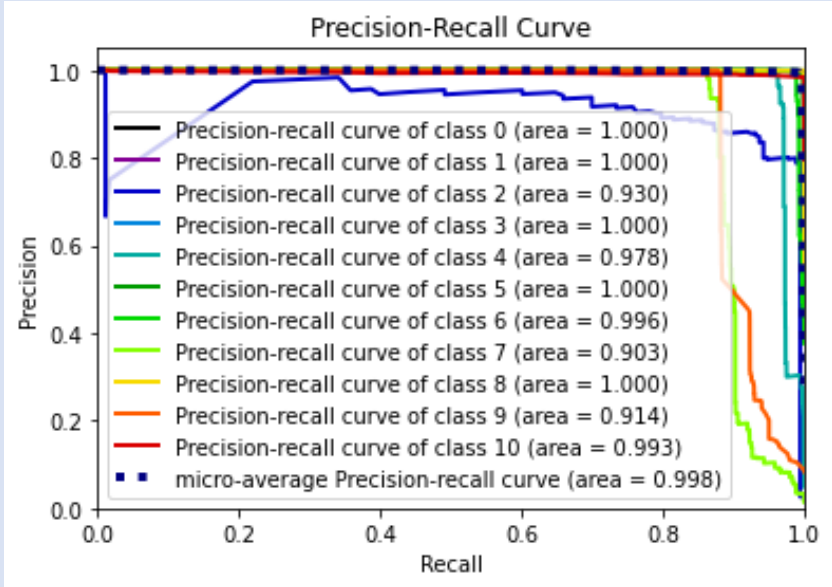
es el *eje y*. FPR se refiere a la probabilidad de que se prediga **incorrectamente** como una muestra positiva entre las muestras negativas reales. TPR se refiere a la probabilidad de que la predicción sea **correcta** en la muestra positiva real.

Es decir, relaciona la **sensibilidad** de nuestro modelo con los fallos optimistas (clasificar los negativos como positivos). Tiene sentido ya que, generalmente, si aumentamos el *recall*, nuestro modelo tenderá a **ser más optimista** e introducirá más falsos positivos en la clasificación.

Tabla 5-6 Evaluación de Conjunto de Prueba 10%: Curvas ROC y P-R [Elaboración propia]

Algoritmo Aplicado	Curva Característica Operativa Relativa Evaluación (Conjunto Prueba 10%)	Curva Precisión-Sensibilidad Evaluación (Conjunto Prueba 10%)
Randon Forest -RF		

Árbol Decisión - DT		
Árbol Extra - ET		

xgboost.Dmatrix xgboost.train	Imposible de realizar, AlgoritmoXGB de API Learning no implementa el método de predicción probabilística `No predict_proba(X)`	Imposible de realizar, AlgoritmoXGB de API Learning no implementa el método de predicción probabilística `No predict_proba(X)`
xgboost.XGBClassifier		

Será una opción interesante usar la curva ROC cuando tengamos un conjunto de datos más balanceado o queramos poner de manifiesto un indicador más relacionado con falsas alarmas (falsos positivos).

En las curvas ROC, nos interesa que la curva se acerque lo máximo posible a la **esquina superior izquierda** de la gráfica, de manera que el hecho de aumentar la **sensibilidad** (el *recall*) no haga que nuestro modelo introduzca **más falsos positivos**. Para nuestro caso, el clasificador con ET, presenta la gráfica de ROC de menor resultado con **0,92** para el ataque **DDoS LOIC-UDP**, mientras el mismo ataque, pero en el que le sigue, el DT con **0,94**. Ahora para ambos algoritmos el caso del ataque **DDoS UPD-Lag** tenemos un empate con **0,95.**

5.5. Evaluación de Pruebas DDoS 2019: Doble métricas de rendimiento

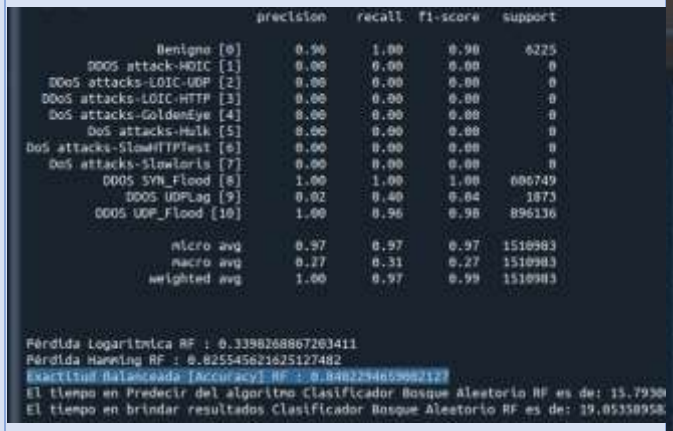
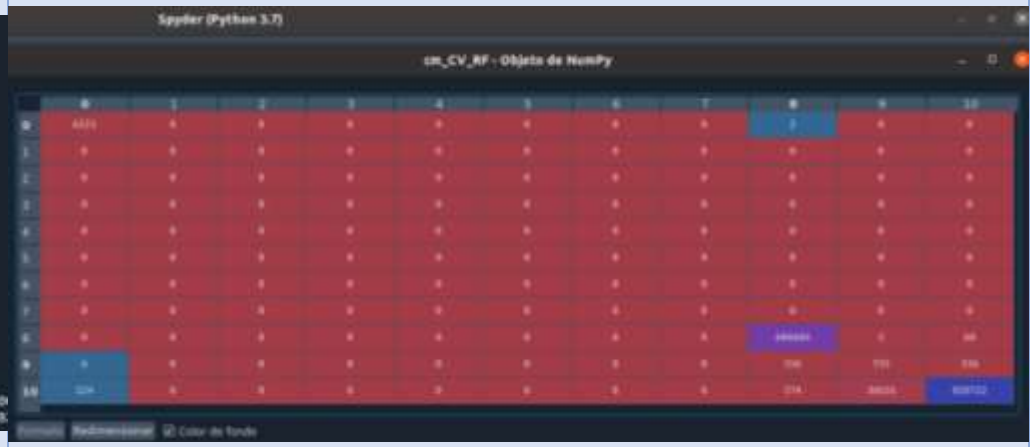
En nuestra segunda evaluación, tenemos un conjunto de prueba formado por CIC-DDoS2019, presenta estos resultados, un poco menores en porcentajes de efectividad respecto a la evaluación normal; pero precisamente esa era la intención, que fueran superior a un millón quinientos, donde solo 300 mil de ellos son flujos benignos, y el resto mayoritario ataques maliciosos.

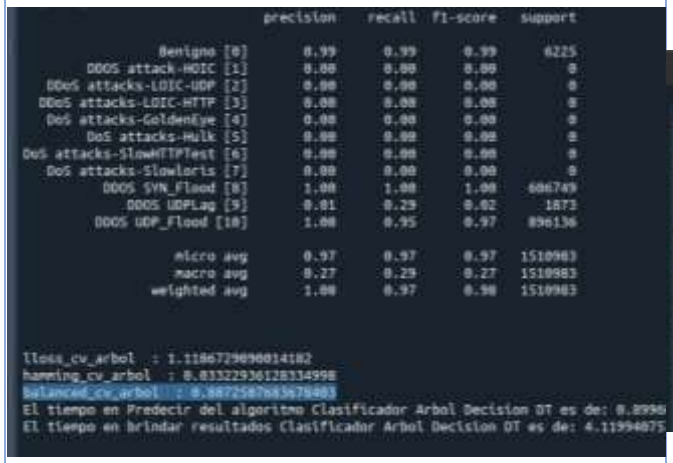
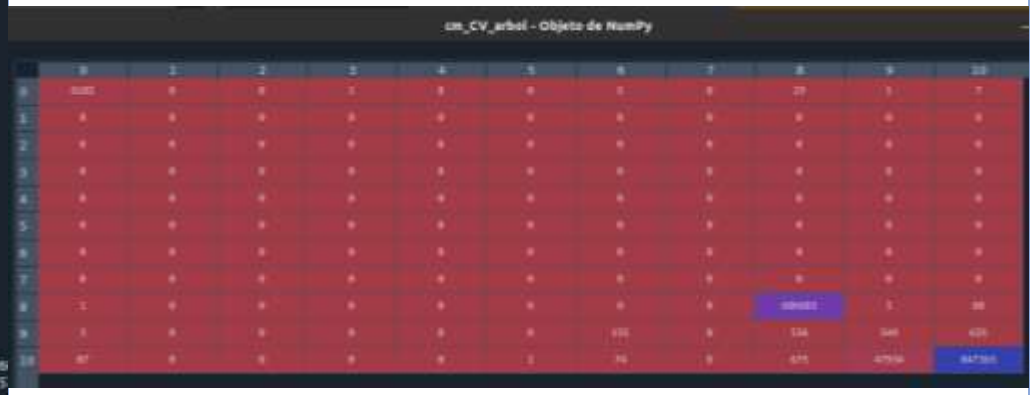
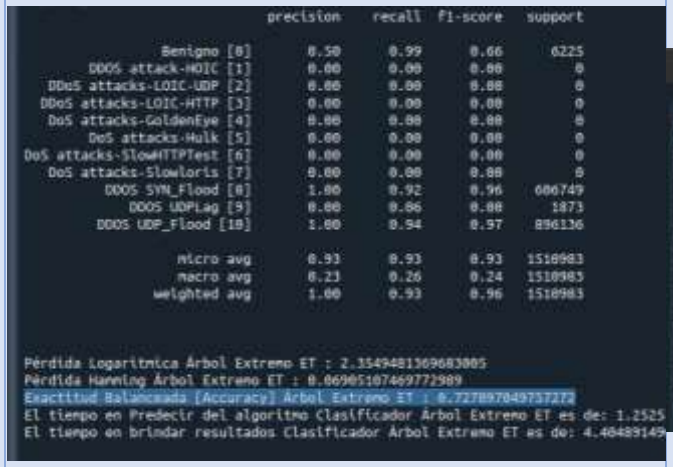
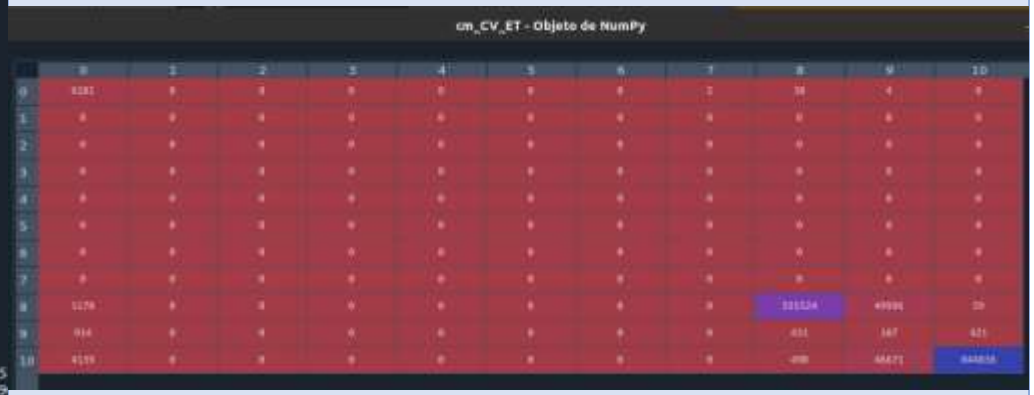
Lo primero que se ha de indicar, es que al no ser ni estratificado, y estar formado solo por el conjunto de CIC del año 2019, solo tienen muestras para 3 exclusivos ataques DDoS de tipo de explotación: únicamente ataques de **SYN-Flood**, **UDP-Flood** y **UDP-Lag**, junto a los **benignos**. Por esta misma situación, de comparar solo 4 tipos de clases, contra las 11 clases finales que tiene nuestros algoritmos clasificadores del modelo, al tener **diferentes dimensiones**, fue imposible obtener las gráficas de las Curvas de ROC y P-R, para el caso de nuestra segunda prueba. Por ello, solo se muestras los resultados de la Tabla 5-7, para la deducción de los **reportes de clasificación**, como sus **matrices de confusión**, con valores nulos, pues no existen en las muestras clases de ataques para siete (7) familias maliciosas denegación de servicio.

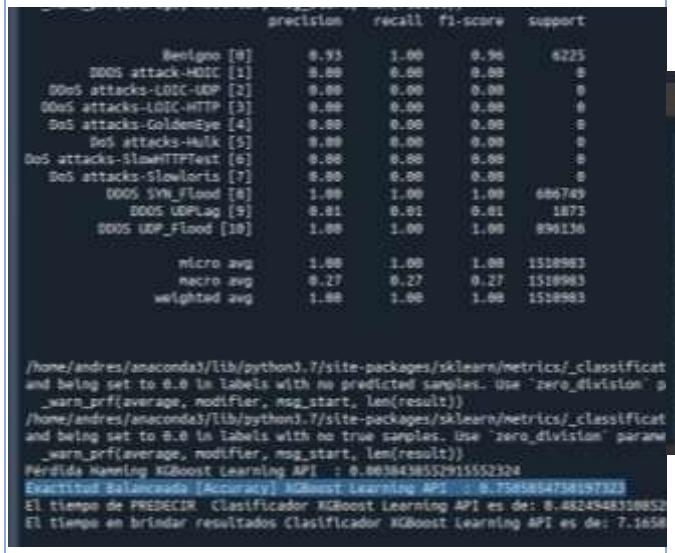
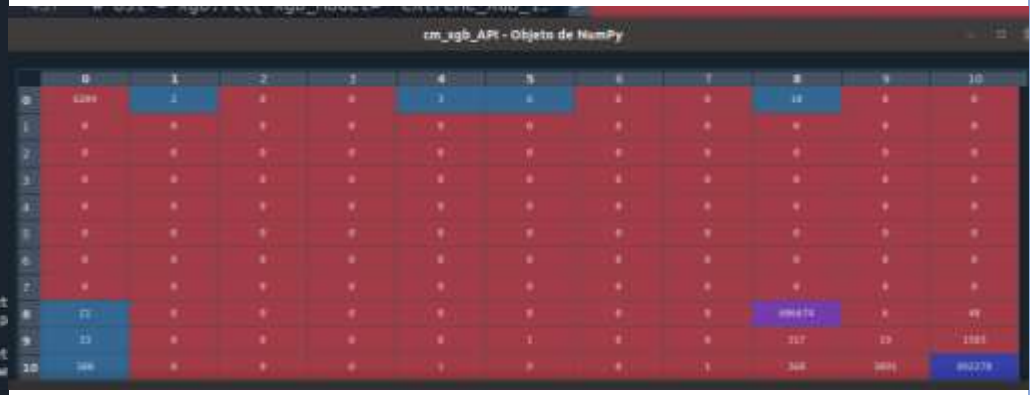
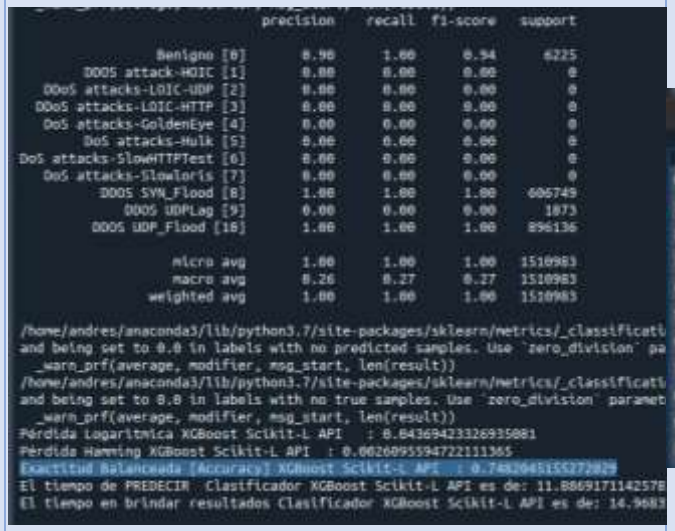
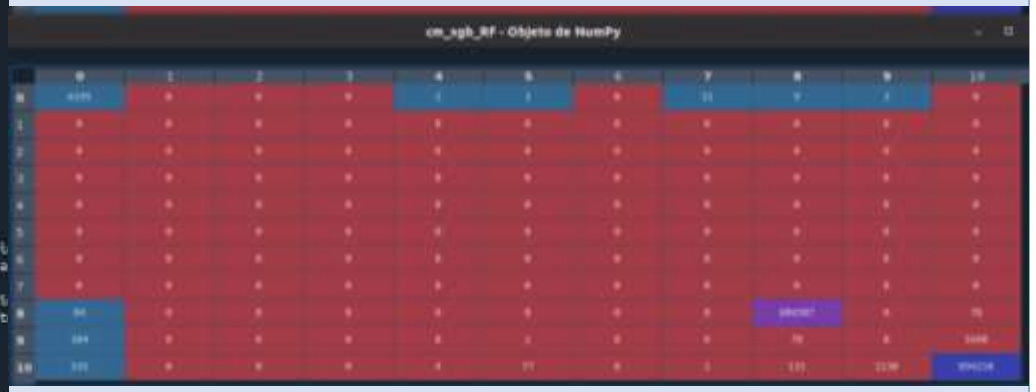
De igual forma a la realizada en la evaluación normal, se tendrá muy en cuenta la información ofrecida por las matrices de confusión, y a su vez, sin dejar atrás el reporte de calificación que en esta ocasión si presenta porcentajes no tan perfectos. Análisis que infiere, que la prueba en el mundo real el reto es muy grande, y no presenta casos ideales, ni utópicos.

