Detección de anomalías en redes usando técnicas de aprendizaje automático.
Anomaly detection in networks using machine learning.

Trabajo de Fin de Grado
Curso 2023–2024

Autor
Pablo Heredero García

Director
Rafael Aurelio Moreno Vozmediano

Doble Grado en Ingeniería Informática y Matemáticas
Facultad de Informática
Universidad Complutense de Madrid

Detección de anomalías en redes usando técnicas de aprendizaje automático.

Anomaly detection in networks using machine learning.

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Pablo Heredero García

Director
Rafael Aurelio Moreno Vozmediano

Convocatoria: *Mayo* 2024

Doble Grado en Ingeniería Informática y Matemáticas
Facultad de Informática
Universidad Complutense de Madrid

27 de mayo de 2024

Dedicatoria

A mi madre y mi hermano, por su imprescindible apoyo todos estos años de estudio.

Agradecimientos

A mi tutor, Rafael, por ayudarme en todo momento y guiarme en el desarrollo de este trabajo, con una forma de trabajar que ha hecho más fácil una tarea compleja.

Resumen

Detección de anomalías en redes usando técnicas de aprendizaje automático.

Cada día millones de personas y organizaciones en todo el mundo utilizan sistemas informáticos y se conectan a Internet con todo tipo de propósitos. En las últimas décadas, el uso de Internet se ha extendido extraordinariamente, hasta llegar a ser un elemento fundamental en las comunicaciones de la sociedad y la economía actuales. Paralelamente a este desarrollo, se ha visto cómo los ataques informáticos a través de la red se han multiplicado exponencialmente. Estos ataques pueden suponer una grave amenaza para los usuarios o las instituciones que los sufren, por lo que resulta de vital importancia poder combatirlos. Los sistemas de detección de intrusos son una buena respuesta ante este problema, y concretamente aquellos basados en la detección de anomalías resultan especialmente eficaces. Esto, unido a los últimos avances en el campo del aprendizaje automático, resulta una combinación realmente prometedora. En este trabajo se estudian distintas posibilidades para implementar los sistemas de detección de anomalías en redes utilizando modelos de aprendizaje automático. Los datos de tráfico de red han tomado de los datasets públicos CICIDS2017 y UNSW-NB15, los cuales presentan una amplia variedad de ataques. En cuanto a los modelos de aprendizaje automático, se utilizan 12 algoritmos supervisados y 4 no supervisados. Para cada uno de ellos, y con distintos enfoques y configuraciones de parámetros, se han realizado pruebas y recogido los datos de rendimiento, los cuales han resultado de un nivel muy alto.

Palabras clave

Aprendizaje automático, IDS, detección de anomalías, CICIDS2017, UNSW-NB15, ataque informático, aprendizaje supervisado, aprendizaje no supervisado, F-Score.

Abstract

Anomaly detection in networks using machine learning.

Every day, millions of people and organizations worldwide use computer systems and connect to the Internet for various purposes. In recent decades, the use of the Internet has expanded extraordinarily, becoming a fundamental element in today's society and economy's communications. Alongside this development, there has been an exponential increase in cyberattacks across Internet. These attacks can suppose a serious threat to users or institutions that suffer them, making it vitally important to combat them. Intrusion detection systems are a good response to this problem, and specifically those based on anomaly detection are particularly effective. This, combined with the latest advances in the field of machine learning, is a truly promising combination. This work examines different possibilities for implementing anomaly detection systems in networks using machine learning models. Network traffic data has been taken from the public datasets CICIDS2017 and UNSW-NB15, which present a wide variety of attacks. As for the machine learning models, 12 supervised and 4 unsupervised algorithms are used. For each of them, and with different approaches and parameter configurations, tests have been conducted and performance data collected, which have yielded very high levels of performance.

Keywords

Machine learning, IDS, anomaly detection, CICIDS2017, UNSW-NB15, cyberattack, supervised learning, unsupervised learning, F-Score.

Índice de acrónimos

Acrónimo	Significado
IDS	Instrusion Detection System
FTP	File Transfer Protocol
IRC	Internet Relay Chat
DoS	Denial of Service
DDoS	Distributed Denial of Service
U2R	User to Remote
R2L	Remote to Local
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
IoT	Internet of Things
CSV	Comma-Separated Values
XSS	Cross-Site Scripting
MITM	Man In The Middle
HTTP	Hypertext Transfer Protocol
TCP	Transmission Control Protocol
PHP	Hypertext Preprocessor
SQL	Structured Query Language
GNB	Gaussian Naïve Bayes
MNB	Multinomial Naïve Bayes
ID3	Iterative Dichotomiser 3
RF	Random Forest
KNN	K Nearest Neighbours
AB	AdaBoost
GBC	Gradient Boosting Classifier
MLP	Multi-Layer Perceptron
QDA	Quadratic Discriminant Analysis
LR	Logistic Regression
SVM	Support Vector Machine
SGD	Stochastic Gradient Descent
IF	Isolation Forest
LOF	Local Outlier Factor
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
IP	Internet Protocol

Tabla 1: Índice de acrónimos

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Motivación

En el mundo globalizado y altamente conectado en el que vivimos, la importancia de Internet es cada día mayor. A diario millones de usuarios en todo el planeta se conectan a la red para comunicarse entre sí o con empresas e instituciones de todo tipo. En las últimas dos décadas el número de usuarios de Internet a nivel global se ha multiplicado extraordinariamente, superando los 5.400 millones de usuarios en 2023, lo que supone un 69 % de la población mundial [26].

Ante esta realidad imparable, se puede observar también cómo paralelamente al crecimiento de Internet se ha multiplicado el número de ataques en la red, ya sea a usuarios particulares o a corporaciones de cualquier tipo. Estos ataques pueden suponer un gran riesgo en términos económicos, de privacidad, de seguridad o incluso geopolíticos.

En un panorama global en el que la inmensa mayoría de instituciones y empresas a nivel mundial funcionan digitalmente, en el que el tratamiento de datos masivos cobra cada vez más importancia, y en el que la irrupción de nuevas tecnologías en el campo de la computación está transformando la realidad, cada vez resulta más crítico defenderse ante este tipo de ataques informáticos.

A la hora de evitar dichos ataques, la primera línea de defensa de cualquier red o sistema suele ser el cortafuegos, el cual realiza un primer filtro al tráfico de entrada o salida, descartando posibles amenazas. Sin embargo, muchos de los ataques consiguen superar el cortafuegos, y es ahí donde interviene el segundo nivel de defensa en profundidad: el sistema de detección de intrusos (IDS, por sus siglas en inglés). Este sistema se basa en la suposición de que el comportamiento del intruso difiere del de un usuario legítimo de manera cuantificable. De esta forma, es capaz de identificar el tráfico de un atacante y, si la intrusión se detecta suficientemente rápido, la amenaza podrá ser identificada y neutralizada antes de que realice algún daño.

En la actualidad existen fundamentalmente dos tipos de IDS, los basados en reglas y los basados en detección de anomalías.

La detección basada en reglas, también llamada detección de firmas, utiliza una base de datos en la que se define un conjunto de reglas o patrones de ataques (firmas) para decidir si un comportamiento dado es una intrusión o no. Este método es bastante eficaz, pero tiene el inconveniente de que las bases de datos necesitan actualizarse constantemente y procesar nueva información sobre ataques. Además, incluso si las bases de datos están actualizadas, siguen siendo vulnerables a los ataques de tipos nuevos, que no hayan sido vistos previamente. Dado que estos ataques no están en la base de datos, no se pueden prevenir. En este método se tienen que describir reglas muy específicas de sistemas y ataques, para lo que es necesario detectar la vulnerabilidad concreta y no necesariamente el ataque, lo que puede resultar complicado. El resultado suele ser una gran cantidad de reglas a procesar, lo cual puede resultar poco eficiente en caso de tener que analizar grandes cantidades de tráfico. Además, dado que buena parte del tráfico actual viaja encriptado y no se puede ver su contenido directamente, resulta especialmente difícil para estos sistemas detectar ataques entre este tipo de tráfico.

Por otro lado, los sistemas de detección de intrusos basados en anomalías detectan el comportamiento inusual examinando el tráfico en la red, comparándolo con el comportamiento de los usuarios legítimos. Detectan desviaciones de este comportamiento normal, que podrían ser ataques, aplicando pruebas estadísticas. Hoy en día, gracias al avance de los algoritmos de aprendizaje automático, esta metodología está avanzando considerablemente, pudiéndose automatizar procesos antes costosos y obtener resultados muy prometedores. Una de las ventajas que tiene la detección de anomalías es que sí puede detectar ataques nunca antes vistos, ya que puede compararlos igualmente con el tráfico normal. Además, puesto que estos sistemas analizan propiedades generales como el tamaño o el tiempo de conexión, pueden detectar ataques incluso entre el tráfico cifrado sin necesidad de leer su contenido. Sin embargo, uno de los grandes problemas que presenta la detección de anomalías es que para "entrenar" el sistema de detección suelen ser necesarias grandes cantidades de datos, tanto de tráfico normal como de ataques, y no siempre son fáciles de conseguir. En cualquier caso, estos IDS son cada vez más utilizados en detección y prevención de ataques.

Vistas las ventajas que presentan los IDS basados en anomalías, en este trabajo se van a estudiar en detalle este tipo de sistemas, concretamente aquellos que utilizan algoritmos de aprendizaje automático. Además, se pondrán a prueba con datos de tráfico reales utilizando datasets públicos, para después recoger y analizar el rendimiento de los mismos.

1.2. Objetivos

Este trabajo sigue el camino marcado por K. Kostas, en su estudio "Anomaly Detection in Networks Using Machine Learning" [3], en el que se hace igualmente un estudio de distintos algoritmos de aprendizaje automático para después aplicarlo en detección de anomalías sobre un dataset y estudiar los resultados. El objetivo del presente trabajo es ampliar el estudio de K. Kostas en múltiples direcciones, con la idea de obtener

un resultado más amplio y con el que se puedan concluir resultados más completos.

Teniendo en cuenta lo anterior, los objetivos que se pretenden alcanzar al terminar este trabajo son los siguientes:

- **Explorar y seleccionar datasets para entrenar el IDS.** Se explorarán distintos datasets disponibles y se seleccionarán dos de ellos para entrenar el sistema de detección de intrusos. De entre todos se escogerán aquellos que resulten más interesantes teniendo en cuenta sus características y el objetivo de este estudio. Además, se buscará que los datasets guarden relación con otros trabajos similares (como es el de K. Kostas [3]) para poder así comparar y ampliar los resultados de dichos estudios con los de este.

- **Examinar los algoritmos de aprendizaje automático que se puedan usar para la detección de anomalías.** De entre todos se escogerán, además de los usados en el trabajo de K. Kostas [3], aquellos que resulten prometedores o interesantes de comparar. La idea en este caso es ampliar muy notablemente el número y la variedad de algoritmos utilizados, permitiendo una comparación más amplia. En particular, se utilizarán algoritmos tanto supervisados como no supervisados, lo que permitirá una comparación muy interesante entre estos dos enfoques del aprendizaje automático.

- **Aplicar los algoritmos de aprendizaje automático seleccionados.** Se implementará código para detectar ataques utilizando ambos datasets y recoger los resultados. Esto se hará, para ambos datasets, desde dos enfoques distintos: separadamente por cada tipo de ataque presente en los datos, y sobre el conjunto total de datos sin distinguir el tipo de ataque. A su vez, para la implementación se estudiarán distintas selecciones de parámetros del tráfico y se compararán los resultados para cada una de ellas. El código implementado en el desarrollo de este trabajo se pondrá a disposición del público en general, junto con la memoria, a través de este repositorio de GitHub.

- **Estudiar los resultados obtenidos en la fase de evaluación.** Una vez construidos los modelos de detección de anomalías, se procederá a la evaluación de los mismos, recogiendo los resultados de rendimiento de la clasificación. Dada la variedad en cuanto a algoritmos usados, datos y tipos de ataques, se obtendrán muchos e interesantes datos de salida, los cuales se estudiarán y compararán adecuadamente.

1.3. Plan de trabajo

Antes de comenzar con el desarrollo concreto del trabajo, se estudiarán los conceptos generales más importantes sobre los sistemas de detección de anomalías. Esto proporcionará un conocimiento previo fundamental a la hora de entender y enfocar el trabajo siguiente.

Siguiendo con la documentación previa, se explorarán los distintos datasets disponibles para detección de anomalías, estudiando y comparando las características de cada uno. Además del dataset utilizado en el trabajo de K. Kostas [3], el CICIDS2017, se escogerá, de entre todos los datasets analizados, otro que resulte interesante atendiendo a sus características técnicas. De esta forma, posteriormente se podrá realizar con ambos datasets un trabajo análogo y poder así combinar y comparar los resultados que se obtengan, teniéndose en el proceso una gran variedad de datos y de ataques.

Por otro lado se explorarán los algoritmos de aprendizaje automático disponibles y que puedan utilizarse en detección de anomalías. De entre todos se escogerán y estudiarán varios de ellos para su posterior implementación dentro del IDS. Además de los usados en el trabajo de K. Kostas [3], se seleccionarán aquellos que resulten prometedores o interesantes de comparar, incluyéndose en este caso algoritmos de aprendizaje no supervisado. La idea en este caso es ampliar muy notablemente el número y la variedad de algoritmos utilizados, permitiendo una comparación más amplia.

Una vez entendidos los aspectos fundamentales de la detección de anomalías, la última tarea de documentación consistirá en examinar las publicaciones previamente realizadas en este campo, haciendo especial énfasis en aquellas más actuales. Puesto que, como se ha explicado en la Sección anterior, este trabajo es en buena medida una ampliación del realizado por K. Kostas [3], la documentación alrededor de este trabajo centrará buena parte de nuestra atención.

Una vez estudiada y recogida toda la información previa necesaria, se pasará a la fase de implementación. En ella se analizarán y preprocesarán ambos datasets para después desarrollar múltiples programas en Python que implementen el sistema de detección de anomalías utilizando los algoritmos seleccionados anteriormente. Dichos programas leerán los registros del dataset, con los que entrenarán el modelo de detección. Una vez preparado el modelo, se pondrá a prueba su rendimiento clasificando tráfico del mismo dataset, recogiendo los resultados de rendimiento que se obtengan.

Por último, se compilarán y analizarán los resultados recogidos al ejecutar los modelos para así comparar los distintos algoritmos aplicados en cada dataset. De esta manera, se podrán concluir resultados comparativos que arrojen luz sobre qué modelos funcionan mejor o peor dependiendo de los parámetros, los ataques o los datos.

Capítulo 2

Estado de la Cuestión

En este capítulo se recogen todos los conocimientos previos necesarios para desarrollar este estudio. En primer lugar se incluye una introducción a la detección de anomalías, que en nuestro caso aplicaremos para detectar ataques informáticos utilizando algoritmos de aprendizaje automático. A continuación se estudiarán una serie de datasets que recogen tráfico de red con anomalías de diferentes tipos, y se escogerán aquellos que resulten interesantes para este estudio. Una vez escogidos los datasets, en el siguiente apartado veremos los ataques informáticos presentes en cada uno de ellos en forma de anomalías del tráfico. Se incluye también una sección sobre los algoritmos de aprendizaje automático que se van a utilizar más adelante. Por último, se hará un estudio de los trabajos más recientes y relacionados con este trabajo, para tenerlos en cuenta en los pasos posteriores.

2.1. Detección de anomalías

Tal y como se ha expuesto en el Capítulo 1, la detección de anomalías es una tarea crucial a la hora de identificar intrusiones o ataques en una red antes de que sea demasiado tarde. Poder identificar en tiempo real un comportamiento sospechoso y abortarlo antes de que cause daños en los sistemas es fundamental si se pretende mantener segura una red. En esta sección vamos a introducir conceptos importantes sobre estas metodologías.

Se entiende por detección de anomalías a los procedimientos que detectan datos anómalos o anormales dentro de un conjunto de datos. Esta tarea suele involucrar descubrir patrones extraños o llamativos en los datos. Se considera una anomalía *"una observación que se desvía tanto del resto como para resultar sospechosa de haber sido generada por otro mecanismo"* [6]. Las anomalías se consideran muy importantes ya que pueden indicar situaciones realmente graves, que requieran una reacción rápida para solventarlas. Si un computador presenta un patrón de tráfico inusual, esto puede indicar que ha sido hackeado y que está transfiriendo sus datos a destinos no autorizados. Pero la detección de anomalías no sólo resulta crítica en el ámbito de la informática: un comportamiento anómalo en las transacciones de una tarjeta de crédito

puede indicar actividades fraudulentas, o una anomalía en una imagen por resonancia magnética podría indicar la presencia de un tumor maligno. Por ejemplos como estos, la detección de anomalías se viene utilizando en multitud de ámbitos, como la salud, detección de fraudes, detección de intrusos, procesamiento de imágenes, róbótica, etc.

El proceso de detección de anomalías puede resumirse a grandes rasgos con el diagrama de la Figura 2.1. Los datos de entrada han de preprocesarse ya que suelen ser de distinto tipo (variables categóricas y numéricas) y presentar ciertos errores que conviene corregir. Las técnicas de detección de anomalías, generalmente de aprendizaje automático (que pueden ser supervisadas o no supervisadas), se aplican sobre los datos. La salida proporcionada se analiza dependiendo del tipo que sea, que puede ser evaluada o etiquetada, como veremos a continuación.

La detección de anomalías en general, y concretamente la aplicada a la seguridad en redes, no resulta tan directa o fácil como pudiera pudiera parecer en un primer momento. A pesar de la gran cantidad de técnicas disponibles, la detección de anomalías presenta las siguientes dificultades:

- No existe una técnica de detección de anomalías universal. Un sistema de detección de intrusos para una red cableada, por ejemplo, no resultaría muy útil en una red inalámbrica.
- Los conjuntos de datos suelen contener ruido, ya sea en forma de datos atípicos, entradas mal etiquetadas o valores extremos. Este ruido en los datos suele ser una anomalía en sí mismo, y por tanto es difícil de clasificar. En oncasiones conviene realizar un filtrado previo de los datos para eliminar estos problemas, pero no siempre es fácil.
- No hay demasiados conjuntos de datos etiquetados que sean públicos, y que por tanto se puedan utilizar para entrenar modelos de detección de anomalías desde un enfoque supervisado.
- Dado que los comportamientos de las redes y los computadores están evolucionando continuamente, y los hackers realizan ataques nunca antes vistos y cada vez más complejos, las técnicas de detección de anomalías actuales pueden no resultar lo suficientemente efectivas. Esto da lugar a una necesidad continua de nuevos datasets y de técnicas más sofisticadas de detección.

Como hemos dicho, las anomalías se presentan como patrones en los datos que no se ajustan a las características de los patrones usuales. Dependiendo de la naturaleza de las anomalías, podemos clasificarlas en los siguientes tipos:

- **Anomalía puntual:** se da cuando un dato concreto se desvía del patrón normal del dataset. Si el consumo habitual de un coche es de 5L diarios de combustible, y un día consume 40L, se considera una anomalía puntual.
- **Anomalía contextual:** también llamada condicional, ocurre cuando un dato se comporta de forma anómala en un contexto particular. Un gasto superior al nor-

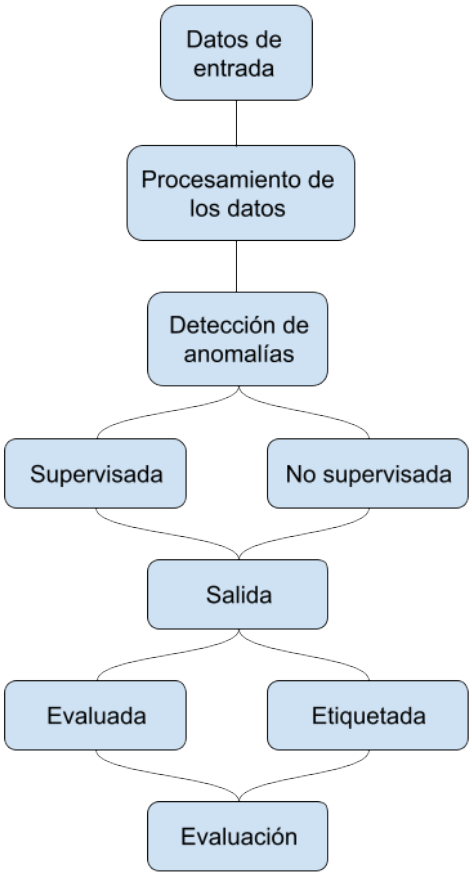

Figura 2.1: Estructura de la detección de anomalías

mal no se considerará extraño si ocurre en Navidad, pero el mismo desembolso en cualquier otro mes puede resultar sospechoso.