La matriz de confusión para el oro del pódium, el clasificador RF presenta un total de FP=230 y FN=2, es decir 230 flujos clasificados mal como benignos, pues son **224 flujos UDP-Flood** y **6 de UDP-Lag**; así como 2 flujos clasificados erróneamente como SYN-Flood pero en realidad son **2 flujos benignos**. Ahora, con esta segunda evaluación si se ve más claro el error que presenta al clasificar erróneamente entre ataques de denegación distribuidos, se observa contiene una suma de errores totales

Tabla 5-7 Segunda Evaluación Conjunto no visto: Reportes y Matrices [Elaboración propia]

Algoritmo Aplicado	Reporte de Clasificación Segunda Evaluación (Conjunto no visto)	Matriz de Confusión Segunda Evaluación (Conjunto CIC-DDoS2019 no visto)
Randon Forest -RF		

Árbol Decisión - DT		
Árbol Extra - ET		

xgboost.Dmatrix xgboost.train		
xgboost.XGBClassifier		

de **38.387 flujos** clasificados de forma incorrecta entre mismos ataques, entre ellos, las etiquetas reales son: **65 flujos de SYN-Flood**, **1.112 flujos son UDP-Lag** y finalmente **37.190 son UDP-Lag**; para un soporte de muestras totales de **606.749**, **1.873** y **896.163** flujos respectivamente de esos 3 ataques.

Ahora, los otros algoritmos de clasificación pues presentan muchos más errores, como lo muestra los porcentajes en las métricas de los reportes de clasificación para el **DT** y **ET**, razón por la cual, no se tuvieron en cuenta en la prueba de implementación del mundo real.

Así mismo, aunque los reportes para el caso de los dos algoritmos de XGB, presentan porcentajes de 100%, de la unidad tanto en precisión [*precision*], exhaustividad [*recall*] y la puntuación F1 [*F1-score*] tanto del promedio Micro, como el promedio Ponderado. Los valores porcentuales del promedio macro, es tan bajo como entre el 26% y 28%. Pero si vemos ahora, sus matrices de confusión presentan mayores errores de falsos positivos y falsos negativos respecto a los flujos benignos, respecto al oro, los **Bosques Aleatorios**.

Cambiando de resultados, y ya explicados los de la Tabla 5-7, se prosigue a el análisis de otras métricas, como las perdidas tanto logarítmica como la de Hammig, mostradas respectivamente en las figuras Figura 5-39 y Figura 5-40. Se aprecia que los mejores resultados son obtenidos por los Gradientes Boosting Extremos XGB, pues entre más cercanos a cero, mejor rendimiento.

hamming_cv_arbol	float64	1	0.03322936128334998
hamming_cv_Ensemble	float64	1	0.08207173740538444
hamming_cv_ET	float64	1	0.06905107469772989
hamming_cv_RF	float64	1	0.025545621625127482
hamming_XGBoost_API	float64	1	0.0038438552915552324
hamming_XGBRFC	float64	1	0.0026095594722111365

Figura 5-39 Errores Hamming Segundo Conjunto de Prueba [Elaboración propia]

lloss_cv_arbol	float64	1	1.1186729090014182
lloss_cv_Ensemble	float64	1	0.474077697821595
lloss_cv_ET	float64	1	2.3549481369683005
lloss_cv_RF	float64	1	0.3398268867203411
lloss_XGBRFC	float64	1	0.04369423326935081

Figura 5-40 Errores Logarítmicos Segundo Conjunto Prueba [Elaboración propia]

Nombre	Tipo	Tamaño	
balanced_cv_arbol	float64	1	0.8072507683678403
balanced_cv_Ensemble	float64	1	0.7160087859385613
balanced_cv_ET	float64	1	0.727097049757272
balanced_cv_RF	float64	1	0.8402294659082127
balanced_XGBoost_API	float64	1	0.7505854750197323
balanced_XGBRFC	float64	1	0.7482045155272029

Figura 5-41 Puntuación promedio balanceada Segundo Conjunto Prueba [Elaboración propia]

Y con la Figura 5-41 que plasma los resultados en esta ocasión para una puntuación promedio balanceada [*balanced accuracy score*], y reafirma nuestra decisión de lección como oro de nuestro pódium para los **Bosques Aleatorios** con un grandioso **84,02%** para una difícil prueba con un conjunto de prueba muy desigual, y de mayoría de ataques maliciosos de denegación de servicios.

Bueno en cuestión de métricas para saber el desempeño y rendimiento, radica la vital importancia de las etiquetas reales, con las cuales se validan respecto a las etiquetas predichas de clasificación. Todos estos resultados, son gracias a la funcionalidad del aprendizaje supervisado y su valor en el etiquetado de los datos. Veamos ahora la salida de nuestros algoritmos, veamos en sí las respuestas que se nos brinda son de dos formas, según el método aplicado de ***predict*** o ***predict_proba***, como **vector columna** de su decisión de clasificación, como una **matriz de predicciones probabilísticas** de clasificación, que se pueden observar en las figuras Figura 5-44 y Figura 5-43.

Estas salidas, y respuestas que determina el uso de las técnicas de aprendizaje automático se evidencian en la figura, y son los resultados que encontramos en nuestras dos evaluaciones, pero para los casos que no se tenga la etiqueta real, dichas métricas son funcionalmente imposible de hallar.

Así la interpretación de resultados de dicha figura, radica en que se ha obtenido una clasificación automática, realizada por la máquina de Turing (computadora) gracias a un entrenamiento previo, y en cuya decisión se fundamenta en operaciones de árboles de

decisión para nuestro caso aplicado, y resulta en unos pesos probabilísticos, para determinar a qué tipo de clase o familia de ataques DoS o DDoS o trafico benigno. Estos son los resultados estadísticos, que denominamos con el tutor del proyecto, visto en la sección 3.11.

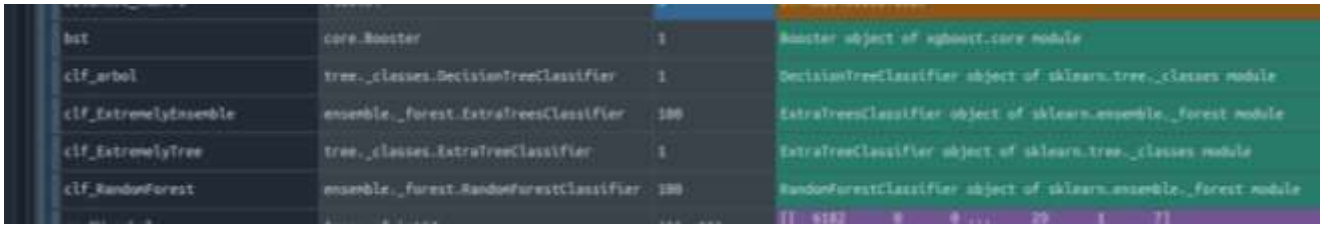


Figura 5-42 Cinco Clasificadores mostrados de nuestro Modelo DetD&DoS [Elaboración propia]

Para finalizar la etapa de evaluaciones de las pruebas, y pasar a la implementación donde el modelo tendrá la decisión de la clasificación y a la detección de nuestro caso ataques maliciosos DoS y DDoS. Se muestran en Figura 5-42 la gráfica de los cinco algoritmos de clasificación del pódium, falta solo el XGBRFC, pero por ser llamado por X, quedo de ultimo en la lista, y no se alcanza a apreciar.

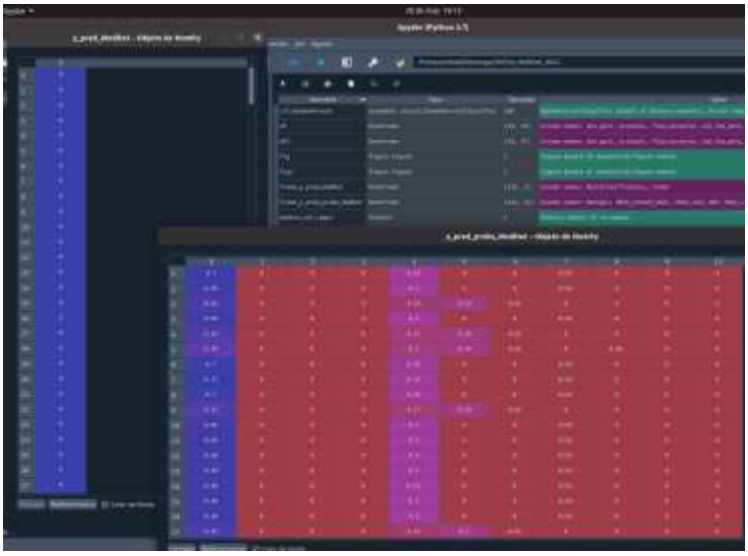


Figura 5-44 Resultados del modelo: vector Columna y Matriz de probabilidades [Elaboración propia]

Figura 5-43 Resultados del modelo Matriz de probabilidades [Elaboración propia]

5.6. Implementación y validación en mundo real en red hogar.

Las capturas de paquetes de trafico de red en tiempo real, en vivo, se ha implementado, se ha esnifado o capturado a través de mi tarjeta de red por la interfaz del puerto ethernet, en mi caso llamado **eno1**, de mi red doméstica, conectado alambricamente a un módem de la ISP (Proveedor de Servicio), en mi caso hacia TVNorte de fibra óptica de red GIPON. Ver Figura 5-47 realizada en packet, de la configuración o topología de la red doméstica mencionada.

5.6.1. Etapa Previa: Capturar Tráfico de red

Primero se ejecuta la herramienta CICFlowMeter en la terminal, depende de la versión cambia el comando, y como se ha explicado en la sección 4.8.1, existen dos formas de obtener el archivo deseado de conjunto de datos (csv), sin importar si el paquete capturado (pcap) ha sido esnifado o ya se tenía almacenado previamente. Como se ha mencionado, los archivos deben quitarse los permisos de seguridad de *root* como se puede ver en la Figura 5-46.

Después de obtener el archivo de comas separadas por variables en crudo, el proceso es el mismo que hemos planteado en nuestra metodología, se prosigue con la etapa de pre-procesamiento, y continua normalmente. Se puede ver en la Figura 5-45, un problema de configuración en el SO Ubuntu 20.04 distribución focal.

5.6.2. Usar el modelo definitivo para tomar una decisión final

Después de todo el análisis desarrollado en las etapas de entrenamiento, y de la doble evaluación que se ha aplicado, tenemos que el mejor sin lugar a dudas de los 5 algoritmos clasificadores, es el de Bosques Aleatorios, para la combinación especifica de nuestro conjunto de datos, con esto se procede a re-entrenar dicho RF, pero esta vez, con todos los flujos etiquetas que se disponen: los más 12 millones, más los 2 conjuntos de prueba, para sumar más de 15 millones de flujos, y se ha procedido a ser usado en nuestra implementación como se observan los resultados estadísticos obtenidos de los paquetes capturados principalmente el disponible por el mismo CIC, pero del año 2016, que cuenta con dicho archivo pcap mostrado en la sección siguiente 5.6.2.1 de más de 300 mil flujos de paquetes, después se analizaran paquetes capturados pcap de presuntos ataques de denegación, encontrados en la red, como muestras para validar nuestro modelo DetD&DoS propuesto.

Figura 5-45 Configurando el controlador de la tarjeta de red [Elaboración propia]

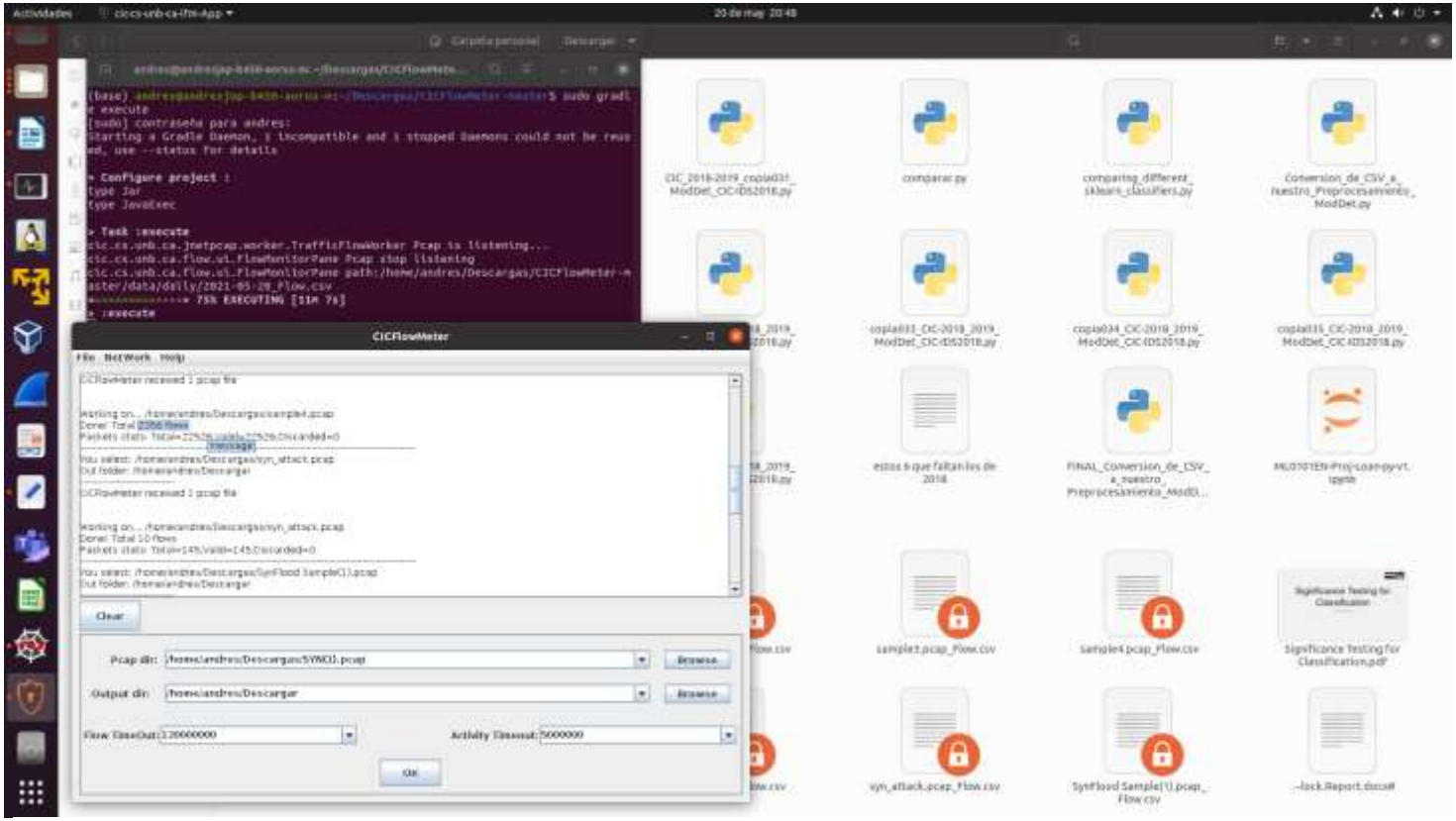


Figura 5-46 Herramienta CICFlowMeter y archivos CSV con permisos de *root* [Elaboración propia]

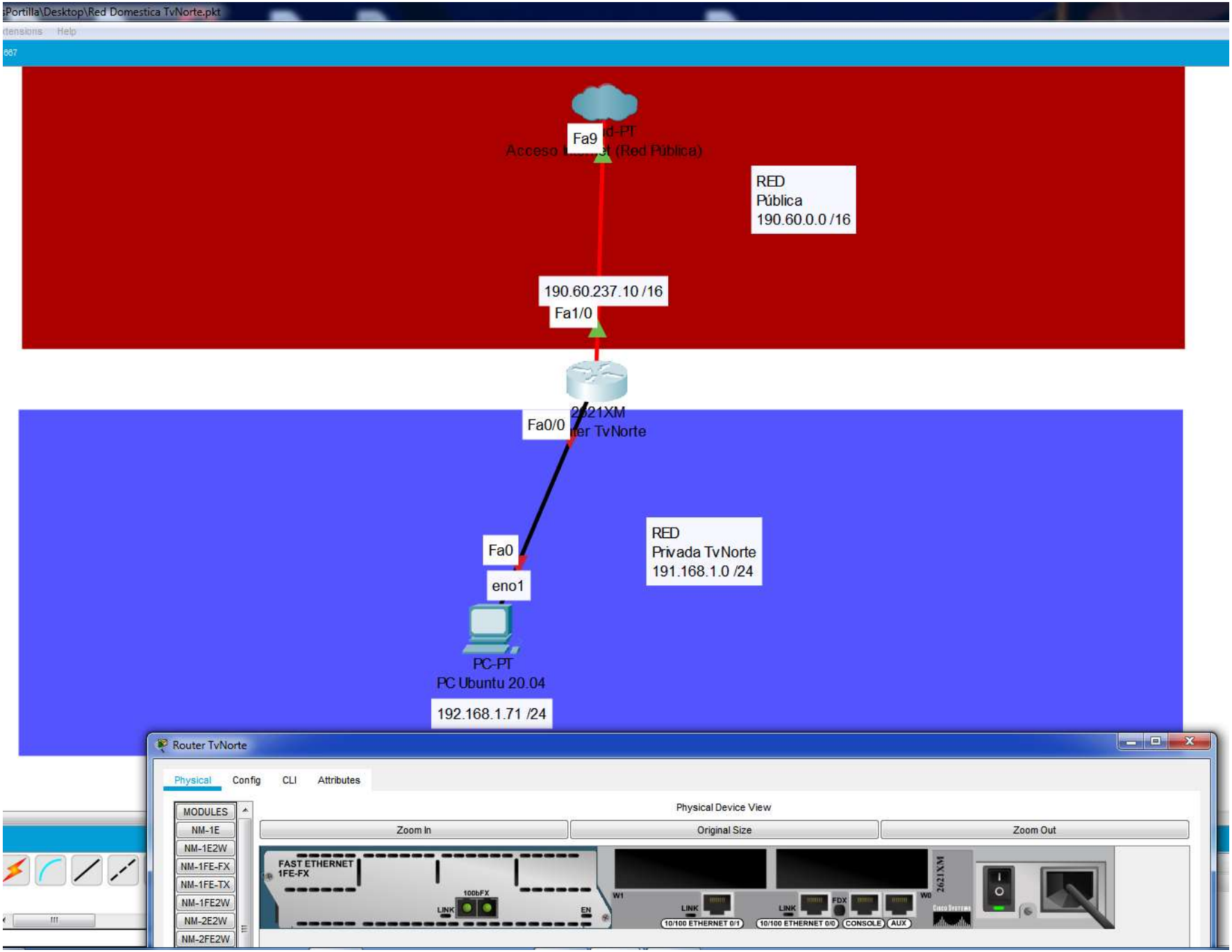


Figura 5-47 Topología de la Red Domestica implementada

En las próximas seis subsecciones se presenta los resultados de la clasificación deducida por nuestro modelo de AA denominado DetD&DoS, para seis (6) archivos de paquetes capturados, donde el primero si se tiene la certeza de contener ataques de denegación de servicio a nivel de protocolos de capa de aplicación y que contiene la estupenda cifra de casi 300 mil flujos de tráfico de red, y la última que fue esnifado, y tengo la cereza que es trafico benigno y clasificado como tal.