- **Anomalía colectiva:** se da cuando hay una colección de datos que se comportan de forma anómala con respecto al conjunto total de datos. Si, por ejemplo, en un electrocardiograma aparecen valores bajos durante mucho tiempo, puede indicar alguna patología cardiaca. Sin embargo, un valor bajo pero aislado no resulta anómalo.

Otro aspecto importante en la detección de anomalías es la forma en la que se representa la salida. Esto determina el análisis posterior, y generalmente es de una de las dos formas siguientes:

- Salida evaluada: las técnicas de detección asignan un valor de anomalía a cada instancia de los datos. Con esos valores, se ordenan todos los datos y el analista establece un umbral a partir del cual un dato se ha de considerar anómalo.
- Salida etiquetada: las técnicas de detección asignan una etiqueta, habitualmente binaria, que marca si los datos son normales o anómalos.

En este estudio utilizaremos técnicas de detección de anomalías cuya salida está etiquetada.

Restringiéndonos al campo de la informática, existe una gran variedad de ataques posibles en una red. Como veremos más adelante, en el apartado de 2.3 de Ataques, cada uno utiliza técnicas distintas e intenta explotar vulnerabilidades diferentes, por lo que la muestra de tráfico que dejan cada uno es diferente también. Esto, unido a que los atacantes utilizan continuamente técnicas nuevas, hace que la detección de intrusos sea realmente difícil.

Existe también una gran cantidad de técnicas de detección de anomalías, cada una de ellas basadas en estrategias y principios distintos. En este estudio nos centraremos en aquellas que utilizan métodos de aprendizaje automático, que por su potencia de cómputo resultan realmente efectivas. Concretamente, tal y como veremos en el apartado 2.4 de Aprendizaje Automático, veremos técnicas tanto supervisadas como no supervisadas.

2.2. Datasets

A la hora de crear los sistemas de detección de anomalías, y en particular aquellos que se basan en aprendizaje automático, se necesitan grandes cantidades de datos de tráfico que reflejen tanto flujo normal como maligno. A partir de estos datos, se entrena el modelo para que una vez listo sea capaz de clasificar tráfico nuevo y detectar en él posibles amenazas. Sin embargo, no es posible utilizar tráfico real de forma pública por motivos de privacidad. Para remediar esta limitación, algunas empresas e instituciones generan conjuntos de datos artificiales pero verosímiles, y otras anonimizan registros reales de tráfico para poder así usar y publicar los datos.

Dependiendo de cómo se quiera diseñar el IDS y de los recursos disponibles para ello, se deberá elegir un conjunto de datos u otro para entrenar el modelo. En esta sección se van a explorar los distintos datasets disponibles para este propósito, detallando para cada uno sus características técnicas y las ventajas e inconvenientes que presentan. En la Tabla 2.1 se resumen todos estos datos. A partir de esta información, se escogerán los dos datasets que resulten más apropiados para los modelos de este trabajo.

2.2.1. DARPA98

Este dataset [4] [9], creado en 1998 por el laboratorio MIT Lincoln con financiación de DARPA, simula la red local de la Fuerza Aérea de los Estados Unidos. Los datos recogidos consisten en procesos como transferencia de archivos vía FTP, búsquedas en internet, envíos y llegadas de e-mails y mensajes IRC. Además de distinguir entre tráfico normal/benigno, se incluyen 38 ataques distintos, agrupados en tipos Denial of Dervice (DoS), User to Remote (U2R), Probe y Remote to Local (R2L).

Este dataset ha recibido fuertes críticas, sobre todo por no reflejar tráfico real, por no estar actualizado, y por no incluir flujo que pudiera ser clasificado como falso positivo. Sin embargo, este dataset reviste gran importancia por ser la fuente de creación de los otros conjuntos como KDD Cup 99 y NSL-KDD.

2.2.2. KDD99

Este dataset [27] [29], desarrollado en 1999 por la Universidad de California para The Third International Knowledge Discovery and Data Mining Tools Competition (La KDD Cup '99), incluye trazas de tráfico de siete semanas en forma de 5 millones de registros binarios, divididos en subconjuntos de entrenamiento y test. Para cada entrada se indican 41 parámetros del tráfico y 38 tipos distintos de ataques agrupados en U2R, DoS, Probe y R2L. De estos ataques, 14 aparecen exlusivamente en la sección de test para representar ataques desconocidos.

En comparación con su predecesor DARPA98, el dataset KDD99 resulta mucho más adecuado para aplicar métodos de aprendizaje automático al estar dividido en entrenamiento y test, además de ofrecer la posibilidad de controlar la detección de ataques.

2.2.3. NSL-KDD

El dataset NSL-KDD [28][18], creado en 2009 por Tavallaee et al. para la University of New Brunswick, surgió como alternativa al KDD99 por los problemas que presentaba este último en cuanto al número de repeticiones y el excesivo tamaño. Estos defectos resultaban en un mediocre desempeño al ser utilizado con algoritmos de aprendizaje automático.

El NSL-KDD corrige estos errores y repeticiones del KDD99, lo cual mejora sensiblemente los resultados al utilizar este dataset. Sin embargo, todavía presenta algunas limitaciones como el reducido número de observaciones para entrenamiento así como de subconjuntos para test. Esto, unido a algunos otros errores detectados más adelante, resulta en una limitada capacidad de este dataset para reflejar tráfico real en la red.

2.2.4. ISCX2012

El dataset ISCX2012 [25][17], desarrollado en 2012 por el Canadian Intitute for Cybersecurity (CIC), recoge el tráfico de Internet generado durante siete días por el banco de pruebas de dicho instituto.

ISCX2012 se construyó a partir del tráfico de dispositivos reales usando diferentes protocolos. Todos los datos están etiquetados y presenta ataques de infiltración, DoS, DDoS, y fuerza bruta SSH. Sin embargo, no recoge tráfico SSL/TLS, que hoy en día supone más de la mitad del tráfico total en Internet. Por esta razón, aunque refleja bien el tráfico de máquinas reales, se considera que no es adecuado para las necesidades actuales.

2.2.5. Ton_IoT

El dataset ToN-IoT [11][30], desarrollado en 2020 por UNSW Canberra, incluye datos de telemetría de dispositivos conectados, registros de sistemas operativos Linux y Windows, y tráfico de red de IoT industrial, entre otros.

ToN-IoT contiene más de 22 millones de datos de tráfico, representados en formato CSV, con etiquetas que indican el comportamiento normal/malicioso y el tipo concreto de ataque, que puede ser DoS, DDoS, data injection, ransomware, password attack, backdoor, cross-site scripting (XSS), scanning, y ataque MITM. Estos ataques se lanzaron y dirigieron a varios sensores IoT e IoT industriales, dentro de una red de IoT de escala media.

Que los datos sean tan actuales, unido al gran tamaño del dataset, hace que este sea un conjunto muy prometedor. Sin embargo, puesto que en el presente trabajo los recursos hardware son limitados, este dataset resulta excesivamente grande, por lo que no se considerará para su uso.

2.2.6. CIDDS-001

El dataset CIDDS-001 [20][14](Coburg Intrusion Detection Datasets), desarrollado en 2017 por la Universidad de Coburgo, recoge el tráfico simulado de una pequeña red de empresa desplegada con OpenStack.

El tráfico malicioso simulado incluye ataques de tipo DoS, fuerza bruta y escaneo de puertos. El resto del tráfico, clasificado como normal, fue generado con scripts de python respetando los horarios de trabajo de la empresa. En total cuenta con 33 millones de entradas de datos, lo que lo convierte en un dataset de tamaño considerable, fuera del alcance computacional disponible en este trabajo. Por esta razón, igual que con el ToN-IoT, este dataset no se considerará para la fase experimental.

2.2.7. CSE–CIC–IDS2018

El dataset CSE-CIC-IDS2018 [23][15], desarrollado en 2018 por el Canadian Institute for Cybersecurity (CIC) y el Communications Security Establishment (CSE) de Canadá, recoge tráfico simulado específicamente para su uso en IDS.

Este dataset presenta siete tipos distintos de ataques: fuerza bruta, Heartbleed, Botnet, DoS, DDos, ataques web e infiltración en la red desde dentro. Sus 16 millones de registros fueron generados por perfiles de tipo "B"(420 máquinas y 20 servidores de tráfico benigno) y de tipo "M"(50 máquinas de tráfico malicioso). El hecho de que cuente con tal cantidad de de datos lo convierte en un dataset prometedor, pero lamentablemente y como ya hemos comentado para los dos datasets anteriores, el CSE–CIC–IDS2018 resulta excesivamente grande para utilizarlo en este trabajo, por lo que no se considerará para su uso.

2.2.8. BETH

El dataset BETH [7][10](BPF-Extended Tracking Honeypot), desarrollado en 2021 por Kate Highnam, Kai Arulkumaran, Zachary Hanif y Nicholas R. Jennings, es uno de los más modernos y completos que hay disponibles en la actualidad. Recoge el tráfico de 23 máquinas distintas (honeypots), resultando en 8 millones de registros de flujo benigno, sospechoso o maligno.

Cada entrada contiene información de 14 parámetros distintos, pero en este caso no distingue entre tipos de ataque. A pesar de lo actualizado y lo completo de este dataset, dado que el objetivo de este trabajo es hacer también un estudio por tipo de ataque, no lo consideraremos para su uso en la fase experimental.

2.2.9. CICIDS2017

El dataset CICIDS2017 [24][16], creado en 2017 por el Canadian Institute for Cybersecurity (CIC) en la Universidad de Nuevo Brunswick, recoge el tráfico generado durante 5 días (3 julio - 7 julio de 2017) en una red de computadores con máquinas reales y modernas, con sistemas operativos como Windows, Linux, Kali o Mac.

De entre sus 2.8 millones de entradas, el tráfico benigno responde al comportamiento normal con protocolos de email, HTTP, HTTPS, SSH O FTP. Por otro lado, el flujo malicioso incluye ataques de 14 tipos distintos, incluyendo Fuerza bruta FTP, Heartbleed, Fuerza bruta SSH, DDoS, infiltration DoS, ataque web y botnet.

Cada entrada cuenta con 84 parámetros distintos, además de una etiqueta adicional que marca si es benigno o, en su defecto, el tipo de ataque del que se trata.

Este dataset tiene la ventaja de recoger tráfico de máquinas reales, además de contar con sistemas operativos y tipos de ataques actuales. Presenta una gran diversidad de sistemas operativos y de ataques, e incluye diferentes protocolos de tráfico benigno. Ha de tenerse en cuenta también que el tamaño de los datos, aunque grande, es manejable para un sistema como el que se utilizará en este trabajo.

Por otro lado, como inconveniente, al tratarse de un dataset de 2017, puede que el conjunto de ataques no esté completamente actualizado. Además, no incluye archivos separados para entrenamiento y test (como sí ocurre con otros datasets), por lo que esta tarea se deja al usuario.

2.2.10. UNSW-NB15

El dataset UNSW-NB15 [12][31], creado en 2015 por la Universidad de Nuevo Brunswick, muestra 2.5 millones de entradas de datos de tráfico generadas a partir de varios servidores virtuales con la herramienta IXIA.

Además del dataset global con todos los datos, cuenta con una sumbuestra de dos archivos separados, para utilizarlos directamente como datasets en entrenamiento y

test. En total recoge 9 tipos distintos de ataques, concretamente Fuzzers, Shellcode, Analysis, Worms, Backdoors, Reconnaissance, DoS, Generic y Exploits.

Para cada entrada de tráfico hay 49 etiquetas distintas que recogen toda la información sobre los parámetros del flujo. Concretamente, hay dos etiquetas que marcan la naturaleza del tráfico (maligno/benigno) y el tipo de ataque, respectivamente.

Este dataset presenta una buena variedad de ataques, y sobre todo una gran cantidad de etiquetas para cada entrada. Esto es fundamental a la hora de aplicar algoritmos de aprendizaje automático con los datos. Se trata de datos razonablemente modernos (2015), y el volumen total es grande pero manejable para este caso (2-3 millones de entradas, similar al CICIDS2017).

Por otra parte, al tratarse de datos de 2015 quizá no estén tan actualizados como idealmente se querría. Además, en este caso el tráfico se ha recogido utilizando servidores virtuales, es decir, simulando tráfico real y no recogiendo datos reales de máquinas físicas.

Dataset	Número de registros (millones)	Tipos de ataques	Año
DARPA98	5	DoS, U2R, Probe, R2L	1998
KDD99	5	DoS, U2R, Probe, R2L	1999
NSL-KDD	5	DoS, U2R, Probe, R2L	2009
ISCX2012	2.4	DoS, Fuerza Bruta, DDoS	2012
UNSW-NB15	2.54	Fuzzers, Shellcode, Analysis, Worms, Backdoors, Reconnaissance, DoS, Generic y Exploits	2015
CIDDS-001	33	Fuerza bruta, DDoS y escaneo de puertos	2017
CICIDS2017	2.8	Fuerza bruta FTP, Heartbleed, Fuerza bruta SSH, DDoS, infiltration DoS, ataque web y botnet	2017
CSE-CIC-IDS2018	16.2	Fuerza bruta, DDoS y ataques web	2018
ToN-IoT	16	DDoS, ransomware, data injection, DoS, password attack, cross-site scripting (XSS), backdoor, man-in-the-middle (MITM) y escaneo	2020
BETH	8	(No distingue)	2021

Tabla 2.1: Características de los datasets

Habiendo estudiado las características de cada dataset (que pueden verse resumi-

das en la Tabla 2.1), con sus ventajas e inconvenientes, finalmente se decide utilizar en este trabajo los datasets CICIDS2017 y UNSW-NB15. En el caso del primero, estaba decidido de antemano ya que fue este el que se utilizó en el trabajo de Kostas[3]. Además, presenta caracteríasticas muy favorables, como la variedad de protocolos y ataques.

De entre los demás, se ha escogido el UNSW-NB15 por su variedad de ataques y etiquetas, además de su tamaño total. En estos términos, es un dataset similar al CICIDS2017, por lo que estudio comparativo puede resultar especialmente interesante. Ambos datasets pueden encontrarse en abierto en formato CSV, lo cual supone una ventaja importante. Pese a no ser datos actualizados al máximo, la variedad que presentan en conjunto dará lugar a conclusiones más amplias, tal y como nos habíamos marcado como objetivo. Los detalles sobre todos los parámetros recogidos en ambos datasets pueden verse en el Apéndice A.

2.3. Ataques

En esta sección vamos a realizar un análisis previo de los dos datasets con los que trabajaremos, centrándonos en la distribución y los tipos de ataques que presentan. Realizaremos este estudio separadamente para cada dataset, puesto que presentan características notablemente distintas.

2.3.1. CICIDS2017

Como se ha visto en la Sección 2.3, el dataset CICIDS2017 presenta 2.830.743 entradas de datos, de las que un 17 % responden a tráfico malicioso. La naturaleza de cada entrada del tráfico viene dada por una etiqueta, que puede tomar 15 valores distintos. Uno de ellos (Benign) representa el tráfico normal, mientras que los otros 14 representan ataques, especificando el tipo. La distribución de ataques, tanto en términos globales como por tipo, puede verse en las Figuras 2.2. y 2.3.

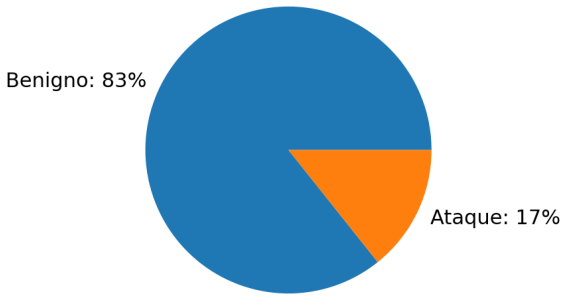

Figura 2.2: Proporción tráfico benigno/ataque (CICIDS2017)

Se pueden apreciar grandes diferencias en cuanto al número de entradas para unos tipos de ataque u otros. El ataque DoS HULK, por ejemplo, supone casi la mitad del

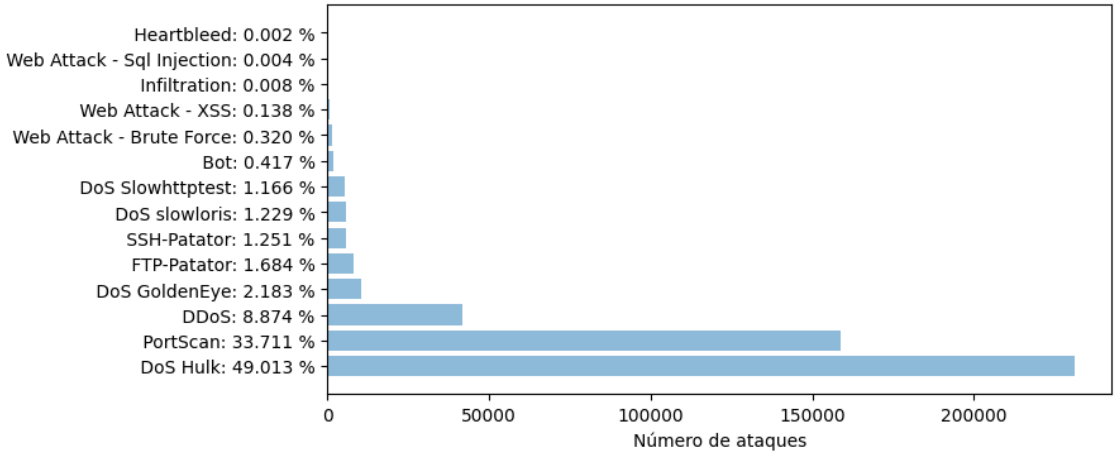

Figura 2.3: Distribución de los ataques (CICIDS2017)

tráfico malicioso, mientras que de tipo Heartbleed sólo hay 11 entradas. Esto se debe a la distinta naturaleza de cada ataque, hay algunos que por sus características generan mucho más tráfico que otros.

Con el objetivo de entender mejor los datos que estamos manejando, vamos a estudiar, uno por uno, todos los tipos de ataques presentes en el dataset CICIDS2017.

2.3.1.1. DoS

Los ataques de denegación de servicio (DoS, por sus siglas en inglés) tienen como objetivo inhabilitar el uso de un sistema, de forma que quede completamente inaccesible para sus usuarios legítimos. Habitualmente esto se consigue sobrecargando el sistema objetivo con tareas o información hasta que sature y quede bloqueado, dejando de prestar servicio. Existe una gran variedad de ataques de tipo DoS, dependiendo de cuántas máquinas están involucradas, qué herramientas se utilicen o qué tipo de mensajes se envíen. A continuación vemos los tipos de ataques DoS que están presentes en el dataset CICIDS2017.

- **DoS HULK:** La herramienta HULK (HTTP Unbearable Load King) está diseñada para realizar una gran cantidad de peticiones únicas a un servidor con el objetivo de saturarlo y bloquearlo. Es una herramienta realmente potente, capaz de inhabilitar un servidor en cuestión de un minuto. Durante el ataque, se realizan inundaciones de tipo TCP-SYN (Transmission Control Protocol - Synchronization) y HTTP-GET. Veamos en qué consisten estas dos técnicas de inundación:

  En la inundación TCP-SYN se explota la apertura TCP de 3 vías del servidor, colapsándolo con nuevas peticiones de conexión que quedan incompletas y a la espera. El atacante (cliente) manda el mensaje SYN a la víctima (servidor), el cual responde con un paquete SYN-ACK. Sin embargo, el atacante no envía el paquete ACK que se espera, por lo que la conexión se mantiene incompeta. Al realizar esta operación múltiples veces, se llena la cola de solicitudes del servidor y este queda completamente saturado.

El comando HTTP GET lo utilizan los clientes para solicitar un archivo a un servidor web. En la inundación HTTP GET, el atacante envía al servidor una gran cantidad de solicitudes de este tipo, de forma que la capacidad del servidor de recibir nuevas conexiones legítimas queda anulada.

- **DDoS:** El ataque de denegación de servicio distribuido (DDoS, por sus siglas en inglés) es una evolución del ataque DoS. Está diseñado igualmente para saturar un sistema objetivo, pero en este caso se realiza desde múltiples máquinas origen con un mismo sistema objetivo. Esta es una de las formas más fáciles y habituales de ciberataque por su sencillez y eficacia. En muchas ocasiones, las máquinas atacantes son sistemas terceros que han sido infectados con un software malicioso y que se coordinan en remoto por el atacante. Estos dispositivos se denominan "zombis", y la red creada por los mismos se denomina "botnet".

- **DoS Goldeneye:** La herramienta Goldeneye permite realizar ataques de denegación de servicio desde una o más máquinas. Tiene la particularidad de realizar ataques en paralelo (multihilo), de forma que una sola máquina puede ser capaz de dejar inutilizado al servidor. Establece conexiones TCP completas y realiza una cantidad moderada de solicitudes HTTP. Activando los modos Keep-Alive y No-Cashed, esta herramienta puede rápidamente consumir los recursos del servidor con una cantidad relativamente pequeña de tráfico HTTP completamente legal.

- **DoS Slowloris:** Este ataque, programado en Pearl y disponible para Windows y Linux, realiza inundaciones TCP-SYN (Ver DoS HULK) para consumir los recursos de la víctima y bloquear así el sistema. Llamado así por el loris perezoso, el primate asiático que se mueve de lentamente, un ataque de Slowloris puede durar mucho tiempo si pasa desapercibido. El atacante establece numerosas conexiones HTTP con la víctima y se realizan varias solicitudes "lentas", que nunca terminan, con el objetivo de mantener las conexiones abiertas el mayor tiempo posible. Gracias a esto, este tipo de ataque consume muy poco ancho de banda y consigue acaparar los recursos del servidor con solicitudes que parecen más lentas de lo normal, pero que por lo demás imitan el tráfico regular.

- **DoS SlowHTTPTest:** Este tipo de ataque abusa de la ventana de envío del sistema receptor (la víctima) para saturarlo y que quede inutilizado. El atacante envía solicitudes HTTP normales al servidor, pero al recibir su respuesta establece el tamaño de ventana a prácticamente 0, de forma que la recepción de mensajes se ralentiza al máximo. El servidor deberá reservar recursos durante mucho tiempo, y sólo podrá enviar al atacante paquetes de un tamaño realmente pequeño, alargando notablemente la conexión. De esta forma, si se establecen varias conexiones de esta forma, el servidor satura sus recursos. Este tipo de ataque no necesita un gran ancho de banda ni un hardware potente, y resulta especialmente difícil de detectar ya que las conexiones HTTP parecen inocentes.

2.3.1.2. PortScan

Los puertos de un sistema son puntos de conexión numerados del 0 al 65635, y se usan para determinar el tipo de conexión que se establece desde la máquina. El escaneo de puertos es un tipo común de ataque informático mediante el cual el atacante analiza los puertos abiertos en la máquina víctima para recopilar información sobre ella. De esta forma el atacante puede descubrir información crítica de la víctima como los dispositivos conectados, el sistema operativo, los procesos en ejecución y el estado de los puertos. En un ataque PortScan el atacante envía mensajes a cada puerto y en función de la respuesta se sabe si el puerto está en uso o no, y con esa información se descubren los detalles del sistema y sus debilidades. Existen varios tipos de ataques PortScan dependiendo del tipo de mensajes que se envíen y los puertos objetivo. Habitualmente se combinan varios tipos distintos para que la información obtenida sea mayor.

2.3.1.3. FTP-Patator

El protocolo FTP (File Transfer Protocol) se utiliza para la transferencia de archivos entre un cliente y un servidor dentro de una misma red. Este protocolo exige un nombre y una contraseña válidos para transferir los ficheros. La herramienta Patator realiza un ataque de fuerza bruta para intentar adivinar dicho nombre y contraseña. Los ataques de fuerza bruta como este realizan una gran cantidad de intentos para adivinar los datos privados y así poder suplantar la identidad del usuario legítimo. Estos ataques suelen presentar una gran cantidad de intentos fallidos, pero se aprovechan de la debilidad habitual de las contraseñas para conseguir descifrarlas.

2.3.1.4. SSH-Patator

El protocolo SSH (Secure Shell) es un estándar criptográfico que proporciona un mecanismo de autenticación para que los usuarios controlen o modifiquen en remoto un determinado servidor. Este protocolo se utiliza habitualmente para acceder en remoto a un sistema alojado en la red. En este ataque, en el que de nuevo se utiliza la herramienta Patator, el atacante intenta conseguir acceso remoto a un sistema y tomar el control sobre él. Para ello se llevan a cabo tres etapas de ataque:

Escaneo: En primer lugar se intenta aprender lo máximo sobre el sistema objetivo (ver PortScan). Particularmente, se intenta encontrar el host que esté ejecutando el servicio SSH con un escaneo de puertos específico.

Fuerza bruta: El atacante intenta acceder al sistema probando con una gran cantidad de usuarios y contraseñas distintas (ver FTP-Patator).

Toma de control: El atacante consigue encontrar las cedenciales de acceso correctas, inicia sesión, y se hace con el control del sistema como si fuera un usuario legítimo.

Dependiendo de en qué etapa del ataque se encuentre, SSH-Patator presenta flujos

de tráfico distintos. En la primera fase se observa una gran cantidad de paquetes TCP-SYN incompletos, mientras que en la segunda se ven bastantes paquetes TCP completos de tamaño pequeño.

2.3.1.5. Botnet

Una red de bots, o "botnet" es un grupo de computadores infectados con un determinado malware que funcionan coordinados por el atacante para llevar a cabo el ataque. Dichas máquinas, también llamadas bots o zombis, funcionan sin conocimiento de sus dueños legítimos y bajo la orden del atacante pueden llevar a cabo de forma sincronizada un ataque de tipo SPAM (emails no deseados) o DDoS.

En CICIDS2017, este tipo de ataque se ha llevado a cabo utilizando la herramienta Ares. Durante el ataque se pueden observar grandes cantidades de paquetes de buen tamaño y con el flag PSH activo. Sin embargo no es fácil detectar una botnet antes de lanzar el ataque, ya que no deja huella de ninguna actividad sospechosa.

2.3.1.6. Web Attack

Estos son un tipo particular de ataques informáticos que explotan las vulnerabilidades de aplicaciones web, ya sea en el lado del cliente o del servidor. Habitualmente tienen como objetivo bloquear, tomar el control u obtener información sensible de aplicaciones web atacando al servidor en el que se encuentran alojadas. En el caso de CICIDS2017, se dan tres subtipos de ataques web, lanzados contra aplicaciones web de PHP y MySQL. Dichos ataques son los siguientes:

- **Fuerza bruta:** Este tipo de ataque se ha detallado en el ataque FTP-Patator.

- **XSS:** Algunas páginas web presentan una vulnerabilidad de tipo XSS (Cross-Site Scripting), por la cual un atacante puede introducir scripts maliciosos en ellas. Cuando la víctima intenta ver la página web, el código malicioso modifica el funcionamiento normal de la aplicación, pudiendo dar lugar a robos de datos o suplantación de identidad del cliente.

- **SQL Injection:** SQL (Structured Query Language - un sistema específico para gestión de bases de datos) Injection es uno de los ataques más populares y más peligrosos en internet. A diferencia del anterior, este ataque pretende explotar las vulnerabilidades de la base de datos de la aplicación web. De esta forma, si consigue acceder, el atacante puede robar información sensible tanto de los clientes como de la propia aplicación, así como modificar interesadamente información de la propia base de datos.

2.3.1.7. Infiltration

Dentro del ámbito de la seguridad informática, el término infiltración se utiliza para denominar el tipo de ataques en los cuales el atacante realiza una intrusión en una red privada y protegida, para una vez dentro realizar accesos indebidos a información o sistemas sensibles. En el caso de CICIDS2017, este ataque responde a un contexto muy específico, en el que se introduce un virus en una de las máquinas de la red, y desde ella se recoge información sobre los demás sistemas. El virus entra en el sistema descargándolo desde Dropbox en Windows o copiándose desde USB flash drive en Macintosh. Una vez el virus está en el sistema, el atacante puede explotar la vulnerabilidad abierta para escanear puertos dentro de la red.

2.3.1.8. Heartbleed

Heartbleed es una vulnerabilidad que presenta OpenSSL (una implementación open source de los protocolos SSL y TLS), concretamente en la función "heartbeat". El funcionamiento de heartbeat es el siguiente: el cliente realiza una solicitud al servidor con una carga de datos aleatoria, y el servidor responde al cliente con un paquete con la misma carga. Durante este ataque, realizado con la herramienta Heartleech, el atacante envía al servidor una solicitud heartbleed que anuncia tener una gran carga de datos pero realmente está vacía. Debido a la vulnerabilidad de OpenSSL, el servidor responde con una sección de memoria de la longitud especificada. Estos datos pueden contener información sensible como usuarios o contraseñas. El atacante puede realizar esta operación indefinidamente, pudiendo obtener una gran cantidad de datos.

2.3.2. UNSW-NB15

Como se ha visto en la Sección 2.3, el dataset UNSW-NB15 presenta 2.540.044 entradas de datos, de las que un 17% responden a tráfico malicioso. La naturaleza de cada entrada del tráfico viene dada por una etiqueta, que puede tomar 10 valores distintos. Uno de ellos (Benign) representa el tráfico normal, mientras que los otros 9 representan ataques, especificando el tipo. La distribución de ataques, tanto en términos globales como por tipo, puede verse en las Figuras 2.4. y 2.5.

Se pueden apreciar grandes diferencias en cuanto al número de entradas para unos tipos de ataque u otros. El ataque Generic, por ejemplo, supone más de la mitad del tráfico malicioso, mientras que de tipo Worms sólo hay 174 entradas. Esto se debe a la distinta naturaleza de cada ataque, hay algunos que por sus características generan mucho más tráfico que otros.

Con el objetivo de entender mejor los datos que estamos manejando, vamos a estudiar, uno por uno, todos los tipos de ataques presentes en el dataset UNSW-NB15. A diferencia de los vistos en CICIDS2017, en este caso no se trata de ataques específicos sino de grupos más amplios que engloban intrusiones de distinto tipo. A continuación

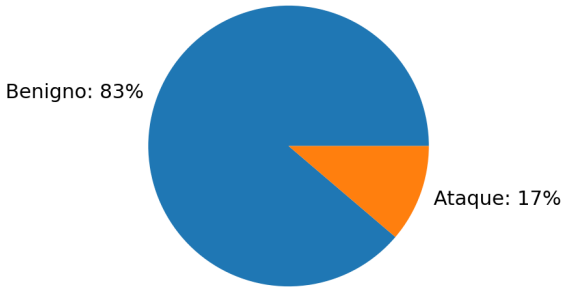

Figura 2.4: Proporción tráfico benigno/ataque (UNSW-NB15)

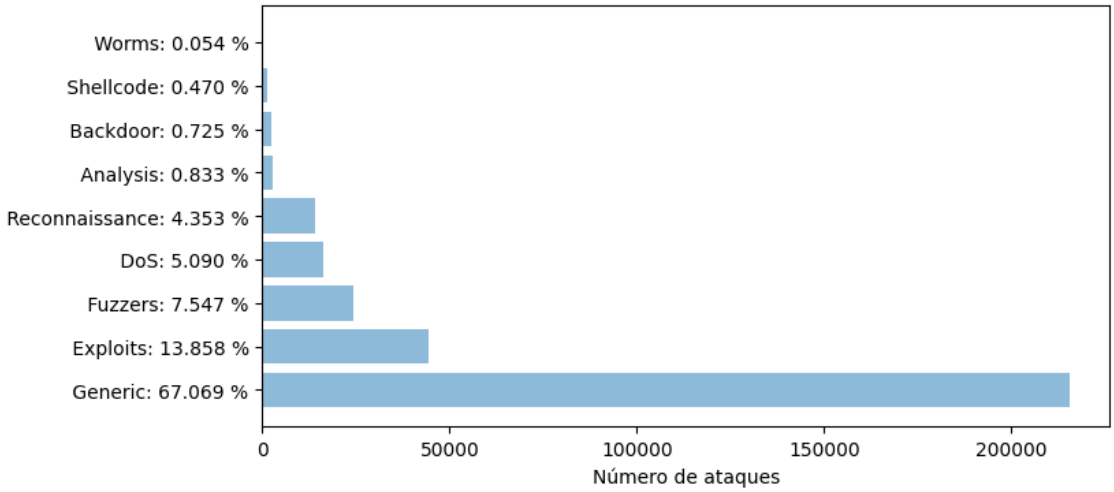

Figura 2.5: Distribución de los ataques (UNSW-NB15)

los estudiamos uno por uno.

2.3.2.1. Generic

Los ataques genéricos son aquellos que atentan contra un sistema criptográfico cualquiera sin tener en cuenta los detalles de la implementación de dicha primitiva criptográfica. Es similar a los ataques de fuerza bruta, de forma que conociendo el tamaño de bloque y la longitud de la clave, se realiza una gran cantidad de intentos para tumbar el sistema de cifrado. Por esta razón, este tipo de ataques acaba reflejando un volumen de tráfico mucho mayor que otros.

2.3.2.2. DoS

Los ataques de denegación de servicio (DoS) se han visto con detalle en el apartado 2.4.1.1, dentro del estudio de los ataques presentes en el dataset CICIDS2017.

2.3.2.3. Exploits

Este tipo de ataques engloba a todos aquellos que aprovechan una vulnerabilidad de la víctima para atacarla. El atacante descubre un punto débil en el sistema operativo o en alguna sección del software, lanza un ataque para abrir una brecha en el sistema y, una vez dentro puede llevar a cabo sus objetivos maliciosos. En esta última fase, es habitual que el atacante introduzca un malware en la víctima para robar información, escalar privilegios o tomar control total del sistema.

2.3.2.4. Fuzzers

Los ataques de tipo fuzzing, o fuzzers, se basan en proporcionar datos inválidos, inesperados o aleatorios a las interfaces de introducción de datos de víctima para llevar al sistema al límite y que éste colapse. Además, este tipo de ataques se pueden utilizar para descubrir nuevas vulnerabilidades en el sistema objetivo y después explotarlas con otro tipo de ataques. Puesto que los fuzzers envían una gran cantidad de datos, terminan generando un tráfico importante, como puede verse en la distribución de ataques en la Figura 2.3.

Cabe destacar que el fuzzing se utiliza también legalmente por parte de usuarios o compañías en sus propios sistemas con el objetivo de descubrir vulnerabilidades desconocidas y así protegerse ante los posibles ataques.

2.3.2.5. Reconnaissance

Este conjunto incluye aquellos ataques que tienen como objetivo recoger toda la información posible sobre la víctima antes de lanzar un ataque más agresivo. Los ataques de reconocimiento buscan también vulnerabilidades en el sistema objetivo para aprovecharlos más tarde en otros ataques.

2.3.2.6. Analysis

Este grupo de ataques, también conocidos como de reconocimiento activo, usa distintos métodos como el escaneo de puertos, penetración de archivos HTML, SPAM o scripts maliciosos contra una aplicación web. El objetivo de estos ataques es entrar en el sistema de la víctima a través de dicha aplicación web, normalmente utilizando un puerto que se encuentre abierto.

2.3.2.7. Backdoor

Las puertas traseras, o backdoor en inglés, son aquellas secciones de código presentes en un sistema software que permiten evitar los sistemas de seguridad implementados. Estos puntos débiles del sistema se incluyen para realizar operaciones de

mantenimiento o de depurado, pero suponen un riesgo si son descubiertos por un atacante, ya que pueden aprovecharse para entrar en el sistema sin permiso. Cuando un atacante descubre o crea él mismo una puerta trasera en el sistema, puede esquivar todos los sistemas de seguridad del mismo para acceder a él y llevar a cabo sus objetivos maliciosos. Este tipo de ataque presenta grandes dificultades a la hora de detectarlo ya que no genera demasiado tráfico, y en cualquier caso no suele resultar sospechoso.

2.3.2.8. Shellcode

El shellcode es un fragmento de código que se utiliza para ejecutar tareas maliciosas en el ordenador de la víctima. Dicho código suele introducirse en el equipo aprovechando vulnerabilidades en el mismo. Generalmente los shellcode están programados en lenguaje ensamblador y convertidos en opcodes (conjuntos de valores hexadecimales) que se inyectan en la pila de ejecución de un programa para conseguir que la máquina en la que reside ejecute la operación maliciosa. El propósito más habitual de estos ataques, y de ahí su nombre, es hacerse con una shell con la que después al atacante podrá ejecutar nuevos comandos.

2.3.2.9. Worms

Los gusanos, o worms en inglés, son un tipo de malware cuya principal característica es que se replica para propagarse a otras máquinas. Una vez infecta un computador, realiza una serie de ataques a las máquinas de su red buscando vulnerabilidades para después infectarlas también. El resultado, si no se contiene, suele ser la extensión de este malware por toda la red. En ocasiones, y a diferencia de los virus, los gusanos no realizan ninguna operación dañina de forma directa en las máquinas infectadas. Sin embargo, el mero hecho de su propagación por una red puede suponer grandes problemas debido a su crecimiento exponencial y el alto consumo de ancho de banda que presenta.

2.4. Aprendizaje automático

Una vez estudiados con detalle los datasets con los que trabajaremos, el siguiente paso es escoger los algoritmos de aprendizaje automático que se van a utilizar para implementar el sistema de detección de intrusos. En esta sección vamos a introducir conceptos importantes sobre el aprendizaje automático, para después estudiar una serie de algoritmos que más tarde utilizaremos en la implementación.

El aprendizaje automático es una de las múltiples disciplinas que existen dentro del campo de la inteligencia artificial, el cual permite a los modelos ”aprender” a partir de una una gran cantidad de datos. Este aprendizaje se lleva a cabo utilizando una serie de algoritmos, los cuales son capaces de reconocer patrones en el conjunto de datos de entrada (también llamado conjunto de entrenamiento), realimentándose y

modificando dinámicamente su propio comportamiento para afinar al máximo dicho reconocimiento. Una vez procesados los datos de entrenamiento, queda definido el modelo y se puede aplicar ahora a nuevos datos (conjunto de test) para comprobar su rendimiento a la hora de reconocer los patrones buscados.

Los algoritmos de aprendizaje automático son especialmente útiles en aquellos problemas donde los métodos clásicos no funcionan bien. Además, son capaces de intepretar eficazmente enormes cantidades de datos, pudiendo resolver rápidamente problemas complejos minimizando la intervención humana.