Así de los 6 pcap de ataques, quedan cuatro (4) archivos donde se presume son ataques, y efectivamente el modelo detecta, y clásico en ciertos ataques DoS y DDoS.

Justamente se va explicar el primer archivo que es más grande y significativo, y cuyos análisis de los otros cinco (5) ataques prosigue de la misma forma. Así es la razón por la cual se ha creado esas tablas que resumen, y sirven para comparar y contrastar uno con otro.

Iniciamos con la Figura 5-48, que representa un gráfico de barras y se aprecia mucho más de 250 mil flujos catalogados como benignos, y cuyo detecta Ataques DoS y DDoS estipulados en 6 posibles tipos de ataques de denegación a saber, y en orden descendente: **HOIC**, **UDO-Lag**, **GoldenEye**, **Slowloris**, **Hulk** y **UDP-Flood**. En la Figura 5-49, se crea un gráfico de torta, para representar la porción y el porcentaje respecto a los flujos totales. Por ultimo para el análisis del vector columna de solución, se realiza una tercera grafica que recorre las más de 300.000 flujos, para ver donde se ubica los flujos maliciosos, observar la gráfica de Figura 5-50.

Ahora bien, para las gráficas de los resultados de la matriz de probabilidades de clasificación, tenemos las dos figuras de línea, ambas figuras iguales, ver figura y figura, pero en diferente tamaño y zoom. Después continúan doce (12) figuras de dispersión, pero solo se muestran las más representativas y que muestren información relevante. Para el caso se muestra la Figura 5-51 y la Figura 5-53 para los ataques maliciosos DoS y DDoS.

El resto de subsecciones siguientes de 5.6.2.2 hasta 5.6.2.6 aplica la misma lógica de interpretación, obviamente con otros resultados obtenidos, mostrados en las siguientes tablas Tabla 5-8, Tabla 5-9, Tabla 5-10, Tabla 5-11 y Tabla 5-12.

5.6.2.1. Resultados Gráficos para conjunto de datos CIC-DoS2016.

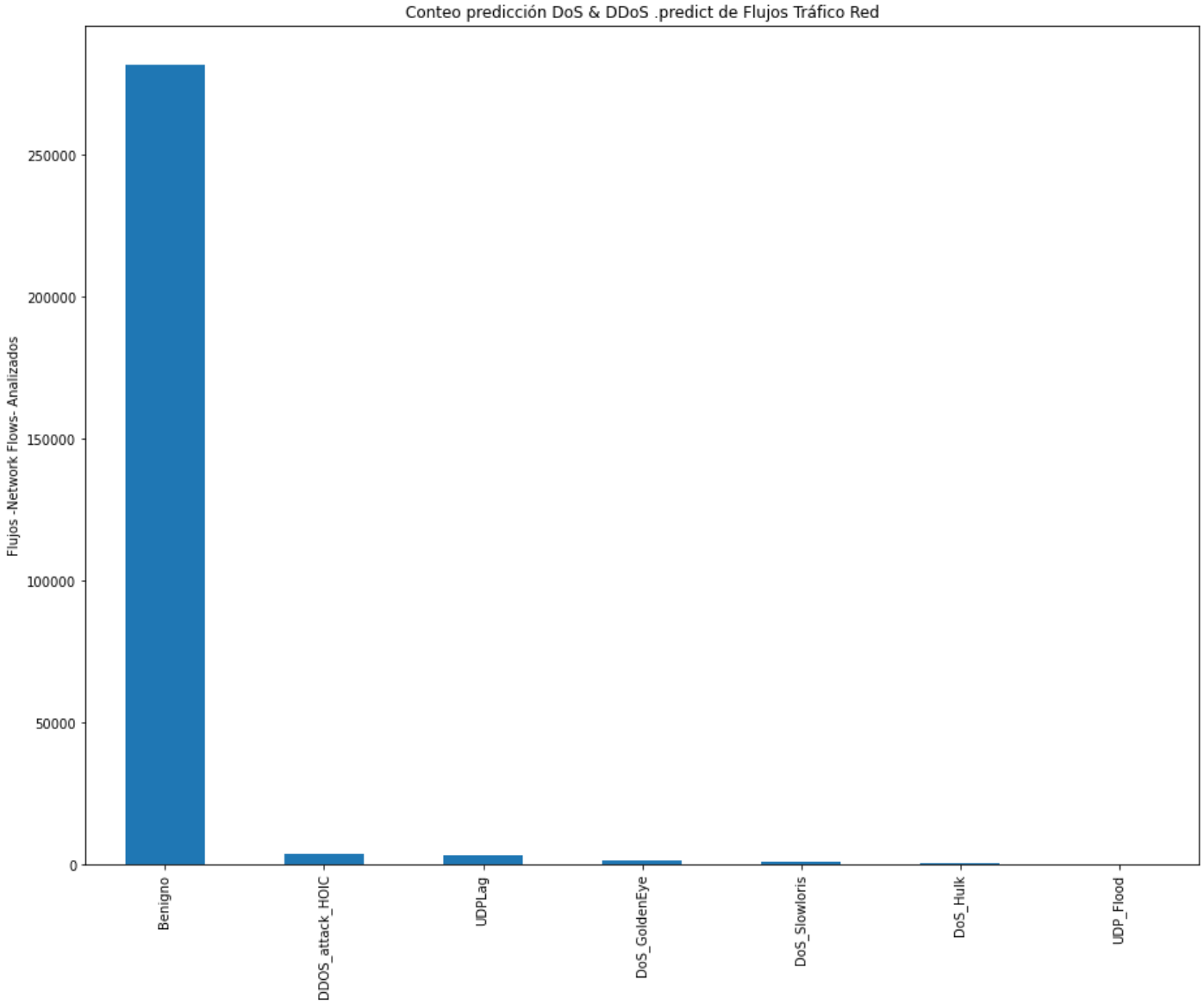


Figura 5-48 Gráfica de Barras para Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

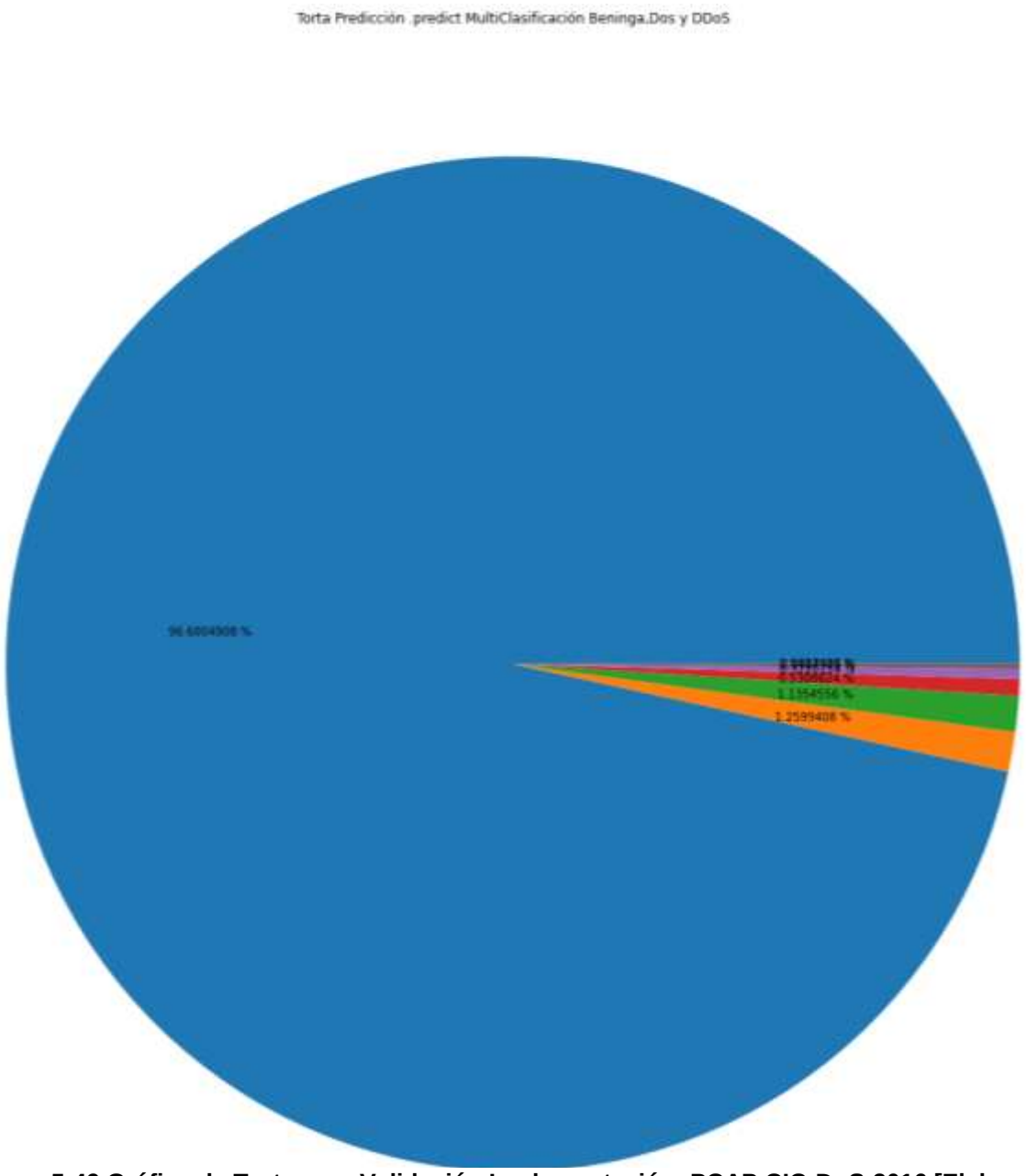


Figura 5-49 Gráfica de Torta para Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

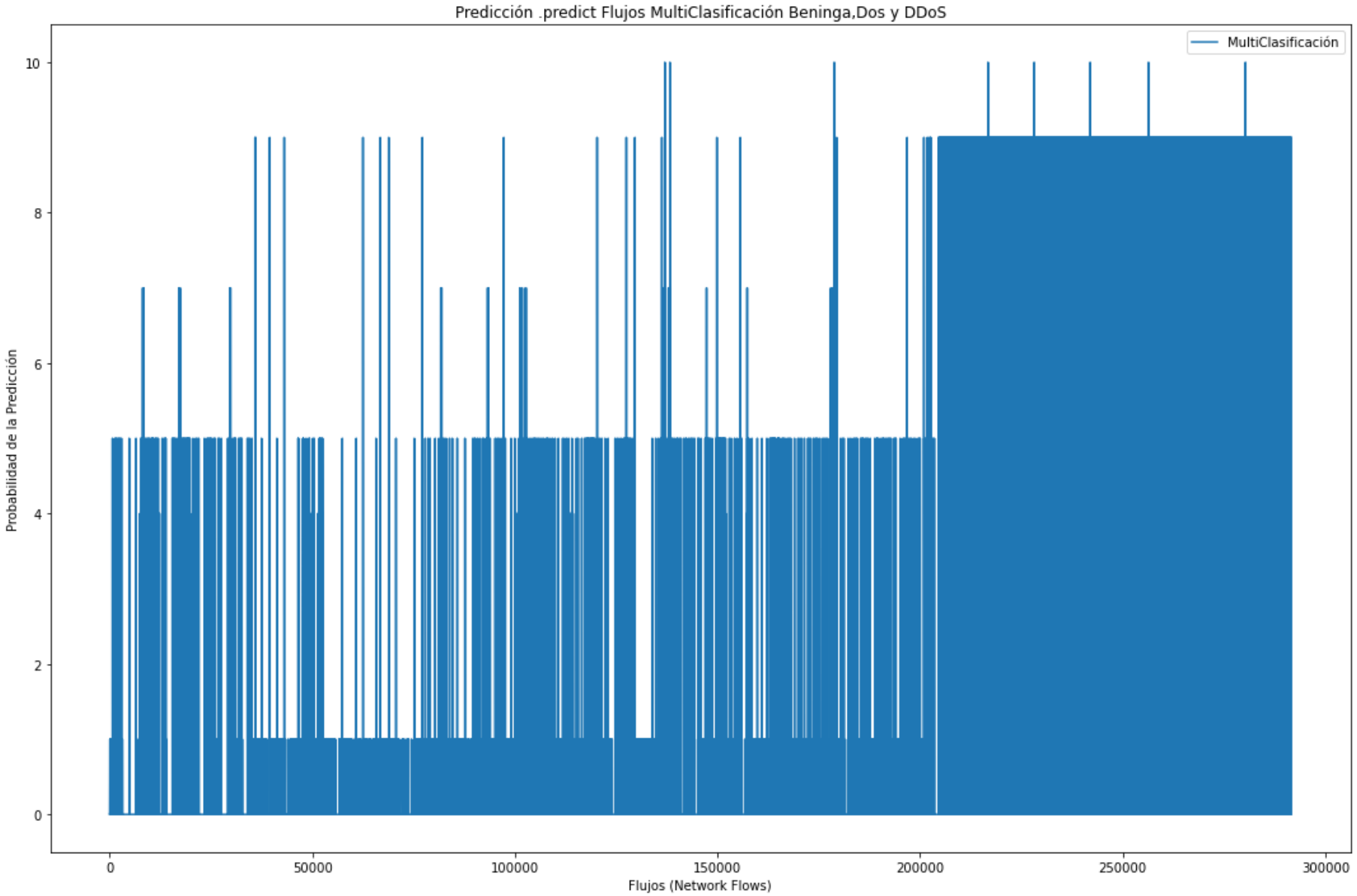


Figura 5-50 Gráfica de Líneas para Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

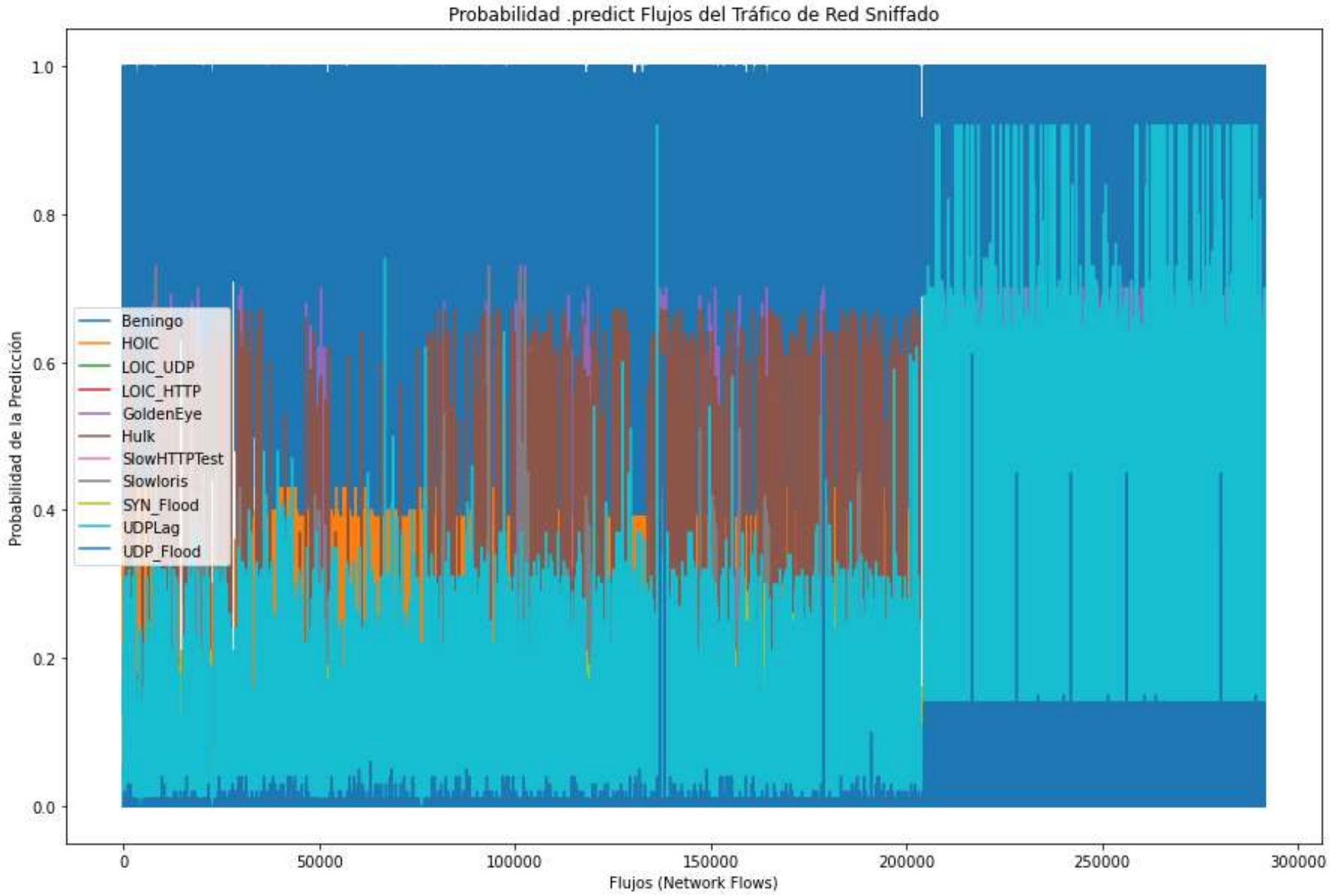


Figura 5-51 Gráfica de Líneas con Matriz Probabilidades Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

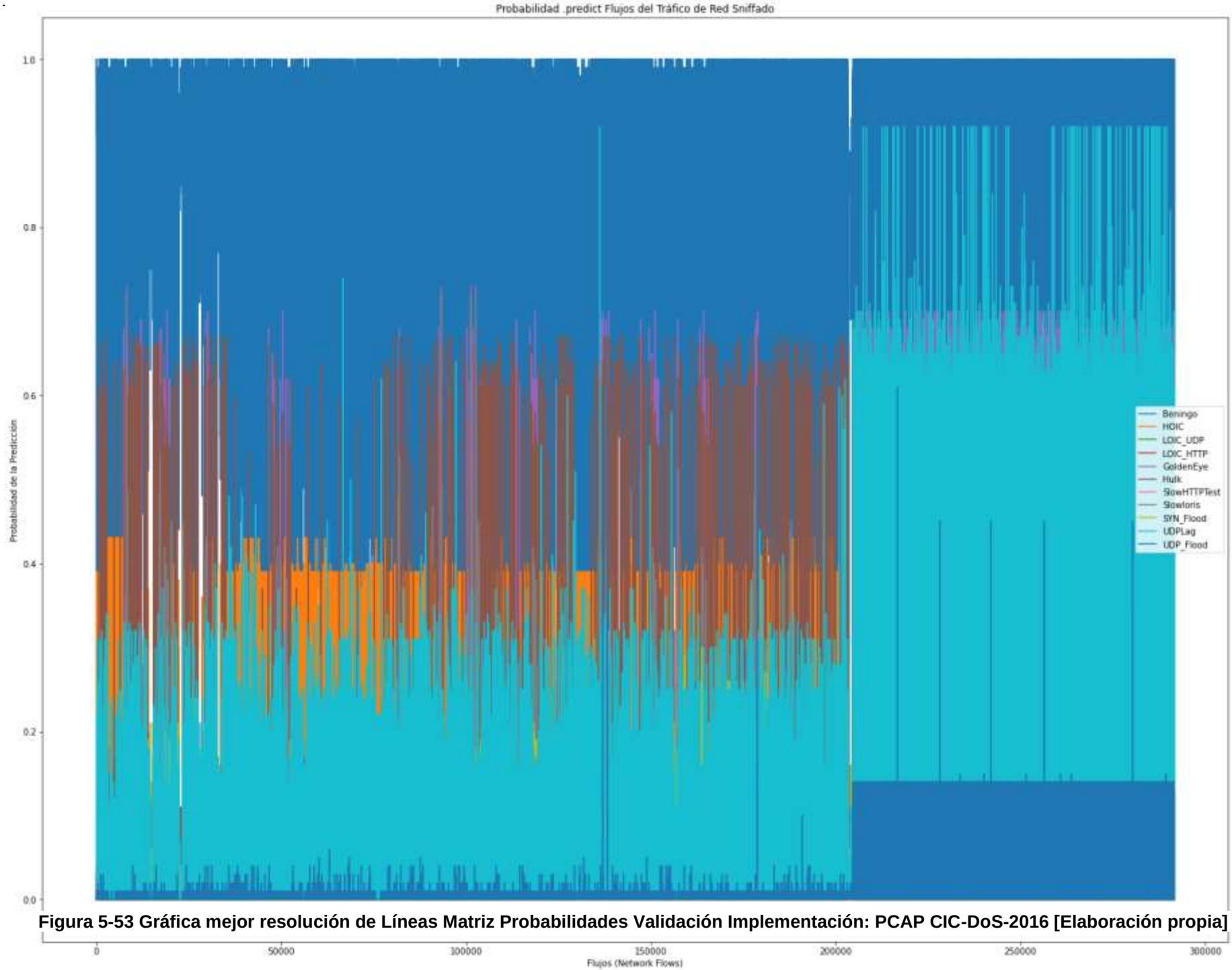


Figura 5-53 Gráfica mejor resolución de Líneas Matriz Probabilidades Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

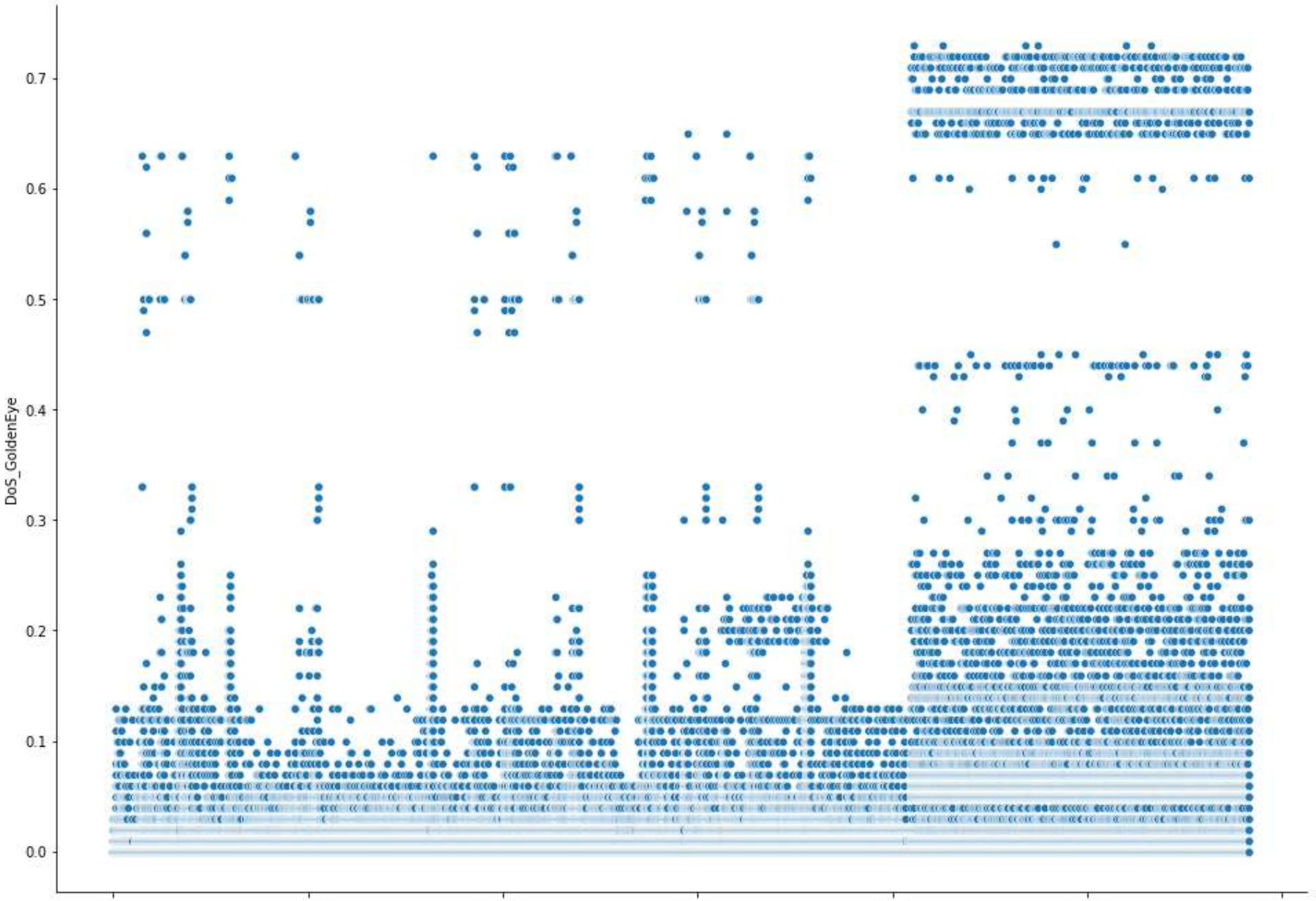


Figura 5-54 Gráfica de Dispersión Ataque GoldenEye Matriz Probabilidades Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

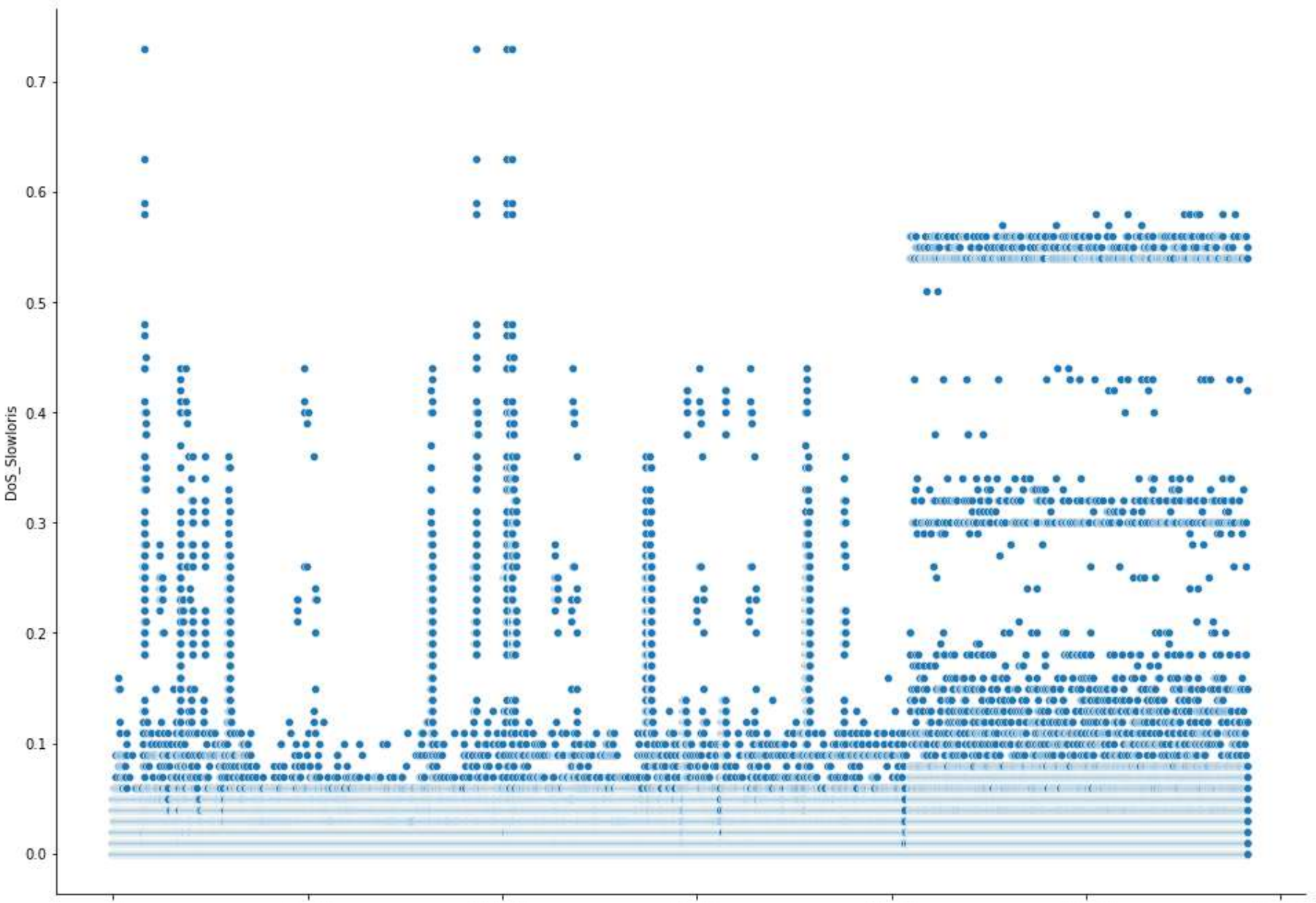


Figura 5-55 Gráfica de Dispersión Ataque Slowloris Matriz Probabilidades Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