Dependiendo del grado de realimentación que presenten los algoritmos de aprendizaje automático, podemos distinguir cuatro grandes grupos: aprendizaje supervisado, no supervisado, semisupervisado y por refuerzo. Nos centraremos en los dos primeros ya que son los más útiles en el caso que nos ocupa. En el aprendizaje por refuerzo, el sistema recibe algún tipo de recompensa (positiva o negativa) cada vez que produce una respuesta, ajustando su comportamiento en función de dicha recompensa. Esta técnica se usa típicamente en aprendizaje robótico. Por su lado, el aprendizaje semisupervisado combina otros dos tipos de aprendizaje: utiliza un enfoque no supervisado para extraer características relevantes de los datos no etiquetados, para después usar estos conocimientos para optimizar el modelo supervisado.

A continuación introducimos los conceptos de aprendizaje supervisado y no supervisado, viendo en cada caso una serie de algoritmos que serán los que utilicemos en la fase de implementación del sistema de detección de anomalías. Todos estos conceptos pueden verse con gran detalle en el libro de Russell y Norvig [21] sobre inteligencia artificial. En cuanto a la implementación, para la que se va a utilizar la librería scikit-learn de Python, los detalles pueden encontrarse en el libro de Garreta y Moncecchi [5], y en la documentación de la propia librería [1].

2.4.1. Aprendizaje supervisado

Este tipo de algoritmos necesitan que los datos de entrada, además de contener sus variables predictoras, estén correctamente clasificados y etiquetados. El sistema procesará dichos datos buscando patrones y corrigiéndose a sí mismo gracias a al etiquetado del conjunto de entrenamiento. Una vez definido el modelo, se pone a prueba con el conjunto de test, comparando los resultados obtenidos por el algoritmo con las etiquetas de los propios datos. Dependiendo del tipo de variable de salida que se busque, podemos tener problemas de regresión (si es una variable numérica) o de clasificación (si es categórica). Puesto que nuestro objetivo es clasificar las muestras como normales/ataques, nos centraremos en el segundo tipo.

Los algoritmos de aprendizaje supervisado suelen dar muy buenos resultados, pero no siempre es fácil tener los datos etiquetados y clasificados de antemano. En el caso de un sistema de detección de intrusos, los datos han de estar clasificados como normales/maliciosos, y los algoritmos deberán aprender a predecir, a partir de los datos de entrenamiento, la naturaleza del tráfico nuevo.

Puesto que ambos datasets con los que vamos a trabajar tienen sus datos ya etiquetados, podemos utilizar algoritmos de aprendizaje supervisado. Veremos ahora una serie de algoritmos de este tipo, que serán los que utilicemos para implementar el sistema de detección de anomalías. A la hora de escogerlos, además de incluir los utilizados en el trabajo de Kostas [3], se han buscado aquellos que fuesen más populares y pudieran encajar mejor en el objetivo de este trabajo.

2.4.1.1. Gaussian Naïve Bayes (GNB)

Los algoritmos de Naïve Bayes, basados en el teorema de Bayes, utilizan técnicas probabilísticas para hacer predicciones sobre los elementos. Estos modelos parten de la base de que la presencia o ausencia de una determinada característica no está relacionada con la de cualquier otra variable. Es decir, que todas las características (variables predictoras) son independientes entre sí y cada una contribuye independientemente a la probabilidad de pertenecer a una clase. Además, en el caso del Gaussian Naïve Bayes, se entiende que los datos de entrada tienen una distribución normal, o gaussiana. Estas asunciones simplifican notablemente los cálculos, pero no siempre son correctas. Por esta razón, Naïve Bayes suele ofrecer un rendimiento menor que otros algoritmos de aprendizaje automático.

2.4.1.2. Multinomial Naïve Bayes (MNB)

Este algoritmo es otra versión del Naïve Bayes que acabamos de ver. De igual manera, asume "ingenuamente" que los parámetros de los datos son independientes, por lo que la presencia/ausencia de unos no influyen en la de otros. La particularidad del Naïve Bayes Multinomial es que en este caso asume en los datos una distribución multinomial. Con esa premisa, utiliza el teorema de Bayes y la probabilidad condicionada para realizar predicciones sobre los elementos de entrada.

2.4.1.3. Árboles de decisión

Los árboles de decisión son una familia de clasificadores muy comunes y eficaces que se basan en una serie de sencillas reglas. El modelo genera un árbol (que suele ser binario, aunque hay más posibilidades) de forma que cada nodo consiste en un punto de decisión (o bifurcación), la cual suele consistir en evaluar una variable concreta. El algoritmo toma un elemento partiendo de la raíz y, cuando llega a un nodo, dependiendo de la decisión tomada bifurca por la rama izquierda o por la derecha, repitiendo el proceso hasta llegar al final del árbol. Al alcanzar una hoja, el árbol de decisión determina de qué clase es el elemento en cuestión, normalmente con cierto margen de probabilidad. Existen distintas implementaciones de esta estrategia, y en nuestro caso utilizaremos el algoritmo ID3 (Iterative Dichotomiser 3).

2.4.1.4. Random Forest (RF)

Este método de aprendizaje automático se basa en la técnica de los árboles de decisión, generando N árboles individuales, cada uno a partir de un subconjunto aleatorio de los datos de entrenamiento. A la hora de realizar predicciones sobre nuevas observaciones, se combinan las predicciones de todos los árboles que conforman el modelo, escogiendo aquella que más veces aparezca. Este algoritmo permite combinar los resultados de múltiples árboles de decisión, que individualmente no son tan fiables, pero en conjunto forman un modelo muy robusto.

2.4.1.5. K Nearest Neighbours (KNN)

El algoritmo de los K vecinos más cercanos es uno de los más utilizados en el ámbito del aprendizaje automático. Su funcionamiento es simple, y se basa en la premisa de que los elementos de una muestra con propiedades similares estarán cercanos entre sí. De esta forma, KNN utiliza una determinada métrica para medir la distancia entre dos puntos basándose en los valores de sus parámetros. Para un nuevo elemento, se observan los K vecinos que estén más cerca tomando dicha métrica de distancia. De esta forma, el nuevo elemento se catalogará en la clase más repetida entre los K vecinos observados.

2.4.1.6. AdaBoost (AB)

AdaBoost (Adaptative Boosting) es un algoritmo de boosting, lo cual implica que potencia otros clasificadores débiles para dar lugar a un clasificador más robusto. El algoritmo inicialmente asigna un peso idéntico a los datos de partida, que serán un subconjunto aleatorio del total. En cada iteración, se clasifican los datos con un algoritmo débil, y se asignan pesos mayores a aquellos datos que no se han clasificado correctamente. Además, se asigna un peso total al algoritmo débil, proporcional al total de aciertos, de forma que tendrá una mayor influencia cuantos más aciertos consiga. En la siguiente iteración, se llama de nuevo al algoritmo débil y se vuelve a ajustar, esta vez, empleando los pesos actualizados en la iteración anterior. El nuevo algoritmo débil se guarda, obteniendo así un nuevo modelo para el conjunto total. Este proceso se repite M veces, generando un conjunto de M algoritmos débiles distintos.

Para clasificar nuevas observaciones, se obtiene la predicción de cada uno de los algoritmos débiles que forman el conjunto y se agregan sus resultados, ponderando el peso de cada uno acorde al peso que se le ha asignado en el ajuste. De esta forma, cada nuevo algoritmo débil se centra en predecir correctamente las observaciones que los anteriores no han sido capaces. AdaBoost ofrece muy buenos resultados a partir de algoritmos inicialmente muy débiles. En nuestro caso, en la implementación utilizará como algoritmo débil un árbol de decisión con una o pocas ramificaciones.

2.4.1.7. Gradient Boosting Classifier (GBC)

Gradient Boosting es una generalización del algoritmo AdaBoost, de forma que permite emplear cualquier función objetivo, siempre que esta sea diferenciable (en nuestro caso, una función que clasifique un elemento en función de sus parámetros). La flexibilidad de este algoritmo ha hecho posible aplicar esta optimización o boosting a multitud de problemas (regresión, clasificación múltiple...) convirtiéndolo en uno de los métodos de aprendizaje automático más populares. Aunque existen distintas adaptaciones, la idea general de todas ellas es la misma: entrenar modelos más débiles de forma secuencial, de forma que cada modelo ajusta los errores de los modelos anteriores. Habitualmente como modelo débil se utilizan árboles de clasificación con una o pocas ramificaciones, igual que en el algoritmo AdaBoost.

2.4.1.8. Multi-Layer Perceptron (MLP)

El perceptrón multicapa es un tipo de red neuronal artificial formada por múltiples capas, cada una de ellas formada por una serie de "neuronas", que suelen ser perceptrones simples. Cada neurona recibe una serie de entradas, evalúa una determinada función de acuerdo con unos pesos asignados a cada parámetro, y devuelve una salida. La salida de cada capa de la red neuronal es la entrada de la siguiente, así hasta llegar a la última capa, la cual devuelve una predicción. Al finalizar este proceso (también llamado "epoch"), se comprueba la predicción esperada con la devuelta por el MLP, y si ha fallado, modifica convenientemente los pesos asignados a cada parámetro en cada neurona, de forma que el modelo aprende de sus propios errores.

Las redes neuronales son de los sistemas de aprendizaje automático que mejores resultados suelen ofrecer, siendo capaces de resolver problemas realmente complejos aumentando la profundidad de la red. Sin embargo, tienen el inconveniente de ser difícilmente interpretables, funcionando en muchas ocasiones como una caja negra.

2.4.1.9. Quadratic Discriminant Analysis (QDA)

El análisis discriminante cuadrático es un clasificador estadístico que se basa a su vez en el análisis discriminante lineal (LDA). QDA asume que los datos de entrada tienen una distribución normal, pero que las observaciones de cada clase no tienen por qué tener la misma matriz de covarianza. Con esas premisas, para clasificar un elemento evalúa en él y para cada clase cierta función cuadrática, que devuelve un valor u otro dependiendo de la clase tomada. De esta forma, el modelo QDA clasificará al elemento en la clase que mejor resultado haya dado al evaluar dicha función cuadrática.

2.4.1.10. Logistic Regression (LR)

La regresión logística es un algoritmo de clasificación estadístico que analiza la relación entre las distintas variables para clasificar los elementos. Para la predicción se

utiliza la función logística binaria, la cual mapea cada elemento de la entrada con un valor entre 0 y 1. Este valor responderá a la probabilidad de pertenecer a una clase o a otra. Dependiendo de si la probabilidad supera o no cierto valor umbral, el modelo clasificará al elemento en un determinado grupo o en otro. Este algoritmo, junto con el de regresión lineal, aunque es uno de los clasificadores más simples, en muchas ocasiones ofrece resultados similares a los de modelos notablemente más complejos. Esta relación entre bajo coste y buen desempeño lo convierten en un algoritmo de clasificación más que razonable.

2.4.1.11. Support Vector Machine (SVM)

Las máquinas de vector soporte son un conjunto de algoritmos de aprendizaje supervisado que pueden utilizarse tanto en problemas de clasificación como de regresión. Dado un conjunto de muestras, una SVM construye uno o más hiperplanos en un espacio cuyas dimensiones son los parámetros de estudio. Estos hiperplanos dividen el espacio muestral en regiones que corresponden con las distintas clases en las que se clasifican los elementos. Cuando se quiere clasificar una nueva muestra, se sumerge en el espacio multidimensional y se observa en qué lado de los respectivos hiperplanos se encuentra, determinando de esta forma a qué clase corresponde el elemento.

2.4.1.12. Stochastic Gradient Descent (SGD)

El descenso de gradiente estocástico es un método de de aprendizaje automático que se basa en la optimización de una función objetivo concreta. Dicha función, también llamada función de coste, depende de una serie de parámetros que inicialmente se establecen de forma aleatoria. El entrenamiento del modelo consiste en tomar un subconunto aleatorio de datos, y para cada elemento calcular el gradiente de la función de coste en dicho punto. Después de cada evaluación, se actualizan los parámetros del modelo en la dirección negativa del gradiente en una pequeña cantidad, conocida como la tasa de aprendizaje. Iterando este proceso una serie de epochs, los parámetros finales determinan el hiperplano óptimo que separa los elementos de las distintas clases. De esta forma, a la hora de clasificar una nueva muestra, basta con ver en qué región se encuentra el punto y obtenemos así la clase predicha.

2.4.2. Aprenzidaje no supervisado

También llamado aprendizaje por descubrimiento, este tipo de algoritmos no necesitan que los datos de entrada estén clasificados de antemano. En este caso el algoritmo agrupa los elementos de acuerdo con sus parámetros, sin atender a ninguna categoría prefijada, buscando cierta estructura en los datos. El resultado final consiste en una división de los datos de partida en grupos o clusters, y es labor del supervisor interpretar a qué responde cada uno de los grupos. Una vez creado el modelo, cuando reciba un dato nuevo lo clasificará calculando en qué cluster encaja mejor teniendo en cuenta sus parámetros.

Este tipo de algoritmos son mucho menos costosos que los de aprendizaje supervisado ya que no requieren que los datos de entrada estén categorizados. Por tanto, en escenarios reales, son este tipo de algoritmos los que más se suelen utilizar ya que no es habitual tener grandes cantidades de datos perfectamente etiquetados. Sin embargo, el rendimiento de estos clasificadores suele ser muy inferior al de los algoritmos supervisados.

Aunque ambos datasets con los que vamos a trabajar tienen sus datos ya etiquetados, podemos utilizar con ellos algoritmos de aprendizaje no supervisado. Bastará ignorar o eliminar la etiqueta que clasifique cada entrada y tratar los datos como si no estuvieran clasificados. Veremos ahora una serie de algoritmos de este tipo, que serán los que utilicemos para implementar el sistema de detección de anomalías.

2.4.2.1. K-Means

El algoritmo de K-Medias, o K-Means, es un método de clasificación no supervisado que agrupa los elementos de una muestra en K grupos o clusters teniendo en cuenta sus características. Dicho agrupamiento se realiza miniminzando la suma de distancias entre cada elemento y el centroide de su grupo o cluster. El centroide de un conjunto es la posición promedio de los puntos de dicho conjunto. Inicialmente se escogen los k centroides aleatoriamente. En cada iteración, se asigna cada elemento de los datos a su centroide más cercano, y a continuación se actualizan los centroides de acuerdo con los elementos que tiene asignados cada uno. Este proceso se repite hasta que los centroides no se mueven, o lo hacen en una cantidad menor que cierto umbral. Este algoritmo es uno de los más populares dentro del aprendizaje no supervisado ya que es un método sencillo y rápido. La principal dificultad que presenta es elegir el valor óptimo para K, pero en nuestro caso tomaremos K=2, ya que queremos agrupar las entradas de tráfico en dos clusters: normales/maliciosas.

2.4.2.2. K-Medoids

K-Medoids es un algoritmo de clustering estrechamente relacionado con el método de K-Means. De igual manera, su objetivo es crear una partición del conjunto muestral en clusters, minimizando la distancia entre los puntos de cada conjunto y el centro de este. A diferencia del algoritmo de K-Means, en K-Medoids cada cluster está representado por uno de los puntos que lo forman. Este punto, conocido como medoide, es el que minimiza la distancia promedio con respecto a los demás miembros del cluster. K-Medoids supone una alternativa robusta al clustering de K-Means, siendo menos sensible al ruido y a los valores atípicos. De nuevo, uno de los mayores problemas suele ser determinar el valor óptimo para K, pero en nuestro caso tomaremos K=2, ya que queremos agrupar las entradas de tráfico en dos clusters: normales/maliciosas.

2.4.2.3. Isolation Forest (IF)

El algoritmo Isolation Forest es un método no supervisado para detección de anomalías, que se basa en ideas similares a las del algoritmo Random Forest. En este caso, un Isolation Forest está formado por un conjunto de árboles llamados Isolation Trees, cada uno de los cuales se forma a partir de un subconjunto aleatorio del conjunto de entrenamiento. Para construir cada árbol, se escogen sucesivamente parámetros aleatorios de la muestra para ir dividiendo los datos hasta que todas las observaciones queden aisladas individualmente. Una vez construidos todos los árboles, a la hora de clasificar una observación nueva, se tiene en cuenta el promedio de divisiones que se han necesitado para aislar dicha observación. Cuanto menor sea este valor, mayor será la probabilidad de ser una anomalía. Este algoritmo tiene un orden de complejidad lineal y bajo consumo de memoria, por lo que funciona realmente bien con grandes volúmenes de datos.

2.4.2.4. Local Outlier Factor (LOF)

El algoritmo de Valor Atípico Local es un algoritmo de detección de anomalías no supervisado que se basa en la desviación de la densidad local de un punto con respecto a sus vecinos. La densidad para un elemento se calcula localmente midiendo la distancia con respecto a sus k vecinos más cercanos. Al comparar la densidad local de un punto con la de sus vecinos, el modelo LOF es capaz de reconocer regiones con similar densidad, y elementos con densidad sensiblemente inferior que la de sus vecinos, los cuales se consideran valores anómalos.

2.5. Trabajo relacionado

En este apartado comentaremos brevemente algunos estudios centrados en el uso de aprendizaje automático para la detección de anomalías en redes. Se irán viendo en orden cronológico, incluyendo en cada caso los algoritmos y datasets usados, así como los resultados que hayan arrojado. Los estudios que aquí se muestran se han escogido atendiendo al uso de diferentes algoritmos de aprendizaje y datasets, y que además pudieran ofrecer resultados interesantes en este ámbito. Conocer todos estos estudios relacionados resultará fundamental a la hora de desarrollar el presente trabajo, motivando la progresión de un conocimiento más integral sobre la detección de anomalías en redes.

En un estudio realizado en 2017 por Sharafaldin et al. [22], se utilizaron siete conocidos algoritmos de aprendizaje automático supervisado (GNB, RF, KNN, MLP, AdaBoost, ID3 y QDA) para detectar 15 tipos diferentes de ataque. Para ello se usó el dataset CICIDS2017, obteniendo los siguientes resultados de rendimiento a la hora de detectar los ataques: GNB:0.84, KNN:0.96, RF:0.97, MLP:0.76, AdaBoost:0.77, ID3:0.98, QDA:0.92 (tomando la medida F-Score como métrica).

También en 2017, y siguiendo el camino marcado por el trabajo de Sharafaldin et al. [22], se llevó a cabo el estudio de Kostas et al. [3], el cual utilizó también el dataset CICIDS2017 para profundizar en la detección de anomalías. Para ello se utilizaron los mismos siete algoritmos de aprendizaje automático que en su trabajo predecesor (GNB, RF, KNN, MLP, Adaboost, ID3 y QDA), y se obtuvieron los siguientes resultados de rendimiento: GNB:0.86, KNN:0.97, RF:0.94, MLP:0.83, Adaboost:0.94, ID3:0.95, QDA:0.86 (tomando la medida F1 como métrica). Puede verse, por tanto, que para algunos de los algoritmos los resultados obtenidos en este caso fueron mejorados.

En otro estudio, realizado en 2018 por Idhammad et al. [8], se utilizaron técnicas de aprendizaje semisupervisado y no supervisado sobre varios datasets, concretamente UNSW-NB15, UNBISCX-12 y NSL-KDD. El objetivo era detectar las anomalías de tipo DDoS sin utilizar los datos etiquetados (aprendizaje no supervisado), y se obtuvieron resultados de rendimiento de 94 %, 99 % y 98 %, respectivamente.

Por otro lado, en 2019 se llevó a cabo un estudio por Amangele et al. [2] para detectar anomalías en una red de dispositivos IoT (Internet de las Cosas). Para ello se exploraron distintos algoritmos de clustering jerárquico sobre el dataset CICIDS2017, obteniendo unos resultados globales de rendimiento de un 99 %.

También en 2019, Nawir et al. [13] realizaron un estudio sobre la aplicación de métodos de aprendizaje automático para la detección de anomalías en redes. En este caso los 5 modelos usados eran exclusivamente supervisados, concretamente GNB, Averaged Onde Dependence Estimator (AODE), Radial Basis Function Network (RBFN), MLP y árboles J48. El dataset utilizado fue el UNSW-NB15, y los resultados ofrecieron una tasa de acierto de 97 %, 76 %, 84 %, 89 % y 98 %, respectivamente.