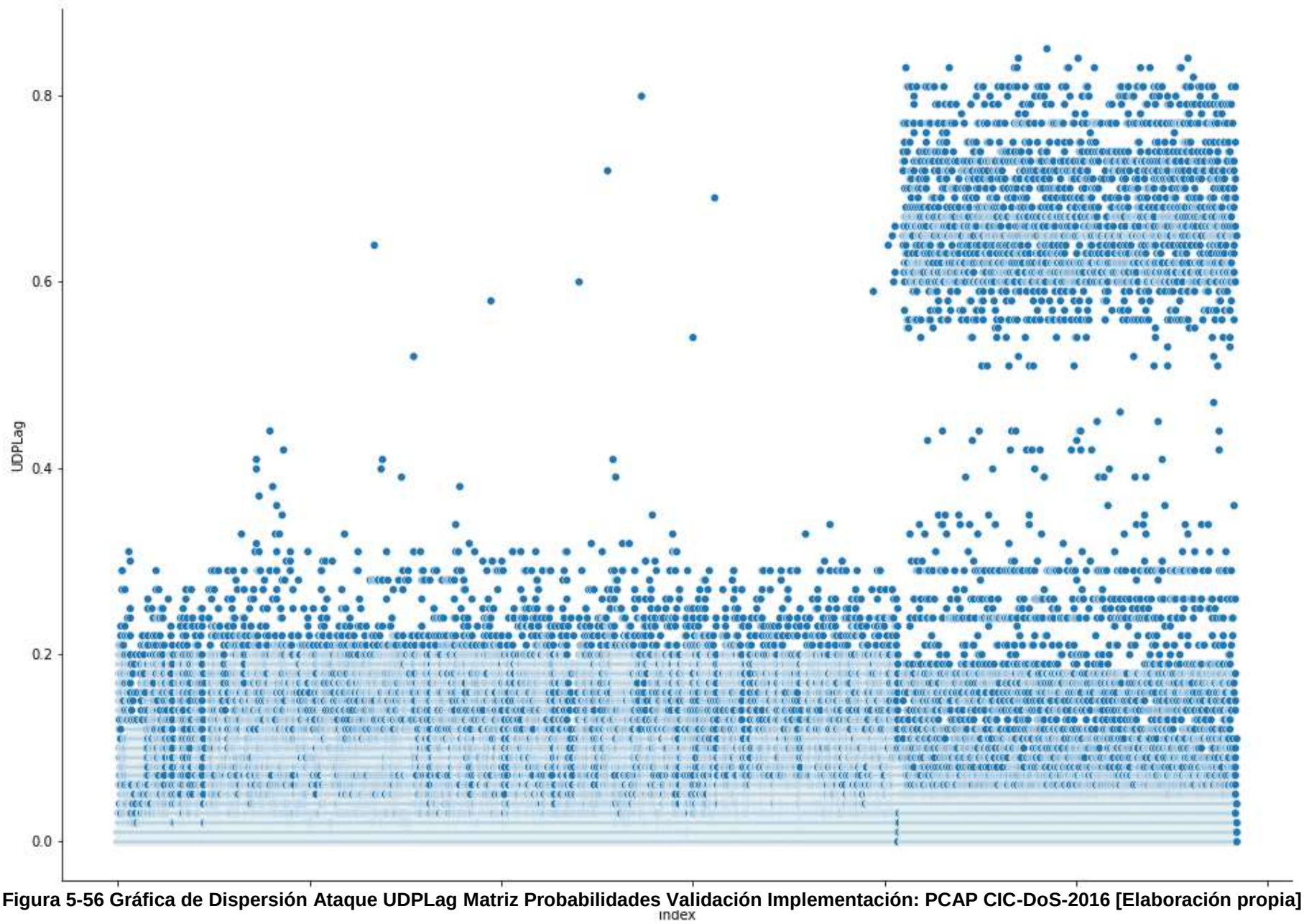
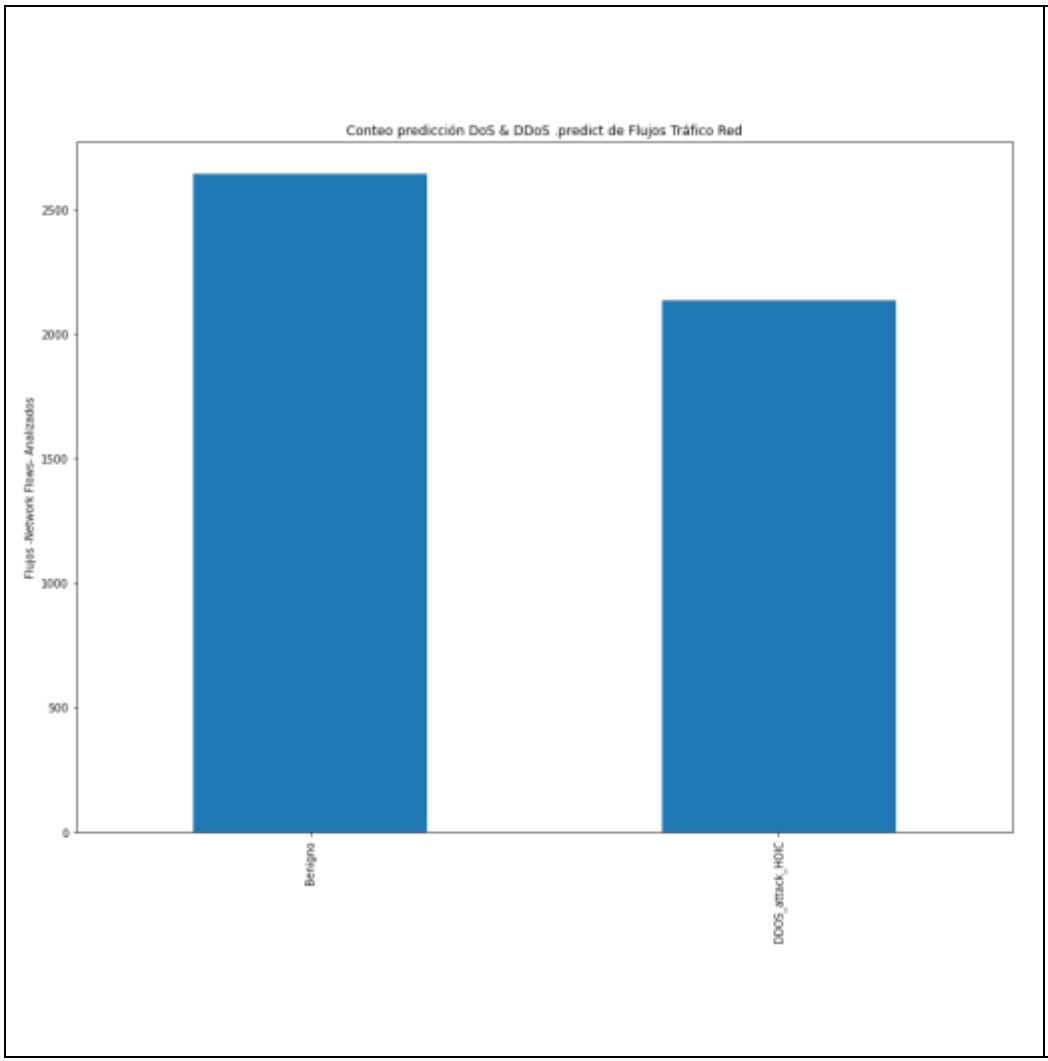
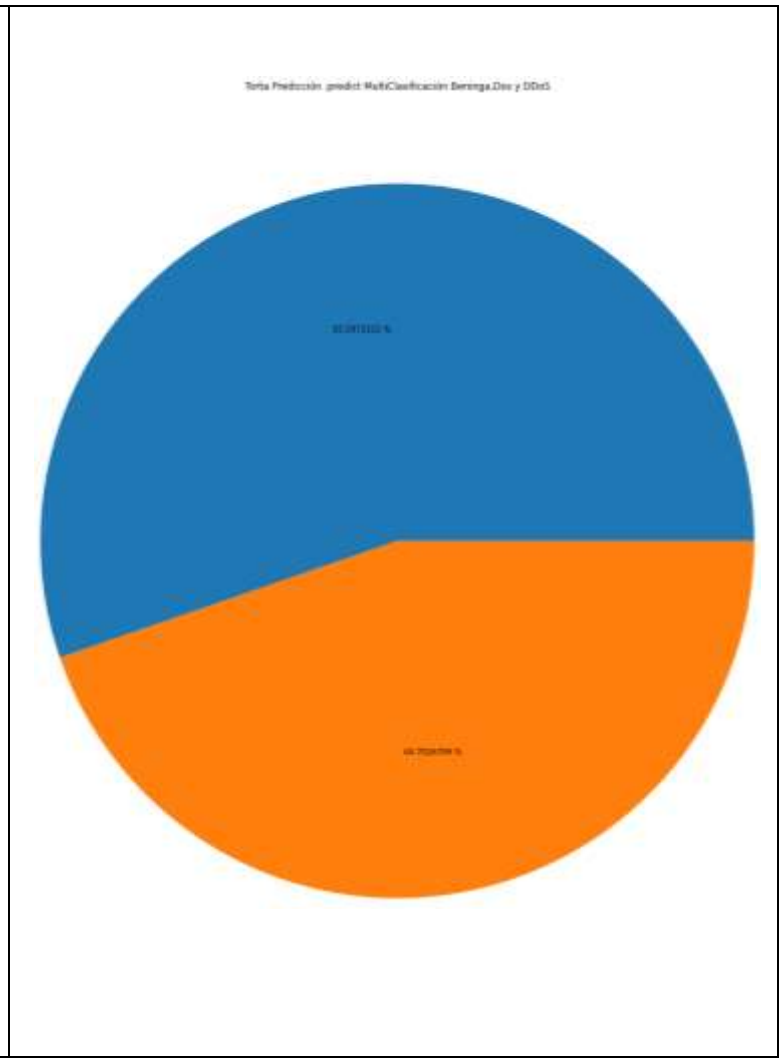
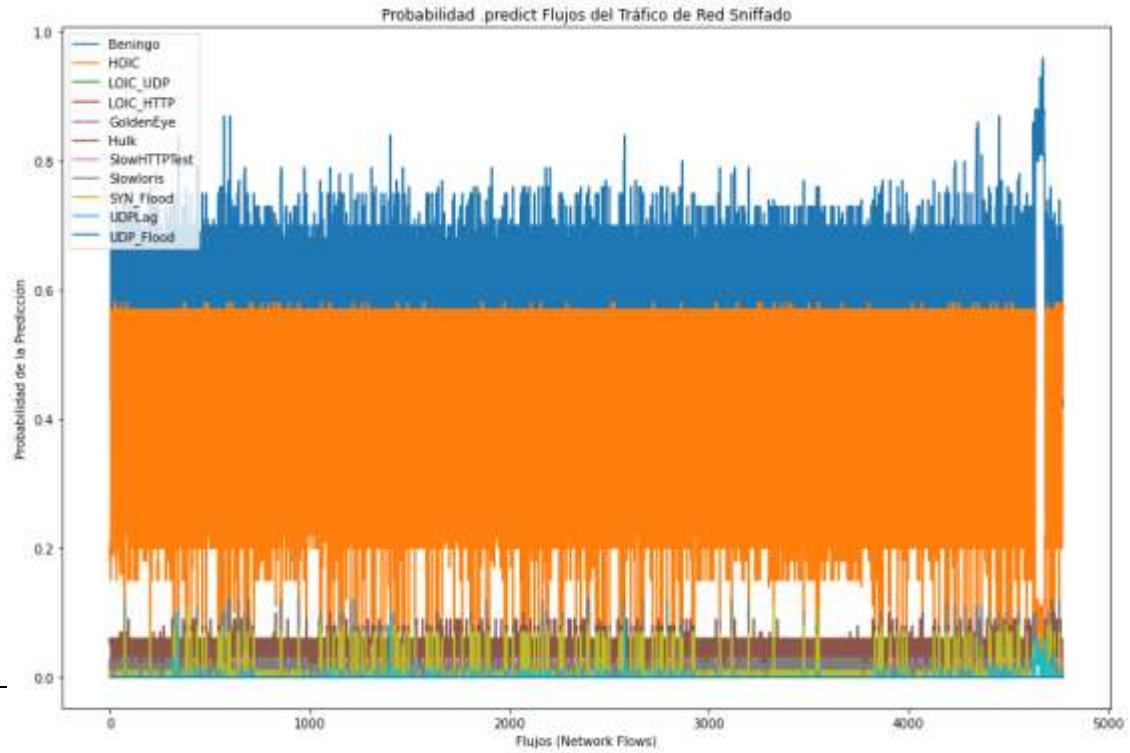


Figura 5-56 Gráfica de Dispersión Ataque UDPLag Matriz Probabilidades Validación Implementación: PCAP CIC-DoS-2016 [Elaboración propia]

5.6.2.2. Resultados Gráficos para presunto ataque HOIC







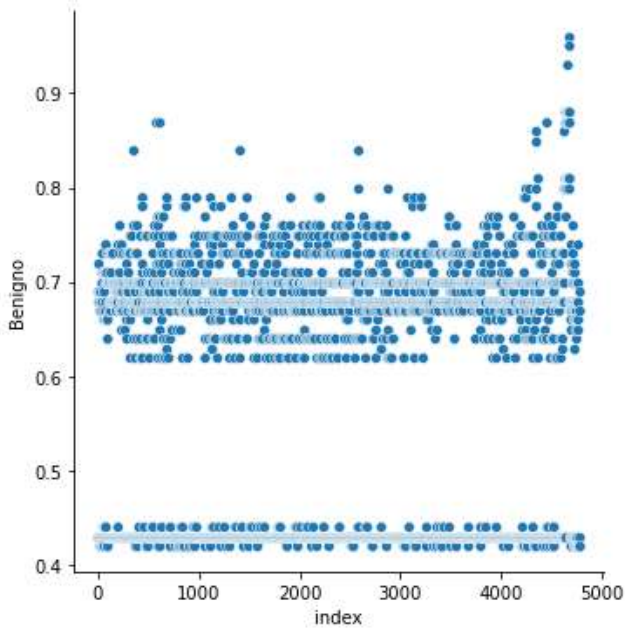
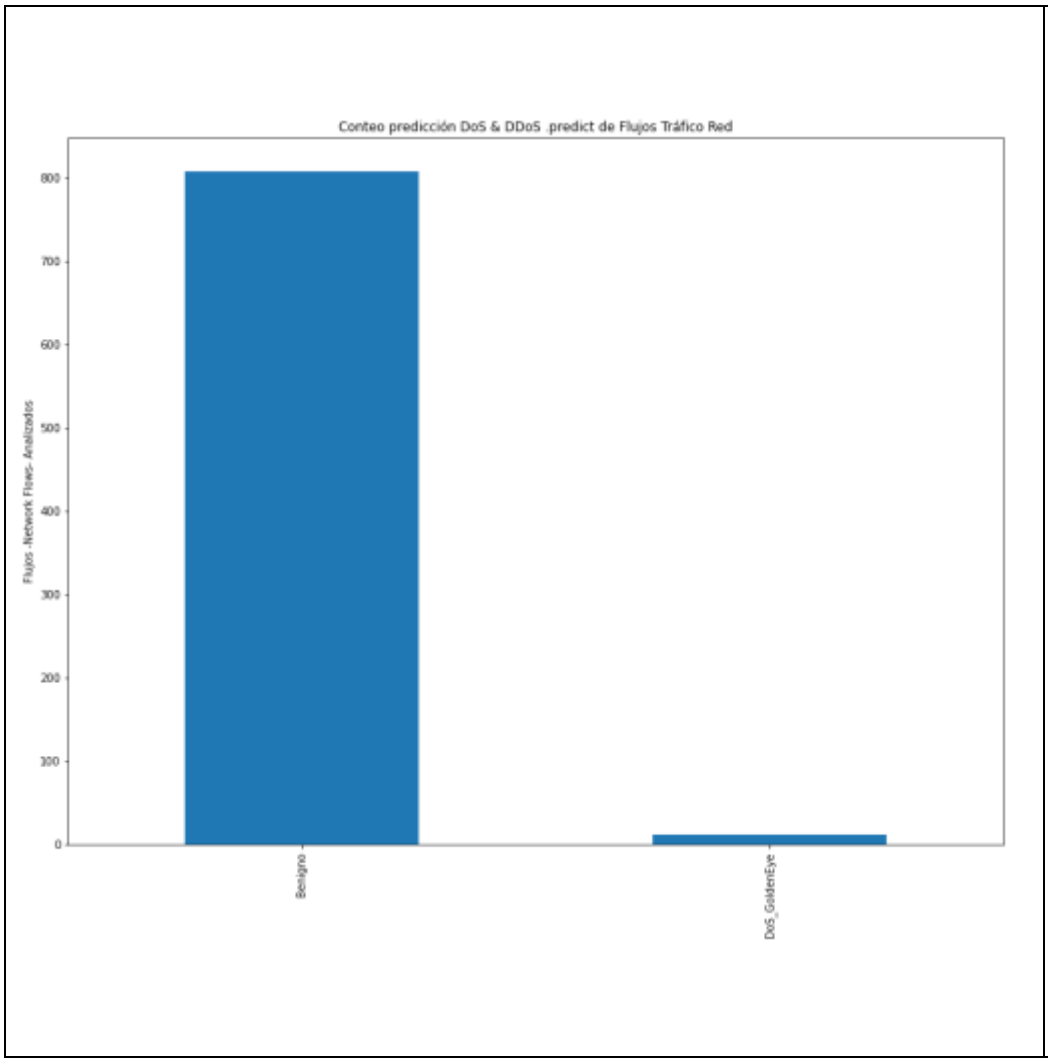
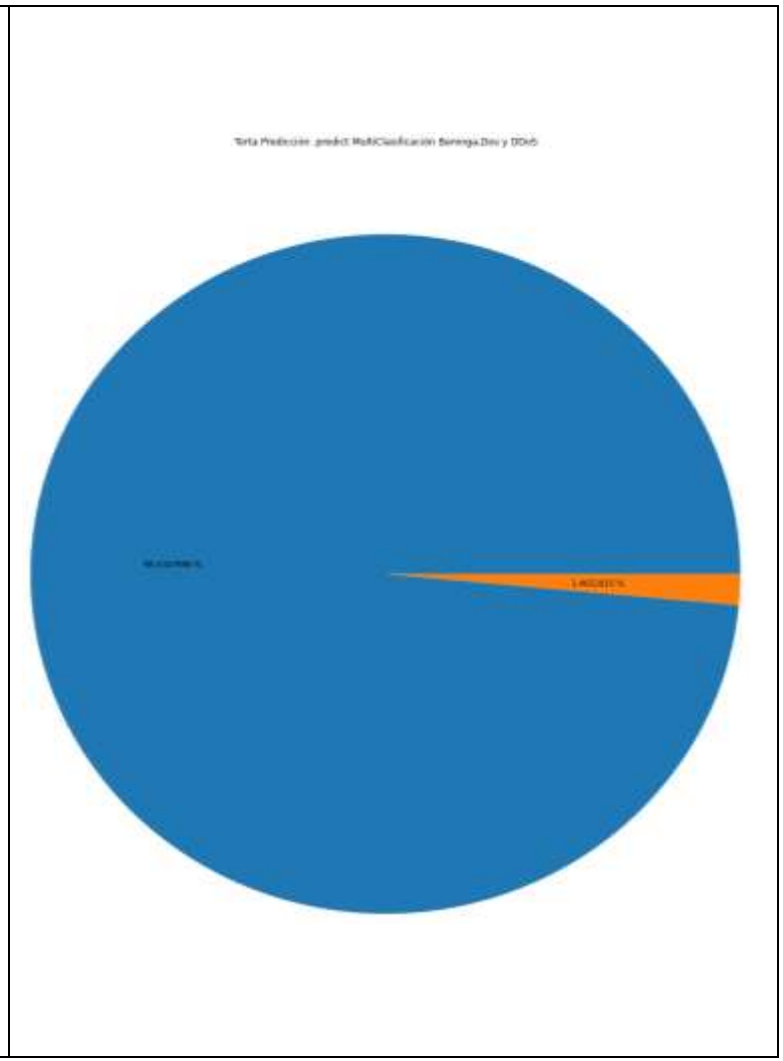


Tabla 5-8 Resultados Gráficos para presunto ataque HOIC [Elaboración propia]