Por último, es importante mencionar el estudio de Pérez et al. [19], realizado en 2020 y que ahondó en la detección de anomalías desde una perspectiva no supervisada. Este estudio resulta especialmente útil ya que, aunque tan sólo se usan 5 algoritmos de clasificación (Local Outlier Factor, Isolation Forest, OC-SVM, Mahalanobis y HBOS), realiza las pruebas sobre 4 datasets diferentes, que son UNSW-NB15, NSL-KDD, CICIDS2017 y KYOTO. En el caso de UNSW-NB15, los resultados de rendimiento fueron de 0.84, 0.69, 0.73, 0.81 y 0.71 (F1-score), respectivamente para cada algoritmo. Para el dataset CICIDS2017, los resultados fueron de 0.84, 0.80, 0.74, 0.82 y 0.75.

Como se puede ver, en los últimos años se han realizado numerosos estudios alrededor de la detección de anomalías en redes, y en varios de ellos se han utilizado los datasets CICIDS2017 y UNSW-NB15. Esto demuestra que, si bien estos no son los datasets más recientes, siguen siendo realmente útiles a día de hoy y se siguen usando en estudios con el mismo propósito.

Tal y como hemos adelantado en secciones anteriores, en este trabajo se seguirá el camino marcado por el estudio de Kostas et al. [3], con la idea de ampliarlo muy notablemente para conseguir resultados más amplios. El trabajo de Kostas [3] resulta realmente interesante por lo completo que es, y nuestro objetivo es ampliarlo todavía más.

Para ello se combinarán ideas de otros estudios mencionados, atacando el problema desde distintos enfoques, buscando obtener una aproximación más integral a la detección de anomalías.

De esta manera, en este trabajo se utilizarán dos datasets (CICIDS2017 y UNSW-NB15) en vez de uno solo, y se ampliará notablemente la batería de algoritmos utilizados, añadiendo 5 nuevos modelos supervisados a los 7 existentes y 4 más no supervisados. Estos cambios pueden verse de forma esquemática en la tabla 2.2. El objetivo de todos estos añadidos y ampliaciones es obtener unos resultados más completos, que abarquen más datos, más variedad de ataques y que utilicen más y más variados algoritmos de aprendizaje automático, tanto desde un punto de vista supervisado como no. En el Capítulo 4, en el que se analizarán los resultados obtenidos, estos se compararán con los de los trabajos de Kostas et al.[3] y Sharafaldin et al.[22] (que fue modelo del primero) para tener así una mejor perspectiva de la profundización en el conocimiento sobre la detección de anomalías.

Estudios	**Datasets (Nº de datos)**	**Tipos de ataques**	**Algoritmos supervisados**	**Algoritmos no supervisados**
Sharafaldin et al.	CICIDS2017 (2.83 millones)	Fuerza bruta FTP, Heartbleed, Fuerza bruta, SSH, DDoS, infiltration, DoS, ataque web y botnet	GNB, RF, KNN, MLP, Adaboost, ID3, QDA	-
Kostas et al.	CICIDS2017 (2.83 millones)	Fuerza bruta FTP, Heartbleed, Fuerza bruta, SSH, DDoS, infiltration, DoS, ataque web y botnet	GNB, RF, KNN, MLP, Adaboost, ID3, QDA	-
Este trabajo	CICIDS2017 (2.83 millones) UNSW-NB15 (2.54 millones)	Fuerza bruta FTP, Heartbleed, Fuerza bruta SSH, DDoS, infiltration, DoS, ataque web, botnet, fuzzers, Shellcode, Analysis, Worms, Backdoors, Reconnaissance, Generic y Exploits	GNB, RF, KNN, MLP, Adaboost, ID3, QDA, LR, SVM, MNB, SGD, GBC	K-Means, K-Medoids, IF, LOF

Tabla 2.2: Comparación de los estudios

Capítulo 3

Metodología

Una vez visto el estado de la cuestión y estudiados todos los conceptos previos necesarios, en este tercer capítulo se explica el trabajo que se ha llevado a cabo para implementar los sistemas de detección de anomalías. Primeramente se muestran las herramientas y métodos utilizados en el proceso, para después explicar detalladamente todo el proceso de implementación como tal.

3.1. Herramientas y métodos

En esta sección se ven las herramientas y métodos utilizados en el proceso de implementación. En primer lugar se detallan los recursos y plataformas software utilizadas tanto para la elaboración del código como para la ejecución. Después se ven los recursos hardware utilizados en el proceso, y por último se explican las métricas de rendimiento que se van a recoger durante la ejecución de las pruebas. Estas magnitudes tendrán gran importancia en el siguiente capítulo, en el que se analizarán los resultados obtenidos.

3.1.1. Recursos Software

Python: Es un lenguaje de programación interpretado y multiparadigma, operativo bajo licencia de código abierto. Su sintaxis simple y su estructura dinámica lo han convertido en uno de los lenguajes más populares en la actualidad. Alrededor de Python existe una extensísima documentación, además de una gran cantidad de librerías de todo tipo, destacando aquellas relacionadas con el aprendizaje automático. Por estas razones, se ha decidido implementar todo el código con Python 3.11.

Jupyter Notebook: Es un entorno de desarrollo de código que permite implementar scripts en diferentes lenguajes de programación, entre ellos, Python. Una de las principales ventajas que presenta es que los fragmentos de código pueden separarse en bloques ejecutables de forma independiente, manteniendo un entorno de variables común. Además, estos bloques de código pueden intercalarse con otros de tex-

to enriquecido con Markdown, permitiendo incluir explicaciones detalladas de cada fragmento de código. En este caso, todos los scripts de Python que se necesiten se implementarán con Jupyter Notebook.

Google Colab: (Google Colaboratory) es un servicio de Google que permite la ejecución en remoto de código en Jupyter Notebook, que no requiere configuración y que ofrece acceso gratuito a recursos de computación, como GPUs y TPUs. Colab es una solución especialmente adecuada para el aprendizaje automático y la ciencia de datos, y en este trabajo se ha utilizado para realizar algunas pruebas y ejecuciones del código implementado, cuando por alguna razón la máquina física usada no resultaba suficiente.

Sklearn: (Scikit-learn) es una librería de aprendizaje automático que puede utilizarse con el lenguaje de programación Python. Sklearn ofrece una amplia variedad de opciones al usuario con una gran cantidad de algoritmos de aprendizaje automático. Esta librería dispone además de una extensa y detallada documentación, que incluye todos los algoritmos y procedimientos usados en este trabajo.

Pandas: Es una potente librería de análisis de datos disponible en Python. A la hora de trabajar con grandes cantidades de datos, Pandas permite realizar fácilmente todo tipo de operaciones sobre los datos. De esta forma, dada su facilidad de uso y su versatilidad, Pandas se ha convertido en una de las librerías imprescindibles en cualquier análisis de datos.

Matplotlib: Es una librería disponible en Python que permite visualizar los datos en forma de gráficos de todo tipo. Matplotlib se utilizará en este trabajo para obtener todas las gráficas que aparecen.

NumPy: Es una librería de Python que permite realizar operaciones matemáticas y lógicas de forma rápida y fácil. En este trabajo se ha dado uso a las funciones y métodos propios de NumPy para realizar todos los cálculos necesarios dentro del código desarrollado.

3.1.2. Recursos Hardware

A la hora de implementar los modelos y realizar las pruebas, una métrica que se suele tener en cuenta es el tiempo de ejecución. Y esta magnitud viene condicionada en buena medida por los recursos hardware de que disponga la máquina en la que se realizan los experimentos. Por tanto, para que se puedan realizar comparaciones con pasados y futuros trabajos similares, a continuación se indican las características del ordenador con las que se han realizado las pruebas:

Procesador:	Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
Memoria RAM:	8,00 GB
Sistema Operativo:	Windows 11 Home 64-bit
Tarjeta Gráfica:	NVIDIA GeForce MX150

3.1.3. Métricas de evaluación de rendimiento

Una vez construidos los modelos con los distintos algoritmos a partir del conjunto de datos de entrenamiento, los sistemas se pondrán a prueba con los datos de test. Para medir el rendimiento de cada algoritmo, en cada ejecución se recogerán cuatro métricas distintas, concretamente Accuracy, Precision, Recall y F1-score. Antes de explicar cada una de ellas, vamos a fijar algunos conceptos importantes que tienen que ver con los resultados obtenidos en la clasificación.

Cuando se obtiene la clasificación para los datos de test, dependiendo de cuáles hayan sido las predicciones (benigno/ataque) para las muestras y cuáles fuesen sus verdaderas etiquetas, podemos agrupar los resultados obtenidos en los siguientes grupos:

- Verdaderos positivos (VP): aquellos datos que forman parte de un ataque y han sido clasificados como tal (detección correcta).
- Falsos positivos (FP): aquellos datos que son de tráfico normal pero han sido clasificados como ataques (detección incorrecta).
- Verdaderos negativos (VN): aquellos datos que son de tráfico normal y que han sido clasificados como tal (descarte correcto).
- Falsos negativos (FN): aquellos datos que forman parte de un ataque pero han sido clasificados como benignos (descarte incorrecto).

En la Figura 3.1 podemos visualizar las ideas anteriores en la siguiente representación, conocida como matriz de confusión:

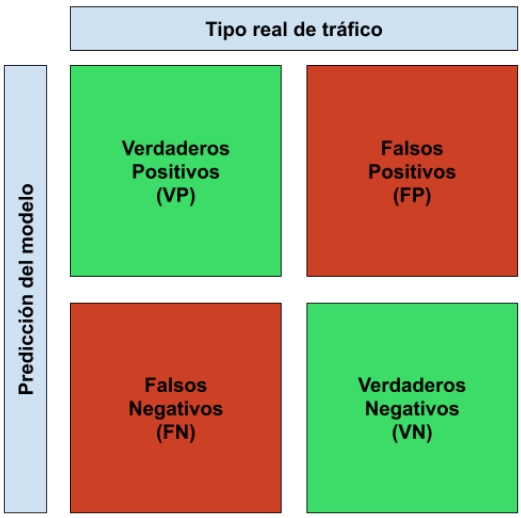

Figura 3.1: Matriz de confusión

Cuando se obtienen los resultados pronosticados por el modelo, estos se han de comparar con la clasificación real del conjunto de datos para comprobar así el rendi-

miento del sistema. Existen distintas métricas para el error y el rendimiento en problemas de clasificación. En nuestro caso, utilizaremos las siguientes:

Accuracy (Exactitud): Tasa de datos correctamente categorizados con respecto a los datos totales.

$$Accuracy = \frac{VN + VP}{FP + VN + VP + FN}$$

Precision (Precisión): También conocida como Valor Predictivo Positivo (VPP), es la tasa de datos correctamente clasificados como ataques con respecto al total de ataques detectados.

$$Precision = \frac{VP}{FP + VP}$$

Recall (Exhaustividad): También conocida como Tasa de Veredaderos Positivos (TVP), es la tasa de datos correctamente clasificados como ataques con respecto al total de ataques reales.

$$Recall = \frac{VP}{VP + FN}$$

F1-Score: Media armónica de precisión y exhaustividad. Esta métrica es, de entre todas, la más adecuada ya que es más sensible a que uno de los dos valores sea muy pequeño.

$$F1 - Score = 2 \cdot \frac{VPP \cdot TVP}{VPP + TVP} = \frac{2 \cdot VP}{2 \cdot VP + FP + FN}$$

Las métricas anteriores en un principio están pensadas para medir el rendimiento de un clasificador supervisado, ya que para calcularlas hay que conocer, además de la predicción del modelo, la clasificación real de cada elemento. Estos últimos datos normalmente no suelen estar disponibles si se están usando algoritmos no supervisados, precisamente por la naturaleza de estos algoritmos. Por esta razón, al trabajar con clasificadores no supervisados, se suelen usar otras métricas de rendimiento basadas en la densidad de los clusters encontrados. Sin embargo en nuestro caso sí que disponemos de todos los datos previamente categorizados, por lo que se podrán calcular, también para los algoritmos no supevisados, las métricas de rendimiento anteriormente definidas. Esto, pese a no ser lo habitual, es enormemente útil ya que nos permite comparar al mismo tiempo y con las mismas medidas el rendimiento de todos los algoritmos utilizados, independientemente de su tipo de supervisión.

Aunque no sea una métrica de rendimiento como tal, durante las ejecuciones también se recogerá el tiempo de cómputo de cada prueba. Esta medida puede variar notablemente dependiendo del dispositivo en el que se realicen las pruebas, pero puede servir para estimar el tiempo de cómputo en otras máquinas.

3.2. Implementación

En este apartado veremos todos los detalles del trabajo realizado sobre los datos y a la hora de implementar los sistemas de detección de anomalías con cada algoritmo. En

primer lugar se explica el análisis y preprocesamiento de los datos, etapa fundamental antes de utilizarlos en algoritmos de aprendizaje automático. Una vez preparados los datos se explicará el proceso de selección de parámetros. Esto se hace para optimizar el modelo, tomando sólo aquellos parámetros que resulten más importantes y reduciendo así la dimensionalidad del problema. Por último se verá con todo detalle la implementación de los sistemas con todos los algoritmos considerados y desde distintos enfoques. Cada una de estas etapas se realizará separadamente con cada dataset, de forma que el resultado final sean modelos análogos pero implementados sobre conjuntos de datos diferentes

Tanto la implementación como el futuro análisis de los resultados se realizarán desde dos enfoques. Por un lado, se crearán los modelos de detección de anomalías con el conjunto total de los datos, de forma que la clasificación se limite a detectar tráfico benigno o maligno. Por otro lado, se separarán los datos en función del tipo de ataque presente para realizar el mismo estudio separadamente con cada tipo de anomalía. De esta forma se obtendrán datos del rendimiento de los clasificadores a la hora de detectar ataques en general (sin distinguir el tipo), y también a la hora de detectar cada tipo de ataque por separado.

Los detalles acerca de la implementación en Python de todo este proceso para ambos datasets pueden verse el Apéndice E, que incluye el archivo README sobre la implementación.

3.2.1. Preprocesamiento de los datos

Tal y como se encuentran los datos públicamente, presentan algunos errores que se deben corregir antes de ser empleados con los algoritmos de aprendizaje automático. En esta sección explicamos el preprocesamiento de los datos para ambos datasets.

3.2.1.1. CICIDS2017

El dataset CICIDS2017 cuenta con 2.830.743 registros de tráfico, divididos en 8 archivos CSV distintos. El primer paso es juntar todos los datos en un solo archivo CSV, con el cual haremos el estudio de los registros presentes. En la Tabla 3.1 se puede ver la distribución de los registros en este dataset. Concretamente, vemos que hay 288.602 registros defectuosos, lo cual puede indicar que los datos son incorrectos o que están incompletos. Dichos registros se eliminarán para no distorsionar el aprendizaje de los algoritmos más adelante.

Los registros de tráfico vienen caracterizados por 86 columnas, cada una para un parámetro como son Flow ID, Source IP, Source Port, etc (en el Apéndice A puede verse la lista completa de estos parámetros con el detalle de cada uno). Sin embargo, el parámetro Fwd Header Length aparece duplicado, una vez en la columna 41 y otra en la 62. Corregimos este error eliminando la columna 62.

En los datos se recogen algunas características del tráfico en forma de parámetros

Tipo	Número de registros
Benigno	2359289
Defectuoso	288602
DoS Hulk	231073
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – SQL Injection	21
Heartbleed	11

Tabla 3.1: Distribución de los registros en el dataset CICIDS2017

de tipo string o categóricos. Puesto que los algoritmos de aprendizaje automático no pueden trabajar con este tipo de variables, las transformamos en numéricas usando la función LabelEncoder() de la librería Sklearn. Sin embargo, la etiqueta "Label", que recoge el carácter benigno/maligno del tráfico y el tipo de ataque en su caso, no se cambia ya que queremos mantenerla para realizar la clasificación por ataques que haremos más adelante.

Por último, se han detectado otros pequeños errores en los datos, como la presencia del carácter "–"(Unicode Decimal Code &#8211), el cual no lo reconoce la librería Pandas, por lo que se ha sustituido por el carácter "–"(Unicode Decimal Code &#45). Por otro lado, algunos parámetros, como Flow Bytes/s o Flow Packets/s, presentaban en algunos registros valores como “Infinity” o “NaN”, los cuales fueron sustituidos por -1 y 0, respectivamente.

Una vez corregidos los errores en los datos, vamos a crear una serie de archivos CSV, uno por cada tipo de ataque, de forma que en cada uno se recoja exclusivamente tráfico maligno de ese tipo en una proporción de 70 % benigno 30 % maligno. Para los ataques de tipo Web Attack (Web Attack–Brute Force, Web Attack–XSS, Web Attack–SQL Injection) se creará un único CSV recogiendo los tres subtipos de ataque bajo una misma etiqueta. Estos nuevos datasets se utilizarán más adelante para entrenar y evaluar los algoritmos de clasificación, de forma que además de obtener resultados para el conjunto total de los datos, sacaremos conclusiones sobre el rendimiento de la clasificación para cada tipo de ataque.

3.2.1.2. UNSW-NB15

El dataset UNSW-NB15 cuenta con 2.540.044 registros de tráfico, divididos en 4 archivos CSV distintos. El primer paso es juntar todos los datos en un solo archivo CSV, con el cual haremos el estudio de los registros presentes. En la tabla 3.2 se puede ver la distribución de los registros en este dataset. Concretamente, vemos que hay 2.218.760 registros benignos y uno solo defectuoso. Dicho registro se eliminará para no distorsionar el aprendizaje de los algoritmos más adelante.

Tipo	Número de registros
Benigno	2218760
Generic	215481
Exploits	44525
Fuzzers	24246
DoS	16353
Reconnaissance	13987
Analysis	2677
Backdoor	2329
Shellcode	1511
Worms	174
Defectuoso	1

Tabla 3.2: Distribución de los registros en el dataset UNSW-NB15

Los registros de tráfico vienen caracterizados por 49 columnas, cada una para un parámetro del flujo recogido, cuyos nombres pueden verse directamente en el fichero UNSW-NB15_features.csv (en el Apéndice A se incluye la lista completa de estos parámetros con el detalle de cada uno). De entre todos los parámetros, 9 de ellos responden a variables categóricas o de tipo string. Puesto que los algoritmos de aprendizaje automático no pueden trabajar con este tipo de variables, las transformamos en numéricas usando la función LabelEncoder() de la librería Sklearn. Sin embargo, las etiquetas "Label", que recoge el carácter benigno/maligno del tráfico, y "attack_cat", que muestra el tipo de ataque en su caso, no se transformarán en variables numéricas, sino que se fusionarán en una sola variable categórica al mismo modo que en el dataset CICIDS2017. De esta forma la columna "Label" recoge el carácter benigno/maligno del tráfico y el tipo de ataque en su caso.

Precisamente alrededor de la clasificación de cada ataque se han tenido que hacer algunas correcciones, ya que, por ejemplo, había etiquetas referentes al mismo ataque escritas de formas distintas ('Shellcode' y ' Shellcode '). Se observaron también errores en la catalogación de tráfico benigno, que en ocasiones tomaba el valor 'nan', por lo que se sustituyó en esos casos por la etiqueta 'BENIGN', igual que en el anterior dataset.

Al terminar todos las modificaciones anteriores, la organización de los datos es completamente igual a la presentada en el dataset CICIDS2017. De esta forma, el trabajo venidero podrá hacerse de forma completamente análoga.

De igual manera que con el primer dataset, una vez corregidos los errores en los

datos vamos a crear una serie de archivos CSV, uno por cada tipo de ataque, de forma que en cada uno se recoja exclusivamente tráfico maligno de ese tipo en una proporción de 70 % benigno y 30 % maligno. Estos nuevos datasets se utilizarán más adelante para entrenar y evaluar los algoritmos de clasificación, de forma que además de obtener resultados para el conjunto total de los datos, sacaremos conclusiones sobre el rendimiento de la clasificación para cada tipo de ataque.

3.2.2. Selección de parámetros

Como hemos visto, en ambos datasets los registros de tráfico vienen caracterizados por un gran número de parámetros (los cuales pueden verse con detalle en el Apéndice A). Los algoritmos de aprendizaje automático utilizan estos parámetros para entrenar los modelos y poder así clasificar elementos nuevos a partir de las mismas características. Sin embargo, no resultaría eficiente que los modelos utilizaran todos y cada uno de los parámetros del tráfico, ya que en la mayoría de casos la clasificación puede determinarse a partir de un reducido subgrupo de ellos.

De esta forma, parece conveniente seleccionar aquellos parámetros que resulten más relevantes en la clasificación, y utilizarlos para entrenar los modelos de detección de anomalías. Como hemos adelantado, el estudio se realizará desde dos aproximaciones: una para el conjunto total de datos teniendo en cuenta sólo la clasificación en registros normales o maliciosos; y otra para cada tipo de ataque, de forma que la detección se haga sobre subconjuntos de datos que sólo presenten ataques de un tipo determinado. De esta forma, para desarrollar los modelos necesitaremos seleccionar los parámetros más importantes tanto para el conjunto de datos total como para los subconjuntos de cada tipo de ataque. Este proceso lo realizaremos para ambos datasets de forma completamente análoga, para poder así después seguir con una implementación paralela. A continuación explicamos el proceso seguido en cada caso y los resultados obtenidos.

3.2.2.1. CICIDS2017

En primer lugar realizaremos la selección de parámetros por importancia en los subconjuntos de datos para cada tipo de ataque. Los archivos CSV creados en la fase anterior presentan, para cada tipo de anomalía, un 30 % de tráfico malicioso y un 70 % de tráfico normal (seleccionado aleatoriamente).

Para medir la importancia de cada parámetro se utilizará el algoritmo Random Forest Regressor, de la librería Sklearn. Este algoritmo crea un “bosque” de árboles de decisión, de forma que a cada parámetro se le asigna un valor de importancia según lo útil que resulta en la construcción del árbol de decisión. Cuando finaliza el proceso, estos valores de importancia se comparan y ordenan. La suma de los valores de importancia de todos los parámetros da el valor de importancia del árbol de decisión. Comparando la importancia de cualquier parámetro con la del árbol completo nos da información sobre la importacia de dicho parámetro en el árbol de decisión.

Ataque / Parámetros	Importancia		Ataque / Parámetros	Importancia
Bot			**FTP-Patator**	
Bwd Packet Length Mean	0.360206		Fwd Packet Length Max	0.255835
Flow Duration	0.011118		Fwd Packet Length Std	0.027022
Flow IAT Max	0.010515		Fwd Packet Length Mean	0.010657
Flow IAT Mean	0.008170		Bwd Packet Length Mean	0.000637
DDoS			**Heartbleed**	
Bwd Packet Length Mean	0.470069		Bwd Packet Length Mean	0.052
Total Backward Packets	0.093654		Total Backward Packets	0.048
Fwd IAT Total	0.013253		Total Length of Bwd Packets	0.048
Total Length of Fwd Packets	0.006444		Bwd Packet Length Max Mean	0.040
DoS Goldeneye			**Infiltration**	
Flow IAT Max	0.474878		Fwd Packet Length Mean	0.224772
Bwd Packet Length Std	0.058527		Total Length of Fwd Packets	0.062421
Total Backward Packets	0.046780		Fwd Packet Length Max	0.028231
Flow IAT Min	0.043043		Flow Duration	0.022988
DoS Hulk			**PortScan**	
Bwd Packet Length Std	0.514667		Flow Bytes/s	0.313358
Fwd Packet Length Std	0.073887		Total Length of Fwd Packets	0.304442
Fwd Packet Length Max	0.004256		Flow IAT Max	0.000255
Flow IAT Min	0.002005		Flow Duration	0.000252
DoS Slowhttptest			**SSH-Patator**	
Flow IAT Mean	0.645387		Fwd Packet Length Max	0.000931
Fwd Packet Length Min	0.082083		Flow IAT Mean	0.000642
Fwd Packet Length Std	0.018954		Flow Duration	0.000580
Fwd Packet Length Mean	0.017246		Flow IAT Max	0.000544
DoS slowloris			**Web Attack**	
Flow IAT Mean	0.474426		Bwd Packet Length Std	0.008407
Bwd Packet Length Mean	0.100013		Total Length of Fwd Packets	0.005940
Total Length of Bwd Packets	0.019652		Flow IAT Min	0.004051
Total Fwd Packets	0.006356		Flow Bytes/s	0.002704

Tabla 3.3: Distribución de los parámetros con mayor importancia para cada tipo de ataque en el dataset CICIDS2017

La distribución de los parámetros con sus valores de importancia para cada tipo de ataque puede verse en la Tabla 3.3, en la que se muestran sólo los 4 parámetros con mayor importancia. Los resultados completos para cada ataque pueden verse en el Apéndice B. A la hora de implementar los modelos de clasificación, utilizaremos para cada ataque los 4 parámetros que más importancia hayan mostrado en esta fase.

A partir de los resultados obtenidos observamos que en la mayoría de ataques hay uno o dos parámetros que destacan sobre los demás. Sin embargo, para los ataques Heartbleed y SSH-Patator la distribución es mucho más uniforme entre los distintos parámetros. Estas diferencias entre ataques pueden verse en la Figura 3.2, en la que se muestran los valores de importancia de los parámetros para los ataques Heartbleed, DoS Hulk, SSH-Patator y PortScan.

Las diferencias entre las distribuciones de importancia para unos ataques y otros se deben a la naturaleza de los mismos. En el caso de PortScan, por ejemplo, los ataques

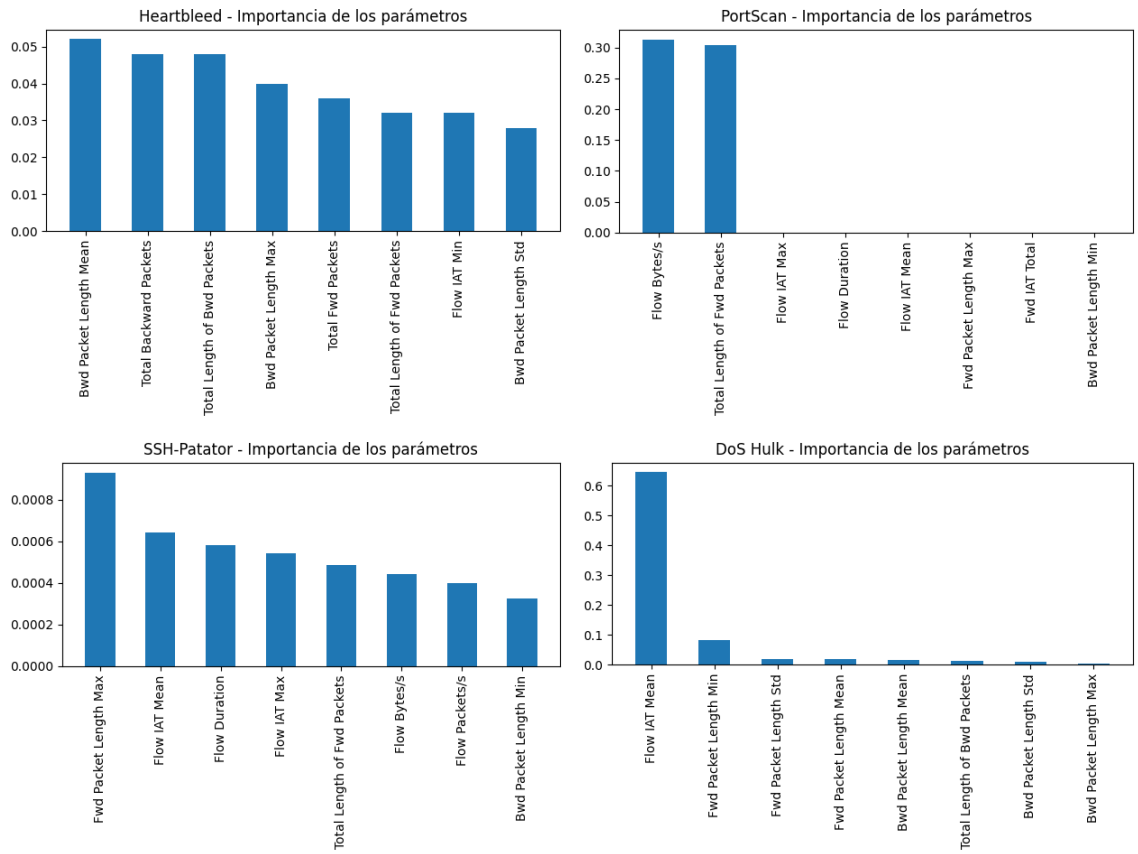

Figura 3.2: Importancia de los parámetros para los ataques Heartbleed, PortScan, SSH-Patator y DoS Hulk

suelen presentar una gran cantidad de mensajes con muy poca carga útil, por lo que los parámetros Flow Bytes/s y Total Length of Fwd Packets son, con diferencia, los que presentan una mayor importancia. Por otro lado, los ataques de tipo SSH-Patator son mucho más complejos, con diferentes fases de ataque, por lo que la importancia de los parámetros no está tan concentrada.

Una vez terminada la selección de parámetros para cada tipo de ataque, haremos ahora lo mismo para el conjunto total de los datos. De igual manera que para el caso anterior, utilizaremos el algoritmo Random Forest Regressor. En esta segunda aproximación los datos están categorizados como normales o malignos, sin distinguir el tipo de ataque en su caso. Los resultados de importancia para cada parámetro pueden verse en la tabla 3.4, que muestra las 20 etiquetas con mayor valor.

En este caso puede verse que los mayores valores de importancia los acumulan un reducido grupo de unos 4 o 6 parámetros. Este hecho puede verse en la Figura 3.3, en la que se muestran los 20 parámetros con mayor importancia. A la hora de escoger los parámetros para generar los modelos de detección de anomalías, se tomarán distintas combinaciones según el enfoque que se quiera dar al estudio. Estos detalles se especifican en el apartado 3.2.3. sobre la implementación.

Parámetro	Importancia		Parámetro	Importancia
Bwd Packet Length Std	0.246351		Flow IAT Max	0.003356
Flow Bytes/s	0.178529		Total Length of Bwd Packets	0.001771
Total Length of Fwd Packets	0.117104		Fwd Packet Length Min	0.000642
Fwd Packet Length Std	0.064064		Bwd Packet Length Mean	0.000560
Flow IAT Std	0.009626		Fwd Packet Length Mean	0.000558
Flow IAT Min	0.007102		Flow Packets/s	0.000541
Fwd IAT Total	0.005040		Fwd Packet Length Max	0.000157
Bwd Packet Length Max	0.004230		Total Backward Packets	0.000116
Flow Duration	0.004029		Total Fwd Packets	0.000076
Flow IAT Mean	0.003401		Bwd Packet Length Min	0.000051

Tabla 3.4: Distribución de los parámetros con mayor importancia para todo dataset CICIDS2017

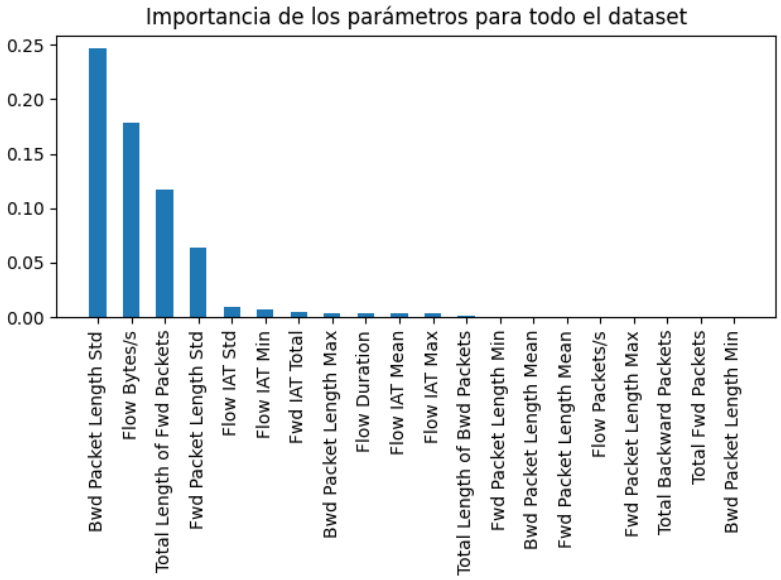

Figura 3.3: Importancia de los parámetros para el dataset completo CICIDS2017

3.2.2.2. UNSW-NB15

De igual forma que con el dataset anterior, con el UNSW-NB15 realizaremos primero la selección de parámetros por importancia en los subconjuntos de datos para cada tipo de ataque. Los archivos CSV creados en la fase anterior presentan, para cada tipo de anomalía, un 30% de tráfico malicioso y un 70% de tráfico normal (seleccionado aleatoriamente).

Para medir la importancia de cada parámetro utilizaremos de nuevo el algoritmo Random Forest Regressor. La distribución de los parámetros resultante con sus valores de importancia para cada tipo de ataque puede verse en la Tabla 3.5, en la que se muestran sólo los 4 parámetros con mayor importancia. Los resultados completos para cada ataque pueden verse en el Apéndice B. A la hora de implementar los modelos de clasificación, utilizaremos para cada ataque los 4 parámetros que más importancia

hayan mostrado en esta fase.

Ataque / Parámetros	Importancia		Ataque / Parámetros	Importancia
Analysis			**Generic**	
sttl	0.573390		service	0.018651
dsport	0.026739		dbytes	0.007203
dstip	0.004418		dsport	0.005746
proto	0.001617		Sload	0.000601
Backdoor			**Reconnaissance**	
sttl	0.927527		sttl	0.949993
dsport	0.025598		dsport	0.023367
dbytes	0.004252		proto	0.001591
Dload	0.000637		Dpkts	0.001574
DoS			**Shellcode**	
sttl	0.926416		sttl	0.823958
dsport	0.025332		dsport	0.012884
dbytes	0.003720		sbytes	0.008264
sbytes	0.001647		dbytes	0.001093
Exploits			**Worms**	
sttl	0.929542		sttl	0.944970
dsport	0.027404		dsport	0.022880
sbytes	0.004697		sbytes	0.003631
dbytes	0.001776		sport	0.000298
Fuzzers				
sttl	0.949261			
sloss	0.004627			
sport	0.004603			
Sload	0.002090			

Tabla 3.5: Distribución de los parámetros con mayor importancia para cada tipo de ataque en el dataset UNSW-NB15

A partir de los resultados obtenidos observamos que en la mayoría de ataques el parámetro “sttl” destaca notablemente frente a los demás y presenta el mayor nivel de importancia. Esto ocurre para todos los ataques excepto para el tipo Generic, que presenta una distribución de importancia algo más repartida. Estas diferencias entre ataques pueden verse en la Figura 3.4, en la que se muestran los valores de importancia de los parámetros para los ataques Generic y Fuzzers.

Una vez terminada la selección de parámetros para cada tipo de ataque, haremos ahora lo mismo para el conjunto total de los datos. De igual manera que para el caso anterior, utilizaremos el algoritmo Random Forest Regressor. En esta segunda aproximación los datos están categorizados como normales o malignos, sin distinguir el tipo de ataque en su caso. Los resultados de importancia para cada parámetro pueden verse en la tabla 3.6, que muestra las 20 etiquetas con mayor valor.

En este caso puede verse que los mayores valores de importancia los acumulan un reducido grupo de parámetros. Este hecho puede verse en la Figura 3.5, en la que

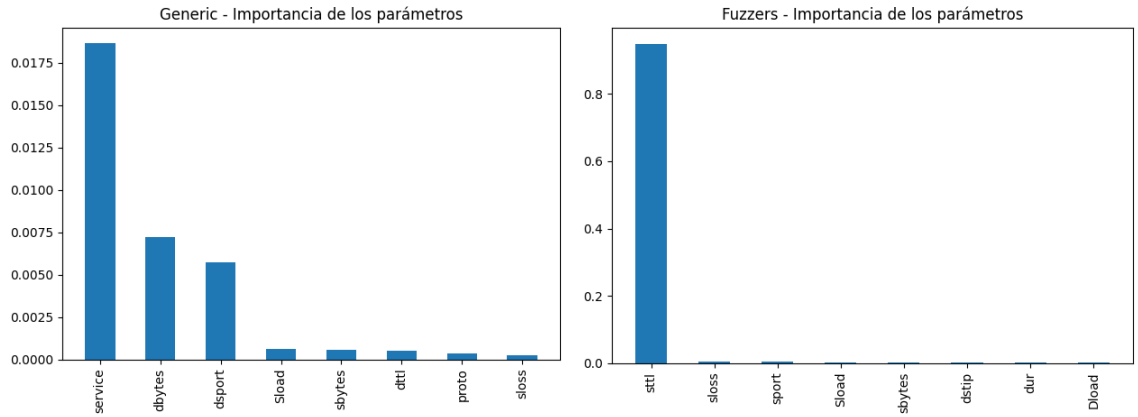

Figura 3.4: Importancia de los parámetros para los ataques Generic y Fuzzers

Parámetro	Importancia		Parámetro	Importancia
dsport	0.039197		Dload	0.001169
sbytes	0.031605		Dpkts	0.000744
sport	0.003610		state	0.000597
Sload	0.002237		dloss	0.000316
dbytes	0.002224		sloss	0.000247
Spkts	0.001922		proto	0.000207
dstip	0.001914		service	0.000076
sttl	0.001599		swin	0.000033
srcip	0.001248		dttl	0.000003
dur	0.001177		dwin	0.000001

Tabla 3.6: Distribución de los parámetros con mayor importancia para todo dataset UNSW-NB15

se muestran los 20 parámetros con mayor importancia. Es interesante observar que, a diferencia de los resultados de importancia para cada tipo de ataque, en este caso el parámetro “sttl” no aparece hasta la octava posición, muy lejos de ser el parámetro con más importancia. A la hora de escoger los parámetros para generar los modelos de detección de anomalías, se tomarán distintas combinaciones según el enfoque que se quiera dar al estudio. Estos detalles se especifican en el apartado 3.2.3. sobre la implementación.

3.2.3. Implementación de los algoritmos de aprendizaje automático

Una vez preprocesados los datos y analizada la importancia de los parámetros para el dataset, en este apartado se explica el proceso seguido en la implementación de los sistemas de detección de anomalías usando algoritmos de aprendizaje automático. Este proceso se hará de forma separada pero completamente análoga para cada dataset, y en el Capítulo 4 se hará un análisis de los resultados obtenidos.