5.6.2.3. Resultados Gráficos para presunto ataque GOLDENEYE





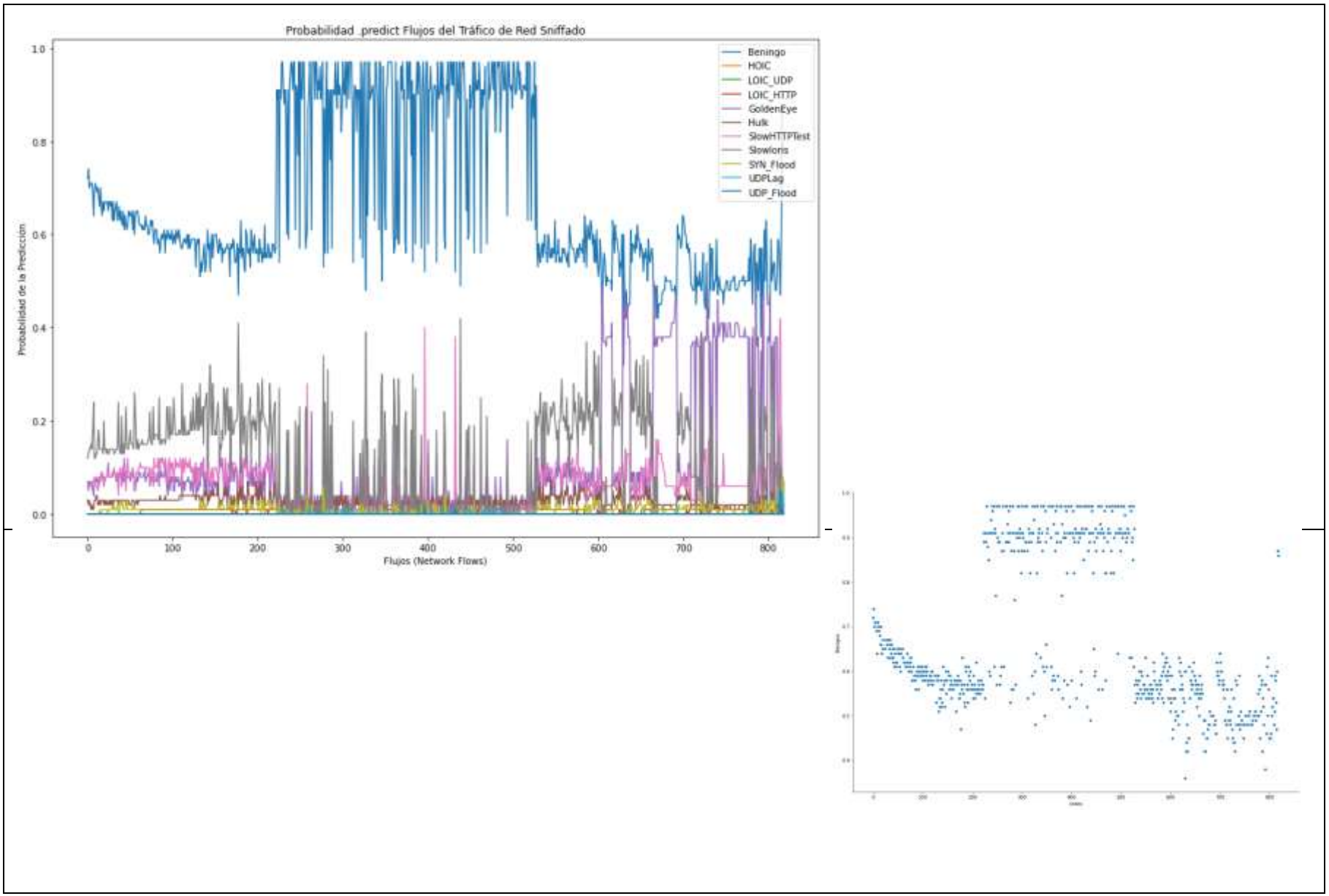
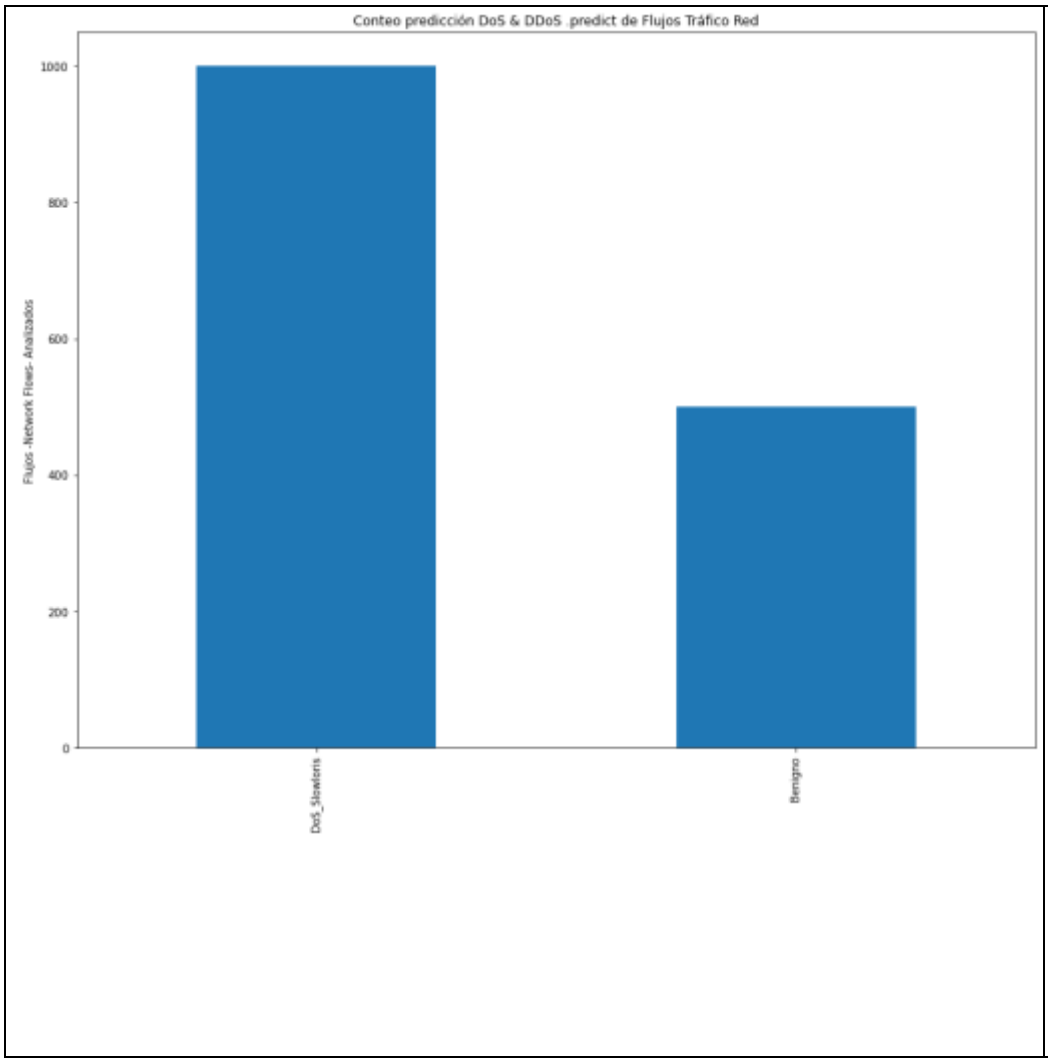
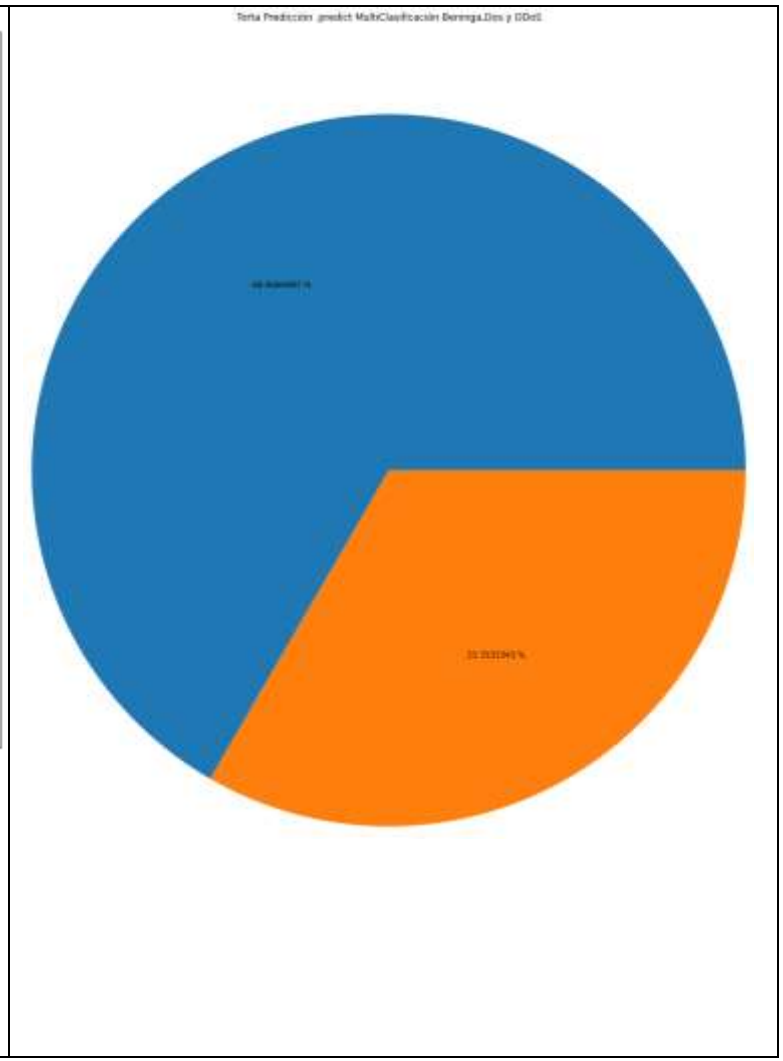
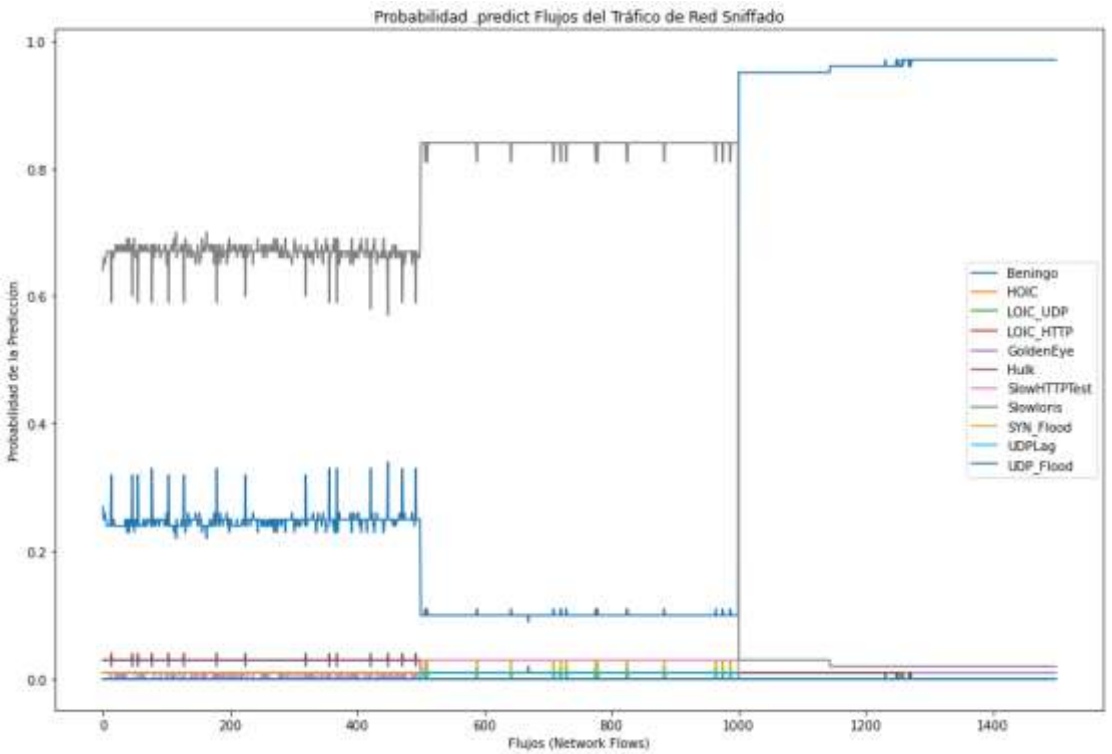


Tabla 5-9 Resultados Gráficos para presunto ataque GOLDENEYE [Elaboración propia]

5.6.2.4. Resultados Gráficos para presunto ataque SLOWLORIS







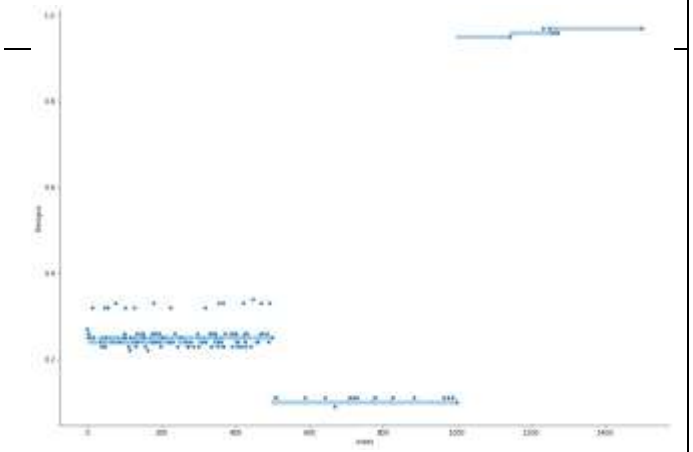
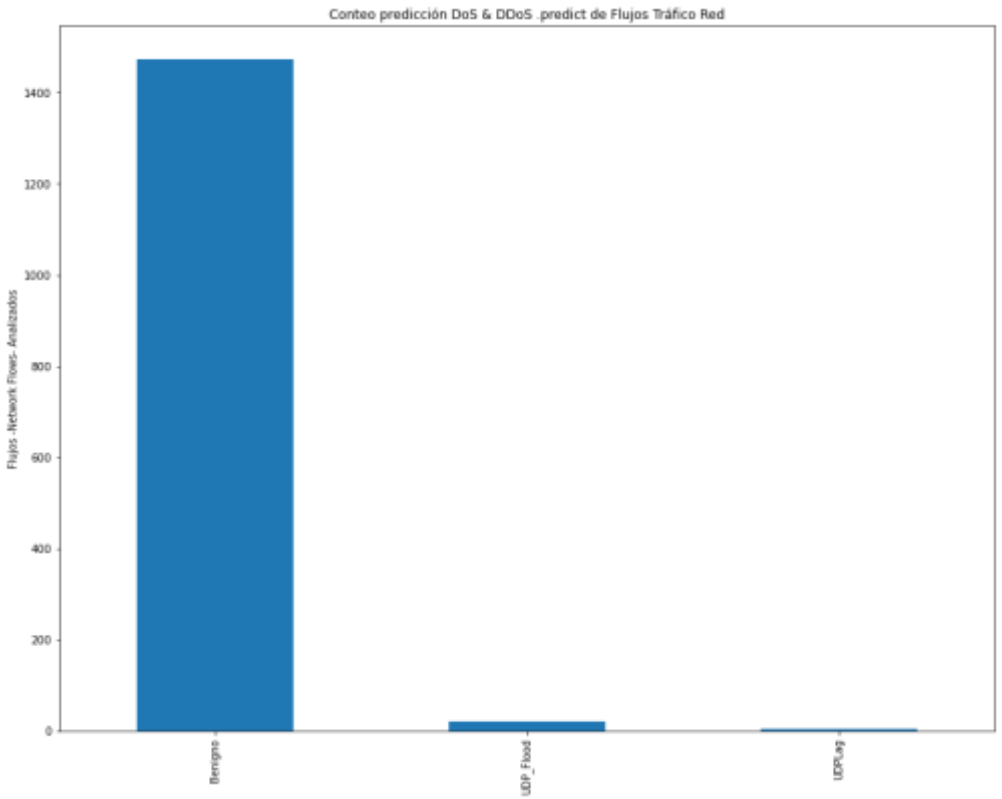
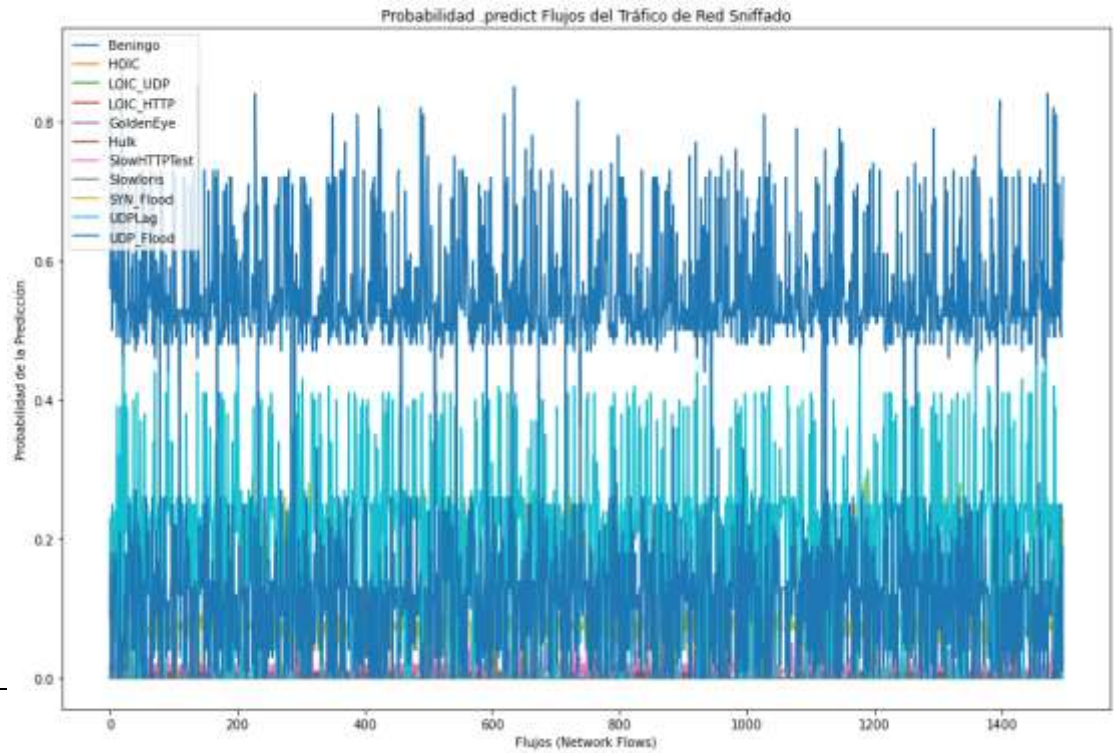


Tabla 5-10 Resultados Gráficos para presunto ataque SLOWLORIS [Elaboración propia]