Los algoritmos de aprendizaje automático utilizados en todos los casos serán los

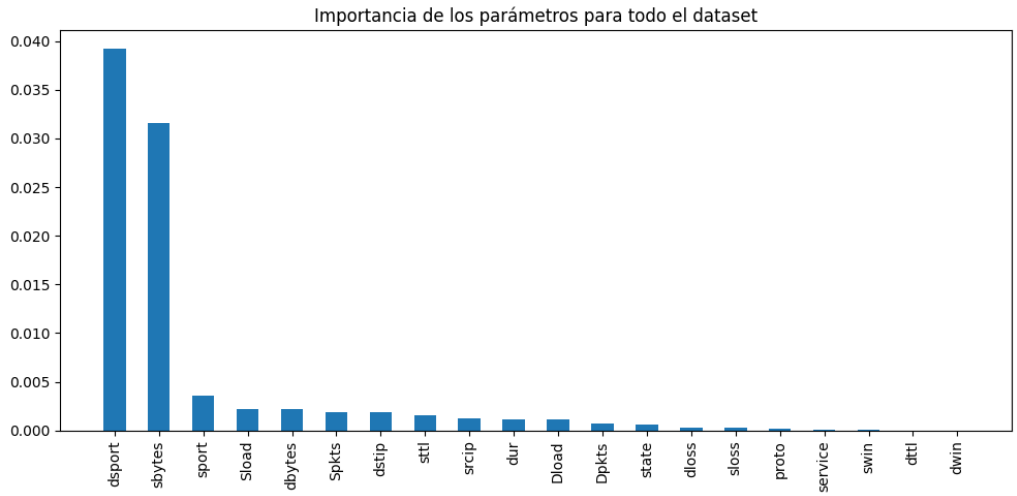

Figura 3.5: Importancia de los parámetros para el dataset completo UNSW-NB15

estudiados en la Sección 2.5.1, de los cuales 12 son supervisados y 4 no supervisados. Para este proceso, en cada caso se divide el conjunto de datos en subconjuntos de entrenamiento y test en una proporción 80 %-20 %, seleccionando los datos aleatoriamente. Cada modelo se entrenará con el primer subconjunto, y se evaluará el rendimiento de la clasificación con el segundo, recogiendo los resultados numéricos. Este proceso (división de los datos en entrenamiento-test, generar el modelo, ponerlo a prueba y recoger los datos) se realizará 10 veces con cada algoritmo y cada conjunto de datos, y tomaremos como resultado final la media de los valores de rendimiento obtenidos. De esta forma, al tomar una división aleatoria de los datos distinta en cada ejecución y repetirlo 10 veces, se reduce la probabilidad de que la división escogida sesgue la clasificación de alguna forma.

Como ya hemos adelantado en secciones anteriores, esta fase del estudio se realizará desde dos aproximaciones, dependiendo de los conjuntos de datos que se utilicen:

- **Clasificación sobre cada tipo de ataque:** Tal y como se explica en el apartado 3.2.1, por cada tipo de ataque se ha creado un archivo CSV que contiene un 30 % de tráfico malicioso (que recoge todas las muestras existentes de dicho ataque) y un 70 % de tráfico benigno (escogido aleatoriamente).

  Para cada uno de estos conjuntos de datos se implementarán distintos sistemas de detección de anomalías basados en diferentes algoritmos, de forma que clasifiquen el tráfico como benigno/malicioso. Estos sistemas utilizarán exclusivamente las 4 variables con mayor importancia en cada caso, que son las que se pueden ver en las tablas 3.3 y 3.5.

  Al tomar esta aproximación en el estudio, obtendremos un resultado separado por cada tipo de ataque, pudiendo comparar así el rendimiento de los clasificadores en función de la anomalía tratada.

- **Clasificación sobre el conjunto total de datos:** En esta segunda aproximación los datos se tratan globalmente, sin distinguir el tipo de ataque y sobre el data-

set completo. De igual forma que en el caso anterior, se implementarán distintos sistemas de detección de anomalías basados en diferentes algoritmos, de manera que clasifiquen el tráfico como benigno/malicioso. Los parámetros utilizados para desarrollar dichos modelos se tomarán de dos formas distintas, dando lugar a su vez a dos variantes dentro de esta aproximación:

- Combinando los 4 parámetros más importantes por cada tipo de ataque, de forma que el conjunto total sea el que se utilice sobre el dataset completo. En el caso del CICIDS2017 hay 12 ataques, y al quitar las repeticiones de los 48 parámetros escogidos obtenemos los 18 que se muestran en la tabla 3.7. Para el dataset UNSW-NB15, los 9 tipos distintos de ataque dan lugar a 36 parámetros, que sin contar los repetidos son los 12 que aparecen en la tabla 3.8.

Bwd Packet Length Max	Bwd Packet Length Mean
Bwd Packet Length Std	Flow Bytes/s
Flow Duration	Flow IAT Max
Flow IAT Mean	Flow IAT Min
Flow IAT Std	Fwd IAT Total
Fwd Packet Length Max	Fwd Packet Length Mean
Fwd Packet Length Min	Fwd Packet Length Std
Total Backward Packets	Total Fwd Packets
Total Length of Bwd Packets	Total Length of Fwd Packets

Tabla 3.7: Selección de parámetros combinando cada tipo de ataque para CICIDS2017

service	dbytes	dsport
Sload	sttl	sbytes
sloss	sport	proto
Dpkts	dstip	Dload

Tabla 3.8: Selección de parámetros combinando cada tipo de ataque para UNSW-NB15

- Seleccionando los 7 parámetros con mayor importancia sobre el conjunto total de datos, según se muestra en las Tablas 3.4 y 3.6. Al escoger estas variables estaremos trabajando con más del 90 % del valor de importancia de todas las variables, concretamente un 97 % para el CICIDS2017 y un 90 % para el UNSW-NB15. Los parámetros seleccionados para cada dataset junto con su importancia global y su porcentaje sobre el total se pueden ver en las tablas 3.9 y 3.10.

Parámetro	Importancia	Porcentaje de importancia
Bwd Packet Length Std	0.246	38.058 %
Flow Bytes/s	0.179	27.580 %
Total Length of Fwd Packets	0.117	18.091 %
Fwd Packet Length Std	0.064	9.897 %
Flow IAT Std	0.009	1.487 %
Flow IAT Min	0.007	1.097 %
Fwd IAT Total	0.005	0.779 %

Tabla 3.9: Los 7 parámetros con mayor importancia para CICIDS2017

Parámetro	Importancia	Porcentaje de importancia
dsport	0.039	43.492 %
sbytes	0.032	35.068 %
sport	0.004	4.006 %
Sload	0.002	2.482 %
dbytes	0.002	2.468 %
Spkts	0.002	2.132 %
dstip	0.002	2.124 %

Tabla 3.10: Los 7 parámetros con mayor importancia para UNSW-NB15

Capítulo 4

Resultados y Evaluación

En este capítulo se exponen los resultados de las pruebas que se han descrito anteriormente. Se detallan separadamente los resultados para cada dataset, y en cada uno de ellos se distinguen las diferentes aproximaciones de la implementación. Los 16 algoritmos utilizados en los modelos de detección de anomalías son los que se detallan en la Sección 2.4.

En todos los casos se incluyen los valores de F-Score, que tomaremos como referencia para comparar los distintos algoritmos. Como ya se ha adelantado en la Sección 3.2.3, para evaluar el rendimiento de los algoritmos, cada ejecución (división aleatoria de los datos en entrenamiento-test, entrenamiento del modelo, pruebas de clasificación con datos de test y recogida de resultados) se ha realizado 10 veces y los resultados que aquí se muestran son la media de todas las ejecuciones. Los resultados completos, con todas las variables recogidas en las ejecuciones, pueden verse en los Apéndices C y D para CICIDS2017 y UNSW-NB15, respectivamente.

En el trabajo realizado sobre el dataset CICIDS2017 se incluye también un apartado de evaluación, en el que se comparan los resultados obtenidos aquí con los de otro estudio relacionado [3]. Esto se hace exclusivamente con el dataset CICIDS2017 y no con UNSW-NB15 ya que dicho trabajo sólo incluye la implementación con el primer dataset.

4.1. CICIDS2017

En esta sección expondremos los resultados obtenidos para el dataset CICIDS2017. Distinguiremos dos casos, uno por cada enfoque de la implementación. El primer caso es para la clasificación separada por cada tipo de ataque, y el segundo para la la clasificación sobre el conjunto total de datos. En todos los casos se recogen los resultados de los 16 algoritmos de aprendizaje automático detallados anteriormente.

Por último, en el apartado de Evaluación, se comparan los resultados obtenidos en el trabajo de Kostas et al. [3] con los de este estudio, para aquellos algoritmos que se

hayan utilizado en ambos trabajos.

4.1.1. Caso 1 - clasificación sobre cada tipo de ataque

En este caso las pruebas se han realizado separadamente sobre cada tipo de ataque, 12 en total. En la tabla 4.1 pueden verse los resultados de F-Score obtenidos con los algoritmos de aprendizaje supervisado, y en la 4.2 con los no supervisados. En ambos casos se destaca en negrita para cada ataque el mejor resultado de entre todos los algoritmos, y subrayado el peor. En caso de empate de F-Score se distinguirá por accuracy, precission, recall y tiempo de ejecución, en ese orden (véase el Apéndice C con todos los valores).

Ataque	**F-Score**											
	GNB	QDA	RF	ID3	AB	MLP	KNN	LR	SVM	MNB	SGD	GBC
Bot	0.45	0.68	0.94	0.95	0.97	0.66	0.96	0.66	0.41	0.52	0.55	**0.97**
DDoS	0.3	0.81	0.97	**0.97**	0.97	0.77	0.92	0.81	0.83	0.68	0.62	0.97
DoS GoldenEye	0.77	0.67	0.99	**0.99**	0.98	0.7	0.97	0.92	0.93	0.72	0.77	0.99
DoS Hulk	0.3	0.4	0.92	0.95	0.95	0.94	**0.95**	0.83	0.85	0.81	0.66	0.95
DoS Slowhttptest	0.92	0.93	0.97	0.96	0.97	0.66	0.97	0.93	0.94	0.77	0.4	**0.98**
DoS slowloris	0.4	0.48	0.93	**0.95**	0.95	0.74	0.93	0.89	0.79	0.87	0.69	0.95
FTP-Patator	**1.0**	1.0	1.0	1.0	1.0	1.0	1.0	0.75	0.77	0.75	0.59	1.0
Heartbleed	0.84	**1.0**	1.0	1.0	1.0	0.54	1.0	1.0	1.0	0.84	1.0	1.0
Infiltration	0.89	0.76	0.87	**0.89**	0.89	0.3	0.84	0.76	0.89	0.55	0.75	0.89
PortScan	0.43	0.79	0.99	**1.0**	0.99	0.49	1.0	0.46	0.84	0.45	0.61	0.99
SSH-Patator	0.42	0.57	0.95	**0.96**	0.96	0.85	0.95	0.45	0.41	0.39	0.41	0.96
Web Attack	0.77	0.83	0.95	0.95	**0.95**	0.88	0.94	0.47	0.46	0.39	0.49	0.95

Tabla 4.1: Resultados de los algoritmos supervisados por cada tipo de ataque en CICIDS2017

Con un primer análisis de los resultados para los modelos supervisados, es fácil ver que los algoritmos Random Forest, ID3, AdaBoost, KNN y Gradient Boost Classifier han conseguido unos resultados de efectividad superiores al 90 % en la mayoría de ataques. De entre ellos, el más destacable es el algoritmo ID3, el cual ha ofrecido los mejores resultados en 6 de los 12 casos. Cabe matizar que en muchos de estos casos su marca de rendimiento estaba igualada con la de otros algoritmos como AdaBoost o Random Forest, pero su velocidad de procesamiento coloca al ID3 como un mejor algoritmo en estos casos.

Por otro lado está el algoritmo Gaussian Naïve Bayes, que arroja los peores resultados en cuanto a F-Score en 4 de los 12 casos de ataque. Cercano a este algoritmo están el QDA y el Multinomial Naïve Bayes, que muestran unos resultados muy similares. Esto se debe a que todos estos algoritmos son métodos fuertemente estadísticos, y habitualmente tienen un desempeño peor que otros. Sin embargo, se observa también que tanto GNB como QDA ofrecen buenos resultados en otros ataques, hasta de 1.0 para FTP-Patator o Heartbleed, por lo que en ningún caso han de considerarse malos algoritmos. Estas diferencias en cuanto a resultados pueden deberse a la naturaleza estadística de los métodos, que dependiendo de la estructura del dataset con el que trabajen su desempeño puede variar notablemente.

Otro punto destacable a partir de los resultados obtenidos es que buena parte de los algoritmos han conseguido un rendimiento del 100 % para los ataques FTP-Patator y Heartbleed, concretamente en 8 y 9, respectivamente, de los 12 algoritmos supervisados que aquí aparecen. Este extraordinario resultado puede deberse a que para estos dos algoritmos los parámetros estudiados resulten especialmente reveladores para diferenciar entre tráfico normal y ataques.

Otros algoritmos, como el Multi-layer Perceptron, Logistic Regression, Support Vector Machine o Stochastic Gracient Descent, presentan un rango muy amplio de valores de rendimiento, desde marcas tan bajas como 0.3 hasta otras con 1.0. Concretamente para el caso del MLP, se observa que los resultados más bajos los presenta para los ataques Heartbleed e Infiltration, que son de los dos con menor número de datos de tráfico. Viendo por otro lado que para DoS Hulk, que es el ataque con más registros, ha obtenido un rendimiento del 94 %, se puede inferir que el algoritmo MLP muestra mejor desempeño cuantos más datos tenga para trabajar.

Ataque	**F-Score**			
	K-MEANS	K-MEDOIDS	IF	LOF
Bot	0.39	0.38	**0.44**	0.43
DDoS	0.59	0.58	**0.61**	0.47
DoS GoldenEye	0.56	**0.84**	0.79	0.47
DoS Hulk	0.42	**0.83**	0.57	0.5
DoS Slowhttptest	0.89	**0.9**	0.59	0.42
DoS slowloris	0.59	**0.61**	0.52	0.42
FTP-Patator	0.41	0.22	0.28	**0.48**
Heartbleed	1.0	**1.0**	0.53	0.3
Infiltration	0.74	0.87	0.63	**0.94**
PortScan	**0.39**	0.39	0.3	0.34
SSH-Patator	0.39	**0.65**	0.5	0.39
Web Attack	**0.41**	0.36	0.35	0.35

Tabla 4.2: Resultados de los algoritmos no supervisados por cada tipo de ataque en CICIDS2017.

En cuanto a las pruebas realizadas con los algoritmos de aprendizaje no supervisado, la comparación resulta más limitada ya que sólo hay 4 algoritmos con los que se han realizado pruebas. En términos generales, los resultados de rendimiento en la clasificación son sensiblemente peores que para los algoritmos supervisados. Este hecho, que cabía esperarse, se debe al distinto enfoque que hacen estos algoritmos, que en ningún momento conocen la clasificación real del tráfico al desarrollar el modelo.

El algoritmo K-Medoids sobresale notablemente entre los demás, contando con la mejor marca en 6 de los 12 algoritmos, llegando a marcar valores de 0.9 y 1.0 para DoS Slowhttptest y Heartbleed, respectivamente.

Para algunos ataques, como Bot, PortScan o WebAttack, ninguno de los algoritmos ha conseguido obtener un resultado superior a 0.5. Estos son resultados en ningún caso pueden considerarse buenos, ya que para dichos ataques los modelos no son ca-

paces de detectarlos ni en la mitad de los casos.

Cabe destacar también que para todos los ataques excepto Infiltration, los resultados varían mucho dependiendo del algoritmo del que se trate, siempre con alguna marca inferior a 0.5 para alguno de los modelos. El caso del ataque Infiltration es seguramente con el que mejor desempeño global se obtiene, con marcas entre el 63 % y el 94 % de precisión.

4.1.2. Caso 2 - clasificación sobre el conjunto total de datos

En esta sección se muestran los resultados de la clasificación sobre todo el dataset CICIDS2017, de forma que todos los ataques quedan bajo una única etiqueta de "ataque". Este enfoque se ha realizado a su vez de dos formas: tomando los 4 parámetros de mayor importancia de cada tipo de ataque, que resultan en 18 parámetros distintos; y tomando los 7 parámetros de mayor importancia sobre el conjunto total de datos (véase la Sección 3.2.2 sobre la selección de parámetros).

4.1.2.1. Para 18 parámetros de ataques

En la Tabla 4.3 se muestran los resultados obtenidos usando los 18 parámetros resultantes de combinar los 4 más importantes para cada ataque. Igual que en las tablas anteriores, se destaca en negrita el algoritmo con mejor desempeño y subrayado el peor, tomando como métrica de orden la F-Score. Estas marcas se hacen separadamente para los algoritmos supervisados y los no supervisados, ya que su naturaleza y rendimiento son notablemente distintos.

Se puede observar en la tabla 4.3 que el desempeño en general de los algoritmos usados es bastante bueno, habiendo sólo un caso con marca inferior a 0.5.

De entre los algoritmos supervisados, destacan ID3, AdaBoost, KNN y GBC, todos ellos con resultados superiores al 90 %. El algoritmo con mejor desempeño es el árbol de decisión ID3, con un valor de F-Score de 0.96. El que peor rendimiento ofrece es el algoritmo QDA, con 0.62 de F-Score, seguido de GNB con 0.63 y de Logistic Regression con 0.65. Otros modelos, como MLP, Random Forest, LR o SVM, si bien no llegan al 90 %, tienen todos marcas superiores al 70 %, por lo que pueden considerarse clasificadores razonablemente precisos.

En el caso de los algoritmos no supervisados, el que mejor marca de rendimiento tiene es K-Means con un F-Score de 0.64. En general, y como ya hemos visto en el enfoque anterior, los resultados de rendimiento que ofrecen los algoritmos no supervisados son sensiblemente inferiores a los supervisados. De los cuatro algoritmos usados, Local Outlier Factor es el que peor marca ofrece con un F-Score de 0.49 (el único por debajo de 0.5), y K-Medoids e Isolation Forest quedan ligeramente por encima del 50 % de precisión.

Algoritmo	**F-Score**
Gaussian Naïve Bayes	0.63
QDA	0.62
Random Forest	0.89
ID3	**0.96**
AdaBoost	0.9
MLP	0.83
K Nearest Neighbours	0.94
Logistic Regression	0.65
Support Vector Machine	0.7
Multinomial Naïve Bayes	0.67
SGD Classifier	0.67
Gradient Boost Classifier	0.91
K-Means	**0.64**
K-Medoids	0.55
Isolation Forest	0.55
Local Outlier Factor	0.49

Tabla 4.3: Resultados de los algoritmos sobre todo el dataset CICIDS2017 con 18 parámetros.

4.1.2.2. Para 7 parámetros generales

En la Tabla 4.4 se muestran los resultados obtenidos usando los 7 parámetros más importantes sobre el dataset completo. Igual que en las tablas anteriores, se destaca en negrita el algoritmo con mejor desempeño y subrayado el peor, tomando como métrica de orden la F-Score. Estas marcas se hacen separadamente para los algoritmos supervisados y los no supervisados, ya que su naturaleza y rendimiento son notablemente distintos.

En un primer vistazo a la tabla 4.4, se puede ver que los resultados son muy similares a los obtenidos al usar 18 parámetros. En la mayoría de casos la variación es a lo sumo de 0.05, a excepción del SGD Classifier, que baja más de 0.25 de F-Score.

Para los algoritmos supervisados, igual que en el caso anterior, destacan ID3, KNN y GBC, con con resultados superiores al 90 %. En este caso, el algoritmo que peor rendimiento ofrece es SGD, con 0.41 de F-Score, seguido por MNB y por GNB con 0.64. En esta aproximación, sólo un algoritmo no supera el umbral de rendimiento de 0.5, y es el SGD Classifier. Otros modelos, como RF, AdaBoost, LR o SVM, si bien no llegan al 90 %, tienen todos marcas superiores al 70 %, haciendo de ellos clasificadores con un rendimiento razonablemente bueno.