5.6.2.5. Resultados Gráficos para presunto ataque DATASET DoS RARO





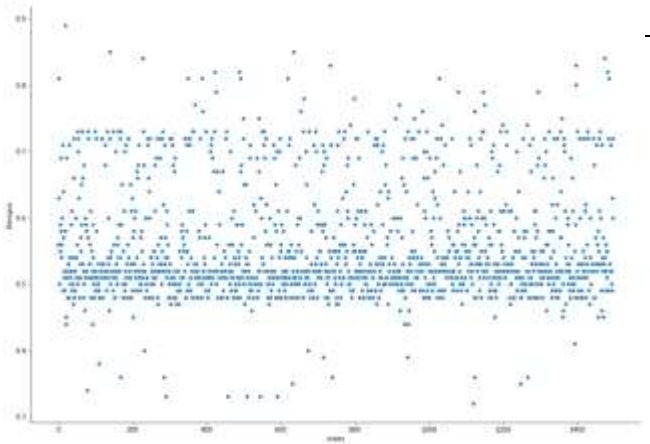
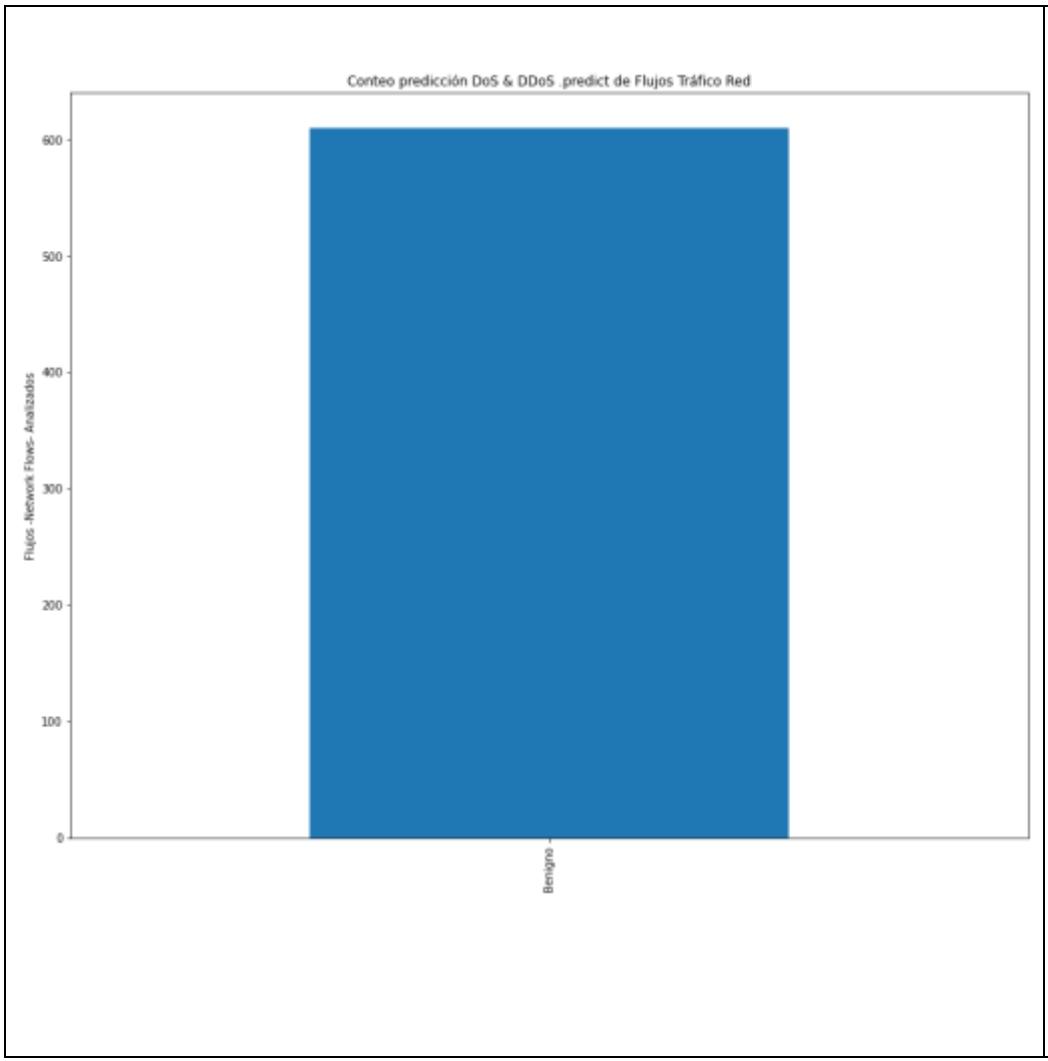
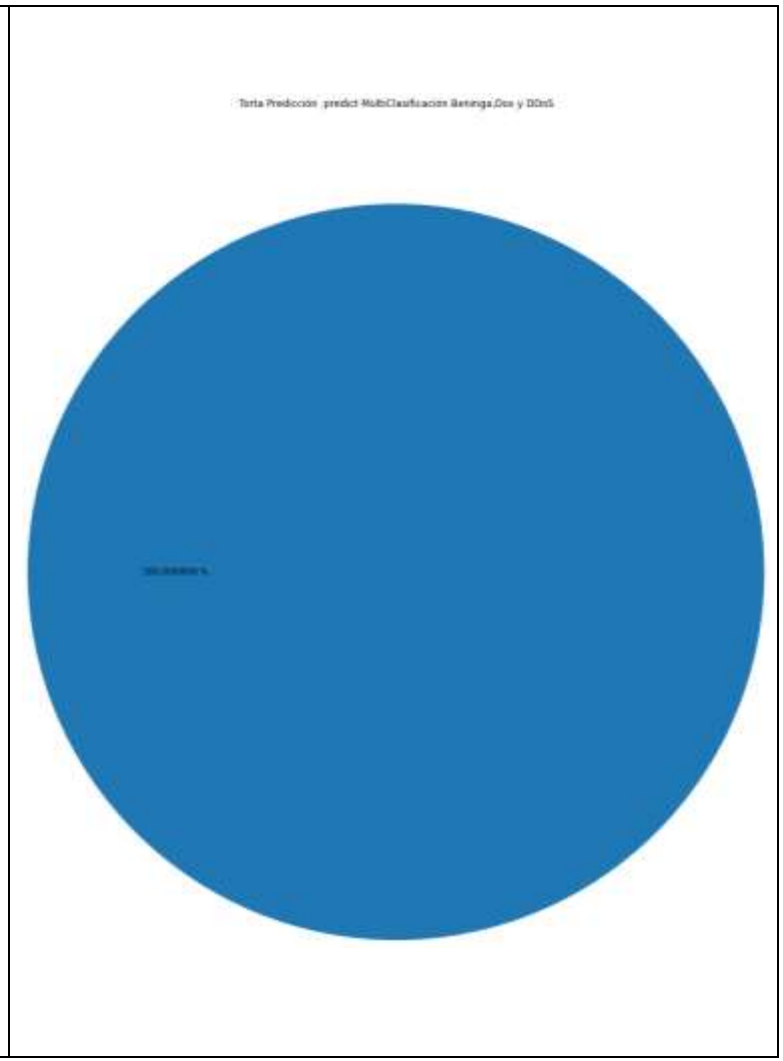


Tabla 5-11 Resultados Gráficos para presunto ataque DATASET DoS RARO [Elaboración propia]

5.6.2.6. Resultados Gráficos de tome un pcap esnifado TODO BENIGNO





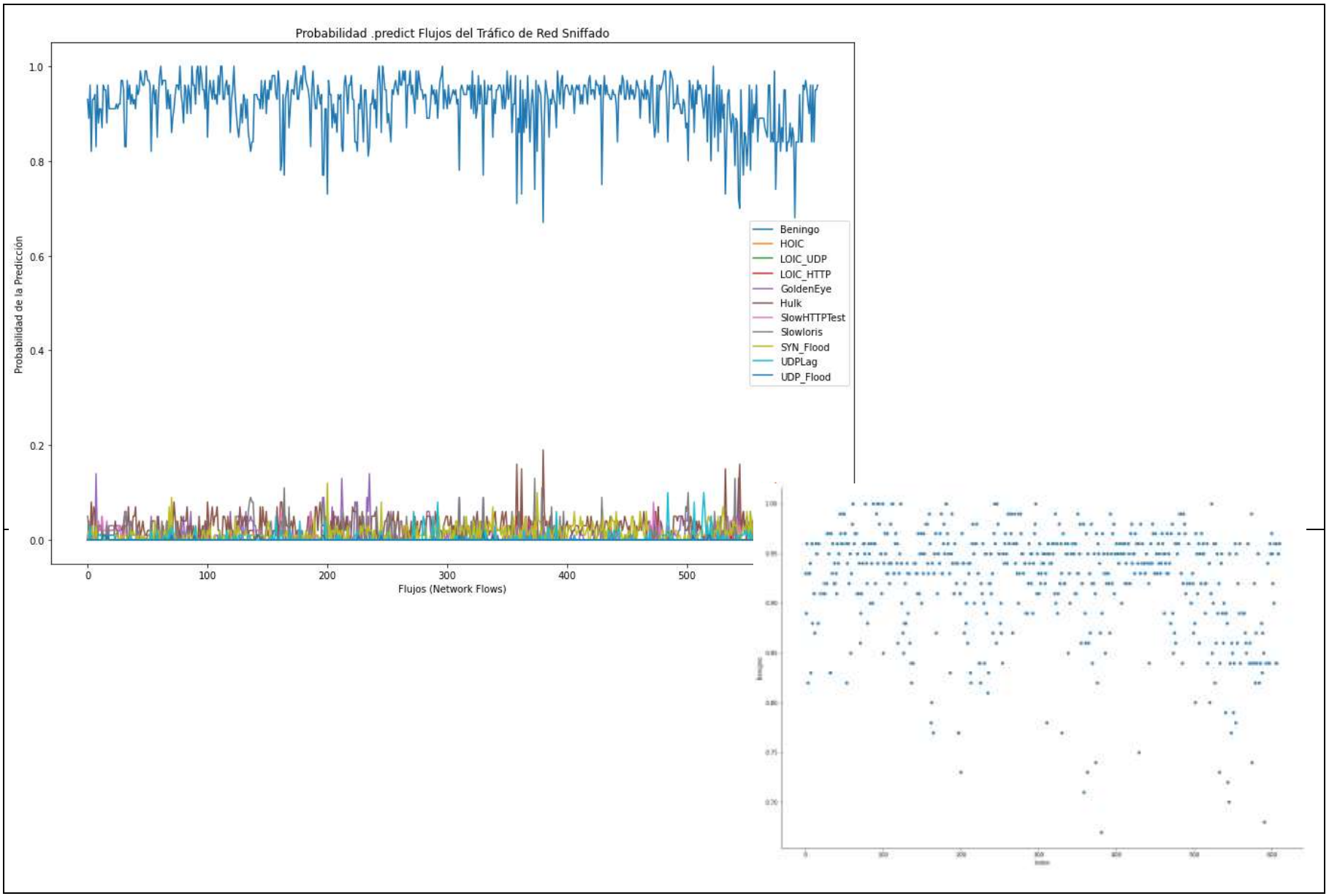


Tabla 5-12 Resultados Gráficos de tome un pcap esnifado TODO BENIGNO [Elaboración propia]

Capítulo 6

CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- ✓ Caracterizar un ataque malicioso.

Al elegir los ataques maliciosos de denegación DoS y denegación distribuida DDoS, no pudo ser mejor la elección como problema informático a afrontar solución que implica que los servicios sean interrumpidos, y se comprometa la disponibilidad de dichos servicios y datos. con esta intromisión al mundo técnico y experimental, es innegable e incuestionable el gran alcance y avance de los ataques de denegación aumentan rápidamente tanto en frecuencia como en intensidad, con informes recientes de ataques que alcanzan 1,7 Tbps, a tal punto que se presencia el aumento del fenómeno de denegación como un servicio, del inglés DoS-as-a-Service, plasmado no más en los booters y zombies. Ante esa gravedad, solo queda la solución de muchas capas, siguiendo fielmente la filósofa de la Defensa en Profundidad, agregando capas que complementen, agregue valor defensivo y aporte a soluciones basadas en el comportamiento, en nuestro caso propuesto en

el análisis del flujo de tráfico de red con técnicas de aprendizaje automático, donde presencialmente todo el valor y la fundamentación del aprendizaje radica en la calidad y consistencia del conjunto de datos del cual aprender; y más ahora, si estos están abiertos y disponibles de manera pública. Allí es donde aparece el Instituto CIC de la UNB, con sus estupendos datasets que han trabajado desde años, y que aportan al mundo investigativo mundial gracias a que son abiertos. La explicación al porque nos casamos completamente con los dataset de 2017, 2018 y 2019 del CIC, es imperativo y necesario gracias a sus ventajas que brinda, por medio mencionarlas, la mayoría de los conjunto de datos anteriores carecen del origen del conjunto de datos siendo sintético en lugar del mundo real, del tráfico completo, de la diversidad de ataques, datos etiquetados, Protocolos disponibles de la heterogeneidad de las fuentes de datos, de la interacción de la configuración de red completa y de la captura completa. Pero sobre todo al Conjunto de atributos extraídos, siendo en sus 84 atributos brindar una total y completa información estadística, que caracteriza y modela el deseado comportamiento del tráfico de red.

- ✓ Análisis.

El análisis del planteamiento del problema propuesto, y que motivo la realización de un modelo detector con técnicas basadas en aprendizaje automático se sugiere afrontar mediante el tratamiento y Transformación total de los conjuntos de datos que dispone el CIC y que se obtiene también con la herramienta de ellos CICFlowMeter, en numerosos archivos con extensión de fichero de variables por comas extensión .csv, mediante los procedimientos de preprocesamiento y procesamiento del conjunto de datos. En el preprocesamiento de se da forma a los datos, se prepara los datos en crudo, en primario y separados por individual para luego ser limpiados y concatenados en uno solo y único conjunto. Este conjunto extenso unificado el procesado para ser comprimido, reducción de su

dimensionalidad en tanto a cantidad de atributos, aplicando un análisis riguroso matemático cuyo diseño presenta selección de los atributos con técnicas mucho más allá de los tradicional, pues ya se ha probado buenas técnicas Gain Ratio, Info Gain, ANOVA F-test Percentile, ReliefF Algorithm, Recursive Feature Elimination (RFE), Cost-Sensitive F-Measures, Envolturas (Wrapper methods), Filtros(Filter methods), entre otras como PCA pero son muy simples, y que no van allá; como si lo llega hacer una perspectiva más matemática, de la aplicación en cascada, en secuencia de primero algo matemáticamente más simple usando el concepto de desviación estándar (standard deviation) se eliminan todas los atributos con una cantidad cero, nula de variación pasando de 79 atributos a 69 atributos, solo 10 atributos reducidos; para luego la secuencia del segundo paso con la aplicación de SciPy un ecosistema de software matemático de código abierto que contiene entre otros a NumPy, donde usando la agrupación jerárquica (.cluster.hierarchy) en las correlaciones de orden de rango de Spearman, se obtiene atributos correlacionados que se encuentran en un cluster o racimo muy cercano y son eliminados. Este buen y acertado análisis matemático hace reducir la dimensionalidad de 69 a 37 atributos que caracterizan fielmente toda la información total, pero si mejorando optimización, al reducir redundancia. Gráficamente se ve maravilloso en la complejidad de un conjunto de datos tan grande (1.087'701.840 celdas de data muchos float 64) del mapa de calor, y del Dendrograma de atributos.

- ✓ Diseño

El diseño del modelo detector y clasificador de ataques maliciosos DDoS y DoS, se centra en su pilar del cerebro, del núcleo, del Proceso de Aprendizaje, gracias a la diversidad de algoritmos de aprendizaje supervisado existen para el problema de clasificación de múltiples clases. Nosotros probamos más de 30 algoritmos que dispone la gran librería scikit-learn, y aunque es una herramienta de programación

única y que amplía el conocimiento de la ciencia e ingeniería de datos, de una forma muy creativa. Así también es difícil el requerimiento de máquina que exige grandes conjuntos de datos, que ponen la capacidad de procesamiento de la maquina al límite, porque su exigencia en cantidad de datas de entrena en muy considerable, tanto que en los más de 30 algoritmos aplicados solo mostraron resultados 20, y uno de ellos, el de crentroide de vecinos cercanos demoro 14 horas para finalizar su entrenamiento. Es así, que solo se adopta un pódium y se descartan los demás, ya que no muestran competitividad respecto al desempeño de dicho pódium. en esto orden, nuestro modelo propuesto llamado DetD&DoS constituye en su etapa de aprendizaje un entrenamiento basado en la validación cruzada para los algoritmos de pódium, y más por el respaldo de las múltiples métricas que proporciona el método Cross Validate() de la clase model selection. Luego de los buenos y prometedores resultados de validación cruzada, se prosigue a aplicar mediante los métodos que incluye la librería scikit de cada algoritmo, inspeccionar la importancia de atributos. feature_importances_ o oob_score_, y para los Arboles Unitarios como DT y ET el método tree_ junto a las funciones plot_tree, export_text, export_graphviz de la clase tree. Por ultimo cabe resaltar la necesidad de almacenar dicho proceso de entrenamiento, tanto es así que la librería XGBoost si implementa como método propio guardar, hacer un save y luego proseguir en ese punto, que por cierto esperaba mucho más de dicho extremo del refuerzo del gradiente, o al menos falto seguir mejorando la descripción y elección correcta de parámetros de los XGB para sincronizar y ponerlos al punto, pero en verdad su número de falsos negativos como falsos positivos, no superaron ni al Random Forest, ni al Decisión Tree y esos que estos últimos no pudieron ser optimizados sus hiperparámetros fue imposible su cálculo computacional, y dichos hiperparámetros están la mayoría por default.

- ✓ Evaluación

La evaluación de 2 conjuntos de prueba arroja resultados muy buenos, y que inspirar en mejorar. debo reconocer que, hasta este punto, no comprendida bien el poder tan abrumador y el alcance que posee el poder contar con las etiquetas dentro de un conjunto de datos, más allá de los aprendizajes no supervisados, donde no se disponen de los datos etiquetados, resulta que la evaluación depende de ello, y la visualización de los resultados y sus respectivas métricas, valores de rendimiento y desempeño solo se pueden comparar respecto a las etiquetas reales. específicamente en nuestros resultados de evaluación obtenidos se ve que en el conjunto de prueba de casi un millón cuatrocientos flujos se obtienen valores casi perfectos, o muy cercanos, parte de ello, por ser estratificados, es decir equilibrados. y esto lleva a los resultados del segundo conjunto de prueba que se formó ser nunca antes visto, no pertenecía al conjunto de entrenamiento, y tenían una peculiaridad de ser muy desequilibrado, casi todos tipos intrusivos, de denegación y la minoría absoluta de flujos benignos, lo que resulto que si detectara identificado ataque de denegación, pero que al momento de clasificar en familias presentara una equivocación o falla al clasificar a que clase o familia era, confundimiento entre udp-lag y syn flood, punto que se deseó mejorar, pero que imposibilito la máquina.

- ✓ Implementación

la implementación que fue realizada en mi red hogar, y donde se capturo tráfico de red, a través de la interfaz del puerto ethernet, para el caso eno1. Fue utilizada la herramienta de CICFlowMeter que extrae los atributos y características de la conversión de los biflujos del tráfico de red 2 formas diferentes tanto de los que son esnifados en tiempo real, como las capturas de paquetes almacenadas por anterioridad. es por ello, que se obtuvieron algunas trazas de paquetes capturados de posibles ataques o que dicen ser ataques en las fuentes de internet de una

manera exhaustiva, por la verdad es demasiado difícil, encontrar archivos así sean individuales de trazas de ataques, tanto que ante los pocos y los muy pocos que no contenían error, se recurrió una solución alternativa de probar y comprobar los resultados de la implementación con otro data set de la literatura disponible como UMW15 que cuanta con los paquetes capturados, pero que al ser este todo de datos sintéticos, no tiene ningún flujo valido, al no ser del mundo real. y puntualizo el problema para comprobación al no contar con ejemplos de muestras de ataques DoS y DDoS, de trazas de ataques disponibles, tanto que ni de un mismo data set del 2015. esa dificultad de disponibilidad abierta limita.

6.2. Trabajo Futuro y recomendaciones

Deseo mencionar y expresar que se deseó hasta el máximo mejorar en el proceso de entrenamiento, conformando un amplio y extenso conjunto de entrenamiento, pero fue imposible, ya que llegue hasta donde mi maquina llego; y puedo decir que hasta enfermizo de obsesión, hasta que aterrice la realidad al saber que específicamente el archivo de TFTP.csv del día 01-12 del CIC-DDoS-2019 que pesa 9,3GB pues sencillamente mi maquina no lo puede ni cargar, ni leer, ni nada, es inasequible; así tal cual cuando solo tenía 2 dimm, y con 16GB era imposible en ese momento de cargar, de hacer lectura de Thuesday-20-02-2018_TrafficForML_CICFlowMeter.csv que pesa 4,1GB. Para contextualizar a los lectores del poder de capacidad computacional que es requiero los ordenadores para el uso de conjuntos de datos extensos, como es caso nuestro de comas separadas por variables de 84 atributos y columnas, y decenas y miles de millones de filas.

En cuanto a los aspectos que no fueron abarcados en este trabajo de grado, pero que sería importante considerarlos en trabajos futuros se puede mencionar:

- Sería necesario entrenar con un conjunto de entrenamiento mejorado, que se obtuvo el 15 de abril que llame DataFrame_14Abril21_ModDet_2021_CIC-2017_CIC-2018_CIC-2019.csv; ya está listo para ser leído, para mí fue imposible leerlo, con un tamaño de 9.3GB.
- En lo posible seria disponer con una máquina de nivel investigación, o ver la viabilidad de procesamiento como un servicio, pero me imagino el requerimiento en el ancho de banda, para incluir un pretencioso detector de intrusiones completo abarcando los dataset completos disponibles por CIC que son 4 años 2016, 2017, 2018 y 2019. Es pretenciosos y presuntuoso porque solo los 2 últimos que forman la mayoría, son 32,1GB para un total de 28 archivos .csv que, con ello, poder ser entrenados por el modelo de aprendizaje propuesto, u otro, pero si sería equiparable quizás a un motor de análisis de un sistema IDS.

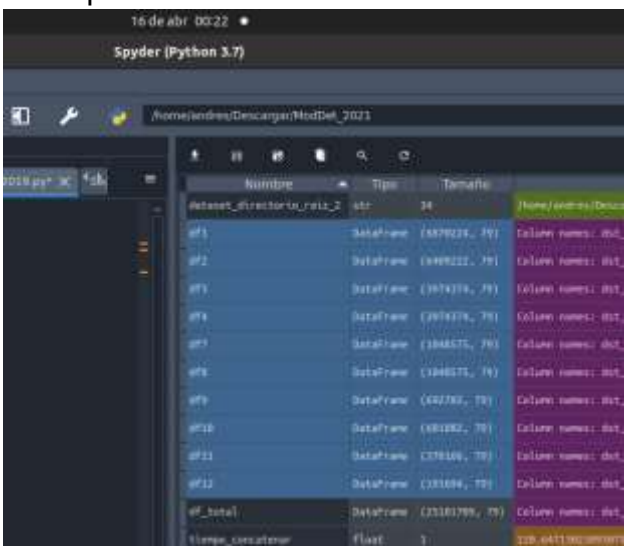


Figura 6-1 Conjunto que combine para más de 25 millones de flujos

Capítulo 7

BIBLIOGRAFÍA

[1] Kaspersky Lab. Más información sobre el malware y cómo proteger todos tus dispositivos [en línea]. AO Kaspersky Lab, 2019 [consulta: 09 de febrero de 2019]. Disponible en: https://latam.kaspersky.com/resource-center/preemptive-safety/what-is-malware-and-how-to-protect-against-it

[2] T. G. Nguyen, T. V. Phan, B. T. Nguyen, C. So-In, Z. A. Baig, nand S. Sanguanpong, “Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks,” IEEE access, vol. 7, pp. 107 678–107 694, 2019.

[3] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, “Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods,” IEEE Access, vol. 7, pp. 51 691–51 713, 2019.

[4] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, “Iot-flock: An open-source framework for iot traffic generation,” in 2020 International Conference on Emerging Trends in Smart Technologies (ICETST). IEEE, 2020, pp. 1–6.

[5] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (ddos) attack dataset and taxonomy,” in 2019

International Carnahan Conference on Security Technology (ICCST). IEEE, 2019, pp. 1–8.

[6] F. Hussain, M. Husnain, and F. Shahzad, “IoT DoS and DDoS Attack Detection using ResNet,” in 2020. IEEE, 2020. DOI: 10.21203/rs.3.rs-120303/v1.

[7] J. Zhang, P. Liu, J. He, and Y. Zhang, “A Hadoop based analysis and detection model for IP Spoofing typed DDoS attack,” in 2016 IEEE TrustCom-BigDataSE-ISPA, 2016, pp. 1978–1985, doi: 10.1109/TrustCom.2016.300.

[8] P. Machaka and A. Bagula, “Using Exponentially Weighted Moving Average Algorithm to Defend Against DDoS Attacks,” in IEEE 2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics, 2016.

[9] S. Hajar et al., “A Neural Network Model for Detecting DDoS Attacks Using Darknet Traffic Features,” in IEEE 2016 International Joint Conference on Neural Networks (IJCNN), 2016, no. November 2014, pp. 2979–2985.

[10] Steve Weisman. What is a distributed denial of service attack (DDoS) and what can you do about them?. [en línea]. NortonLifeLock, July 23, 2020 [consulta: 28 de mayo de 2021]. Disponible en: https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by-norton.html

[11] David Warburton. DDoS Attack Trends for 2020. [en línea]. F5 Application threat intelligence, May 07, 2021 [consulta: 28 de mayo de 2021]. Disponible en: https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020

[12] White paper Cisco public. Cisco Annual Internet Report (2018–2023). [en línea]. CISCO, March 9, 2020 [consulta: 28 de mayo de 2021]. Disponible en: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[13] ACIS. LEVEL 3 PRESENTA INFORME DE AMENAZAS Y ATAQUES EN TODA AMéRICA LATINA. [en línea]. Asociación Colombiana de Ingenieros de Sistemas ACIS [consulta: 28 de mayo de 2021]. Disponible en: https://acis.org.co/portal/content/level-3-presenta-informe-de-amenazas-y-ataques-en-toda-am%C3%A9rica-latina

[14] Equipo de investigación tictac, “Ciberseguridad en entornos cotidianos,”, CCIT, 2020. [consulta: 28 de mayo de 2021]. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiporKs773xAhX_STABHbcDDX0QFjAAegQIAxAD&url=https%3A%2F%2Fwww.ccit.org.co%2Fwp-content%2Fuploads%2Fciberseguridad-en-entornos-cotidianos-vfene-1.pdf&usg=AOvVaw2MZwrhWOR72zj8Ei_yOF-z

[15] Equipo de investigación, equipo de policía nacional, “informe de las tendencias del cibercrimen en Colombia 2019-2020,” Octubre 29 de 2019. [consulta: 28 de mayo de 2021]. Disponible en: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjX6cq0873xAhVSRDABHVlNA30QFjAAegQIAxAD&url=https%3A%2F%2Fwww.ccit.org.co%2Fwp-content%2Fuploads%2Finforme-tendencias-cibercrimen_compressed-3.pdf&usg=AOvVaw3ObVGIOdKktQw0k95F3XZf

[16] Standardization), ISO (International Organization for. "Publicly Available Standards”. [consulta: 28 de mayo de 2021]. Disponible en: http://standards.iso.org/ittf/PubliclyAvailableStandards/c041933_ISO_IEC_27000_2009.zip

[17] "ISTQB Standard glossary of terms used in Software Testing”. [consulta: 28 de mayo de 2021]. Disponible en: http://glossar.german-testing-board.info/

[18] S. Karnouskos: Stuxnet Worm Impact on Industrial Cyber-Physical System Security. In:37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), Melbourne, Australia, 7-10 Nov 2011. Retrieved 20 April 2014.

[19] Internet Security Glossary. doi:10.17487/RFC2828. RFC 2828.

[20] Dan Daniels. 14 Network Security Tools and Techniques to Know. [en línea]. Gigamon blog, June 13, 2019 [consulta: 23 de mayo de 2021]. Disponible en: https://blog.gigamon.com/2019/06/13/what-is-network-security-14-tools-and-techniques-to-know/

[21] Josh Fruhlinger. What is network security? Definition, methods, jobs & salaries. [en línea]. IDG Communications, JUL 3, 2018 [consulta: 23 de mayo de 2021]. Disponible en: https://www.csoonline.com/article/3285651/what-is-network-security-definition-methods-jobs-and-salaries.html

[22] Marina. Seguridad perimetral informática. Qué es y objetivos. [en línea]. Grupo Atico34, 19 febrero, 2021 [consulta: 24 de mayo de 2021]. Disponible en: https://protecciondatos-lopd.com/empresas/seguridad-perimetral-informatica/#Honeypots

[23] accensit_admin. Types of Network Security. [en línea]. Accensit, junio 2nd, 2017 [consulta: 24 de mayo de 2021]. Disponible en: https://www.accensit.com/blog/seguridad-perimetral-informatica-informacion-necesaria/

[24] UNIR REVISTA, INGENIERÍA Y TECNOLOGÍA. Seguridad perimetral informática: objetivos y plataformas recomendables. [en línea]. UNIR - Universidad Internacional de La Rioja, 30/07/2020 [consulta: 24 de mayo de 2021]. Disponible en: https://www.unir.net/ingenieria/revista/seguridad-perimetral-informatica/

[25] Liao, Hung-Jen & Lin, Chun-Hung & Lin, Ying-Chih & Tung, Kuang-Yuan. (2013). Intrusion detection system: A comprehensive review. Journal of Network and

Computer Applications 36 (2013) 16–24. 2017. http://dx.doi.org/10.1016/j.jnca.2012.09.004.

[26] Hoang XD, Hu J, Bertok P. A program-basedanomaly intrusion detection scheme using multipledetection engines and fuzzy inference. Journal ofNetwork and Computer Applications 2009; 32(6):1219–1228.

[27] Kumar, Sunil & Dutta, Kamlesh & Zolkipli, Mohamad & Inayat, Zakira & Khan, Suleman & Anthony Jnr, Bokolo & Chang, Victor. (2016). Intrusion detection in mobile ad hoc networks:techniques, systems, and future challenges. SECURITY AND COMMUNICATION NETWORKS. Security Comm. Networks 2016; 9:2484–2556. DOI: 10.1002/sec.1484.

[28] Scarfone K, Mell P. Guide to intrusion detection andprevention systems (IDPS). Technical Report, Na-tional Institute of Standards and Technology (NIST)Special Publication 800-94. Department of Com-merce, U.S., 2007.

[29] Chandola V, Banerjee A, Kumar V. Anomaly detec-tion: a survey. ACM Computing Surveys (CSUR)2009; 41(3):1–15.

[30] Jyothsna, Veeramreddy & Munivara,Koneti. (2018). Anomaly-Based Intrusion Detection System. DOI: 10.5772/intechopen.82287.

[31] Garcia-Teodoro & Diaz-Verdejo & Macia-Fernandez & Vazquez. (2008). Anomaly-based network intrusion detection: Techniques, systems and challenges. 2008. doi:10.1016/j.cose.2008.08.003.

[32] Khraisat, Ansam & Gondal, Iqbal & Vamplew, Peter & Kamruzzaman, Joarder. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. 2019. https://doi.org/10.1186/s42400-019-0038-7.

[33] Tsai C-F et al. Intrusion detection by machine learning: A review. Expert Systems with Applications. 2009;36(10):11994-12000

[34] Jia, Bin & Ma, Yan & Huang, Xiaohong & Lin, Zhaowen & Sun, Yi. (2016). A Novel Real-Time DDoS Attack Detection Mechanism Based on MDRA Algorithm in Big Data. Mathematical Problems in Engineering. 2016. http://dx.doi.org/10.1155/2016/1467051.

[35] Ericka Chickowski. Types of DDoS attacks explained. [en línea]. CyberSecurity AT&T, JULY 8, 2020 [consulta: 26 de mayo de 2021]. Disponible en: https://cybersecurity.att.com/blogs/security-essentials/types-of-ddos-attacks-explained

[36] Steve Weismann, NortonLifeLock. "¿Qué es un ataque distribuido de denegación de servicio (DDoS) y qué puede hacer al respecto?" 2020.

[37] David Balaban. Are you Ready for These 26 Different Types of DDoS Attacks?. [en línea]. Security Magazine, May 7, 2020 [consulta: 26 de mayo de 2021]. Disponible en: https://www.securitymagazine.com/articles/92327-are-you-ready-for-these-26-different-types-of-ddos-attacks

[38] Bashar Ahmed Khalaf, Salama A. Mostafa, Aida Mustapha, Mazin Abed Mohammed, Wafaa Mustafa Abduallah. "Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods". 2019. IEEE Access Special Section on Artificial Intelligence and Cognitive Computing for Communication and Network. Digital Object Identifier 10.1109/ACCESS.2019.2908998.

[39] S. S. Kolahi, K. Treseangrat, and B. Sarrafpour, "Analysis of UDP DDoS flood cyber attack and defense mechanisms on Web Server with Linux Ubuntu 13," in Proc. Int. Conf. Commun., Signal Process., Their Appl.,Feb. 2015, pp. 1-5.

[40] M. V. Kumar and R. Umar, "Identifying and blocking high and low rate DDOS ICMP flooding," Indian J. Sci. Technol., vol. 8, p. 32, Aug. 2015.

[41] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "n empirical evaluation of information metrics for low-rate and high-rate DDoS attack detection," Pattern Recognit. Lett., vol. 51, pp. 1-7, Jan. 2015.

[42] R. K. Chang, "Defending against flooding-based distributed denial of- service attacks: A tutorial," IEEE Commun. Mag., vol. 40, no. 10, pp. 42-51, Mar. 2002.

[43] W. M. Eddy, "Syn Flood attack," in Encyclopedia Cryptography Security, New York, NY, USA: Springer, 2011, pp. 1273-1274.

[44] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, "OPERETTA: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD attacks against web servers," Comput. Netw., vol. 92, no. 1, pp. 89-100, 2015.

[45] S. M. Specht and R. B. Lee, "Distributed denial of service: Taxonomies of attacks, tools, and countermeasures," in Proc. ISCA PDCS, Sep. 2004, pp. 543-550.

[46] Hongyu Liu, Bo Lang. "Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey". 2019. Applied Sciences. doi:10.3390/app9204396.

[47] Mohammad Waseem. How To Implement Classification In Machine Learning?. [en línea]. Edureka, Brain4ce Education Solutions Pvt. Ltd, Jul 21,2020[consulta: 30 de mayo de 2021]. Disponible en: https://www.edureka.co/blog/classification-in-machine-learning/

[48] Frank Krüger. "Activity, Context, and Plan Recognition with Computational Causal Behaviour Models". 2017. University of Rostock, Thesis.

[49] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Intrusion Detection Evaluation Dataset (CIC-IDS2017). [en línea]. Canadian Institute for Cybersecurity | UNB, 2017 [consulta: 28 de mayo de 2021]. Disponible en: https://www.unb.ca/cic/datasets/ids-2017.html

[50] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. CSE-CIC-IDS2018 on AWS. [en línea]. Communications Security Establishment (CSE) & Canadian Institute for Cybersecurity | UNB, 2018 [consulta: 28 de mayo de 2021]. Disponible en: https://www.unb.ca/cic/datasets/ids-2018.html

[51] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani. DDoS Evaluation Dataset (CIC-DDoS2019). [en línea]. Canadian Institute for Cybersecurity | UNB, 2019 [consulta: 28 de mayo de 2021]. Disponible en: https://www.unb.ca/cic/datasets/ddos-2019.html

[52] Vahid Mirjalili, Sebastian Raschka, *Python Machine Learning: Aprendizaje Automatico y aprendizaje profundo con Python, scikit-learn y TensorFlow (Spanish Edition)*. Tapa blanda – 2019. Lugar de publicación: Marcombo, Segunda Edicion, 21 febrero 2019. Nº Páginas: 618. ISBN: 978-84-267-2720-6

[53] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

[54] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019

[55] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

[56] Raschka, Sebastian, (2015). Python Machine Learning. Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics,

Birmingham - Mumbai: Packt Publishing Ltd, open source community experience distilled, pp. 99-102

[57] Kazil, Jacqueline & Jarmul, Katharine (2016). Data Wrangling with Python. tips and tools to make your life easier, Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc., pp. 162-177