En el caso de los algoritmos no supervisados, el cambio más notable se produce en K-Medoids, que sube 0.09 de F-Score para convertirse en el mejor resultado de entre estos algoritmos, con una marca de 0.64. Lo iguala K-Means en F-Score, pero este último resulta peor en cuanto a tiempo de ejecución. Se observa también que en este caso los cuatro algoritmos utilizados obtienen valores superiores a 0.5, cosa que no ocurría

Algoritmo	**F-Score**
Gaussian Naïve Bayes	0.64
QDA	0.7
Random Forest	0.87
ID3	**0.96**
AdaBoost	0.89
MLP	0.87
K Nearest Neighbours	0.94
Logistic Regression	0.7
Support Vector Machine	0.7
Multinomial Naïve Bayes	0.64
SGD Classifier	0.41
Gradient Boost Classifier	0.9
K-Means	0.64
K-Medoids	**0.64**
Isolation Forest	0.55
Local Outlier Factor	0.5

Tabla 4.4: Resultados de los algoritmos sobre todo el dataset CICIDS2017 con 7 parámetros.

al utilizar 18 parámetros.

4.1.3. Evaluación

En este apartado vamos a comparar los resultados obtenidos aquí con los que ofrece el trabajo de Kostas et al. [3], que como ya hemos adelantado, tomamos como referencia en el desarrollo de nuestro estudio.

Pese a que en el presente trabajo se han utilizado una mayor cantidad y variedad de algoritmos de aprendizaje automático, sólo se puede hacer la comparación con aquellos que fueron utilizados también en el trabajo de Kostas [3]. Dichos algoritmos son los siguientes: Gaussian Naïve Bayes (GNB), Quadratic Discriminant Analysis (QDA), Random Forest (RF), Iterative Dichotomiser 3 (ID3), AdaBoost (AB), Multi-layer Perceptron (MLP) y K Nearest Neighbours (KNN). Para realizar esta comparación tomaremos de nuevo como medida de referencia el F-Score.

Puesto que en el trabajo de Kostas et al. [3] se realizaron las pruebas desde los mismos dos enfoques, aquí compararemos los resultados obtenidos con 18 y 7 parámetros, respectivamente.

En primer lugar veamos la comparación para el caso en el que se tomaron 18 parámetros, resultado de combinar los 4 parámetros de mayor importancia para todos los ataques. En la tabla 4.5 se pueden ver los resultados de F-Score para este estudio y el de Kostas.

En este caso se puede ver que, en general, los resultados se comportan de una forma parecida. Para algoritmos como Random Forest, ID3, AdaBoost, MLP o KNN, el rendimiento obtenido es muy similar en ambos estudios, con diferencias de a lo sumo un 5%. En cambio, los algoritmos GNB y QDA presentan diferencias de rendimiento superiores, llegando a más de 0.3 en el caso de QDA. En la figura 4.1 pueden verse gráficamente todas estas diferencias mencionadas.

Se observa que en algunos casos este estudio obtiene rendimientos superiores a los del trabajo de Kostas[3], como es el caso de QDA, ID3 o MLP. Para los demás, en cambio, los resultados de nuestro trabajo son ligeramente peores.

Algoritmo	**Este estudio**	**Kostas et al.**
Gaussian Naïve Bayes	0.63	0.79
QDA	0.62	0.30
Random Forest	0.89	0.94
ID3	0.96	0.95
AdaBoost	0.9	0.95
MLP	0.83	0.79
K Nearest Neighbours	0.94	0.96

Tabla 4.5: Comparación de los resultados de F-Score obtenidos en ambos trabajos para CICIDS2017 y 18 parámetros

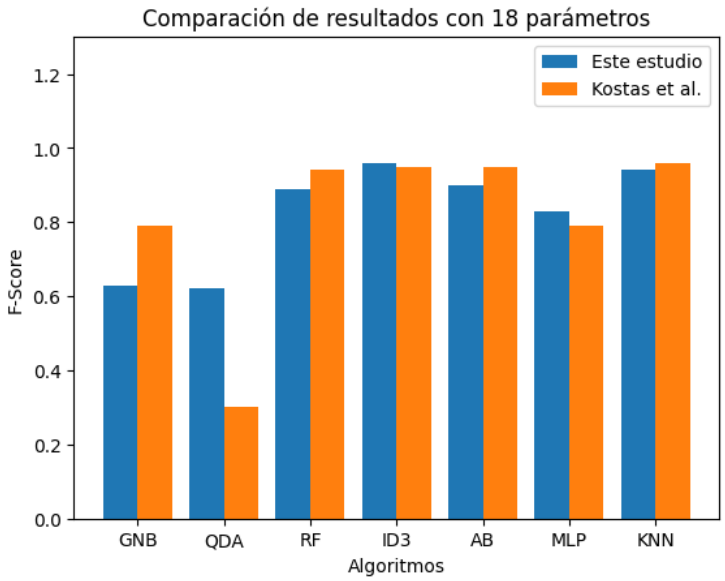

Figura 4.1: Comparación de los resultados de F-Score obtenidos en ambos trabajos para CICIDS2017 y 18 parámetros

Para el caso con 7 parámetros, podemos ver la comparación de resultados en la tabla 4.6. En ella se ve de nuevo una similitud muy grande entre los resultados de ambos estudios. Esto se da de una forma muy clara en los algoritmos Random Forest, ID3, AdaBoost, MLP y KNN, que presentan diferencias de no más del 8%. De nuevo las mayores diferencias se dan para los algoritmos GNB y QDA, cada uno marcando un

rendimiento superior para cada estudio. En la figura 4.2 pueden verse gráficamente los resultados para ambos trabajos.

Para algoritmos como QDA, ID3 o MLP, los rendimientos obtenidos en este estudio son algo superiores a los del estudio de Kostas[3], y en el resto la diferencia es en sentido contrario.

Algoritmo	**Este estudio**	**Kostas et al.**
Gaussian Naïve Bayes	0.64	0.81
QDA	0.7	0.41
Random Forest	0.87	0.94
ID3	0.96	0.95
AdaBoost	0.89	0.94
MLP	0.87	0.79
K Nearest Neighbours	0.94	0.97

Tabla 4.6: Comparación de los resultados de F-Score obtenidos en ambos trabajos para CICIDS2017 y 7 parámetros

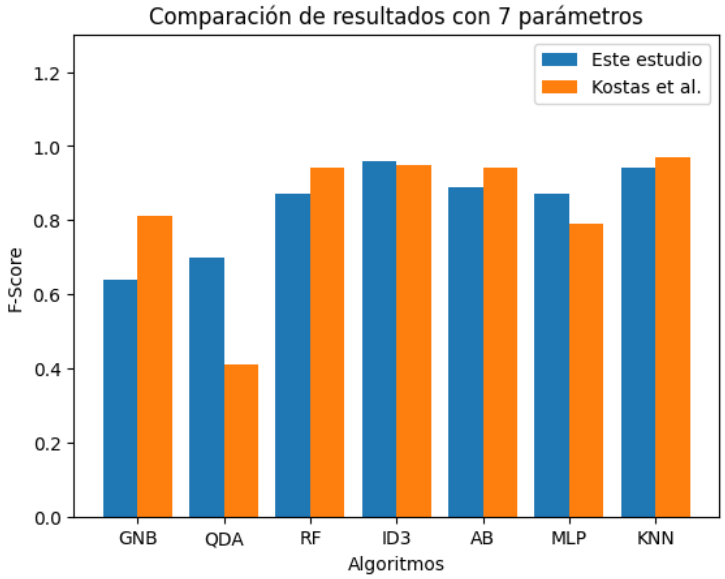

Figura 4.2: Comparación de los resultados de F-Score obtenidos en ambos trabajos para CICIDS2017 y 7 parámetros

Como se puede ver, tanto para el caso de 18 como para el de 7 parámetros, en la mayoría de algoritmos los rendimientos obtenidos son muy similares, en ocasiones mejorando el estudio de Kostas y en otros quedando algo por detrás. Estas diferencias se deben a varios factores:

- El distinto tratamiento de los datos antes de crear los modelos: En nuestro estudio, antes de implementar algunos modelos de aprendizaje automático como QDA o MLP, se han estandarizado los datos, cosa que no se hizo en el trabajo de

Kostas et al. Esto da lugar a una notable mejora en los resultados de estos algoritmos.

- El uso de diferentes hiperparámetros a la hora de implementar los modelos de aprendizaje: Al utilizar las funciones de la librería Sklearn para implementar los algoritmos, se han utilizado hiperparámetros ligeramente distintos, lo que da lugar a modelos algo diferentes, y por tanto a variaciones en los resultados de rendimiento.

- La diferencia temporal entre ambos estudios: El trabajo de Kostas et al. [3] data de 2017, y desde entonces las librerías de Python como Sklearn se han actualizado en numerosas ocasiones, por lo que sus métodos y funciones propias han podido cambiar ligeramente. Estos cambios pueden provocar diferencias sutiles en cuanto al rendimiento de los modelos.

- La selección aleatoria de los conjuntos de entrenamiento y test: aunque se respeten las mismas proporciones (80 %-20 %), esto puede dar lugar a diferentes resultados en la generación de los modelos y en la evaluación posterior. Dependiendo de cuál haya sido la distribución de los datos utilizados en cada ejecución de entrenamiento-test, los resultados obtenidos pueden ser distintos.

En cualquier caso, es importante destacar que el objetivo de este estudio no es una mejora cuantitativa de los resultados obtenidos en el trabajo de Kostas, sino una amplaición en múltiples direcciones. El uso de muchos otros algoritmos supervisados, añadiendo también el enfoque no supervisado, y la aplicación de los modelos sobre otro dataset además del CICIDS2017, suponen una expansión notable en cuanto al ámbito de estudio.

4.2. UNSW-NB15

En esta sección expondremos los resultados obtenidos para el dataset UNSW-NB15. Distinguiremos dos casos, uno por cada enfoque de la implementación. El primer caso es para la clasificación separada por cada tipo de ataque, y el segundo para la la clasificación sobre el conjunto total de datos. En todos los casos se recogen los resultados de los 16 algoritmos de aprendizaje automático detallados anteriormente.

4.2.1. Caso 1 - clasificación sobre cada tipo de ataque

En este caso las pruebas se han realizado separadamente sobre cada tipo de ataque, 9 en total. En la tabla 4.7 pueden verse los resultados de F-Score obtenidos con los algoritmos de aprendizaje supervisado, y en la 4.8 con los no supervisados. En ambos casos se destaca en negrita para cada ataque el mejor resultado de entre todos los algoritmos, y subrayado el peor. En caso de empate de F-Score se ordenará por accuracy,

precission, recall y tiempo de ejecución, en ese orden (véase el Apéndice D con todos los valores).

Con un primer vistazo a las tablas, se aprecia que los resultados de rendimiento en las clasificaciones son en general bastante altos. Los algoritmos que mejor desempeño general tienen son Random Forest, ID3, AdaBoost y GBC, todos ellos con valores superiores a 0.9 en todos los ataques con los que se ha probado. El algoritmo KNN tiene también muy buen desempeño, y presenta tres valores máximos para los ataques DoS, Exploits y Generic, siendo el modelo con más resultados pico.

Por otro lado, el algoritmo GNB es el que peores resultados arroja en términos generales, con cuatro valores mínimos para los ataques Backdoor, Reconnaissance, Fuzzers y Worms. En estos dos últimos, además, ofrece valores que no llegan a alcanzar un rendimiento del 50 %. El algoritmo QDA también ofrece un valor por debajo de 0.5, concretamente para el ataque Shellcode, pero resulta excepcional frente al resto de rendimientos, que son de un nivel realmente alto.

Otros modelos, como MLP, MNB o SGD presentan valores de rendimiento más variados, desde marcas más bajas como 0.42 o 0.6 hasta otras tan altas como 0.98. Cabe decir que en todos ellos, en términos generales, los resultados siguen siendo buenos, pero no tan altos y constantes como otros algoritmos.

Ataque	**F-Score**											
	GNB	QDA	RF	ID3	AB	MLP	KNN	LR	SVM	MNB	SGD	GBC
Analysis	0.83	0.98	**1.0**	0.99	1.0	0.93	0.99	0.9	0.9	0.76	0.68	0.98
Backdoor	0.5	0.96	0.98	0.99	0.98	0.57	0.95	0.97	0.97	0.88	0.93	**0.99**
DoS	0.93	0.93	0.99	0.99	0.99	0.93	**0.99**	0.94	0.95	0.88	0.91	0.98
Exploits	0.81	0.84	0.98	0.99	0.99	0.96	**0.99**	0.95	0.82	0.6	0.74	0.98
Fuzzers	0.41	**0.99**	0.99	0.99	0.99	0.61	0.71	0.99	0.99	0.69	0.92	0.99
Generic	0.58	0.65	0.99	0.99	0.99	0.42	**1.0**	0.87	0.92	0.91	0.78	0.99
Reconnaissance	0.85	0.99	0.99	**0.99**	0.99	0.98	0.99	0.99	0.99	0.87	0.97	0.98
Shellcode	1.0	0.23	**1.0**	1.0	1.0	1.0	1.0	0.99	1.0	0.77	0.98	0.99
Worms	0.42	0.98	0.99	0.99	0.99	0.45	0.55	0.97	0.97	0.68	0.97	**1.0**

Tabla 4.7: Resultados de los algoritmos supervisados por cada tipo de ataque en UNSW-NB15.

En cuanto a los algoritmos no supervisados, podemos ver, tal y como se podía prever, que sus resultados son sensiblemente inferiores a los de los algoritmos supervisados. Esto no supone una sorpresa ni un problema, ya que la naturaleza de estos modelos hace que sean de por sí menos eficaces en cuanto a rendimiento, pero mucho más útiles en el uso real ya que no siempre se dispone de datos etiquetados.

En este caso, podemos ver en la tabla 4.8 que el algoritmo que mejor desempeño muetra es Isolation Forest, con 5 marcas máximas para los ataques DoS, Exploits, Fuzzers, Shellcode y Worms. Sin embargo, este mismo modelo presenta resultados muy bajos para Analysis, Generic o Reconnaisance, para los cuales no llega a alcanzar ni un 50 % de rendimiento.

En el lado opuesto encontramos al algoritmo K-Medoids, que ofrece los peores resultados de entre los cuatro algoritmos, marcando hasta en 5 ataques los valores más

bajos. En todos ellos, no llega a alcanzar el umbral de 0.5. Por contra, este mismo modelo alcanza rendimientos de un 83 % y un 91 %, por lo que presenta una gran variedad de marcas.

Es reseñable ver que para algunos ataques, como Analysis o Reconnaisance, ninguno de los cuatro algoritmos ha sido capaz de obtener un resultado superior a 0.5. Esta gran cantidad de rendimientos bajos, que aparecen para todos los algoritmos no supervisados, supone un problema ya que resultan menos eficaces a la hora de clasificar, pero se ha de tener en cuenta que su capacidad es mucho más limitada que la de los modelos supervisados. El hecho de que se obtengan también resultados altos en la clasificación (80 %, 90 %), resulta esperanzador.

Ataque	**F-Score**			
	K-MEANS	K-MEDOIDS	IF	LOF
Analysis	0.4	0.28	0.36	**0.49**
Backdoor	0.56	**0.83**	0.53	0.52
DoS	0.41	0.38	**0.52**	0.48
Exploits	0.42	0.38	**0.69**	0.55
Fuzzers	0.5	0.48	**0.65**	0.58
Generic	0.57	**0.91**	0.35	0.36
Reconnaissance	0.37	0.37	0.4	**0.41**
Shellcode	0.4	0.27	**0.66**	0.42
Worms	0.59	0.62	**0.63**	0.58

Tabla 4.8: Resultados de los algoritmos no supervisados por cada tipo de ataque en UNSW-NB15.

4.2.2. Caso 2 - clasificación sobre el conjunto total de datos

En esta sección se muestran los resultados de la clasificación sobre todo el dataset UNSW-NB15, de forma que todos los ataques quedan bajo una única etiqueta de "ataque". Este enfoque se ha realizado a su vez de dos formas: tomando los 4 parámetros de mayor importancia de cada tipo de ataque, que resultan en 12 parámetros distintos; y tomando los 7 parámetros de mayor importancia sobre el conjunto total de datos (véase la Sección 3.2.2 sobre la selección de parámetros).

4.2.2.1. Para 12 parámetros de ataques

En la Tabla 4.9 se muestran los resultados obtenidos usando los 12 parámetros resultantes de combinar los 4 más importantes para cada ataque. Igual que en las tablas anteriores, se destaca en negrita el algoritmo con mejor desempeño y subrayado el peor, tomando como métrica de orden la F-Score. Estas marcas se hacen separadamente para los algoritmos supervisados y los no supervisados, ya que su naturaleza y rendimiento son notablemente distintos.

Algoritmo	**F-Score**
Gaussian Naïve Bayes	0.58
QDA	0.94
Random Forest	0.96
ID3	0.97
AdaBoost	**0.98**
MLP	<u>0.48</u>
K Nearest Neighbours	0.97
Logistic Regression	0.95
Support Vector Machine	0.95
Multinomial Naïve Bayes	0.94
SGD Classifier	0.79
Gradient Boost Classifier	0.95
K-Means	<u>0.46</u>
K-Medoids	0.54
Isolation Forest	0.53
Local Outlier Factor	**0.57**

Tabla 4.9: Resultados de los algoritmos sobre todo el dataset UNSW-NB15 con 12 parámetros.

En cuanto a los algoritmos supervisados, los rendimientos obtenidos son realmente buenos, habiendo sólo uno por debajo del 50 %. AdaBoost es el modelo que mejor resultado ofrece, con un 98 % de efectividad. Le siguen ID3, KNN y Random Forest, con valores superiores al 95 %.

El peor registro lo marca MLP, con 0.48 de F-Score, seguido de Gaussian Naïve Bayes, con 0.58. A excepción de estos dos modelos, los demás no bajan prácticamente del 80 % de efectividad, situándose la mayoría por encima del 90 %.

Los algoritmos no supervisados, por su parte, presentan valores muy inferiores, como cabría esperar. El que mejor rendimiento marca es Local Outlier Factor, con un 57 %. El peor valor lo proporciona K-Means, con 0.46 de F-Score. Esto supone que hay un rango realmente estrecho de resultados de rendimiento en estos algoritmos, situándose todos ellos muy cercanos al 50 %.

4.2.2.2. Para 7 parámetros generales

En la Tabla 4.10 se muestran los resultados obtenidos usando los 7 parámetros más importantes sobre el dataset completo. Igual que en las tablas anteriores, se destaca en negrita el algoritmo con mejor desempeño y subrayado el peor, tomando como métrica de orden la F-Score. Estas marcas se hacen separadamente para los algoritmos supervisados y los no supervisados, ya que su naturaleza y rendimiento son notablemente distintos.

Observando los algoritmos supervisados, los resultados obtenidos son razonable-

Algoritmo	**F-Score**
Gaussian Naïve Bayes	0.51
QDA	0.56
Random Forest	0.94
ID3	0.96
AdaBoost	0.97
MLP	0.47
K Nearest Neighbours	**0.97**
Logistic Regression	0.62
Support Vector Machine	0.47
Multinomial Naïve Bayes	0.56
SGD Classifier	0.54
Gradient Boost Classifier	0.81
K-Means	0.46
K-Medoids	**0.6**
Isolation Forest	0.51
Local Outlier Factor	0.58

Tabla 4.10: Resultados de los algoritmos sobre todo el dataset UNSW-NB15 con 7 parámetros.

mente buenos, aunque ligeramente peores que los del enfoque anterior. En este caso, se han obtenido dos por debajo del 50 %. KNN es el modelo que mejor resultado ofrece, con un 97 % de efectividad. Le siguen AdaBoost, ID3 y Random Forest, con valores superiores al 94 %.

El peor registro lo marca de nuevo MLP, con 0.47 de F-Score, seguido de SVM, con 0.47, y de Gaussian Naïve Bayes con 0.51. En este enfoque se dan más resultados cercanos al umbral de 0.5 que en el anterior, como es el caso de SGD, MNB y QDA. Sólo GBC y Logistic Regression quedan en la horquilla entre el 60 % y el 80 %, a diferencia de lo ocurrido con 12 parámetros

Por otro lado, los algoritmos no supervisados presentan resultados sensiblemente inferiores a los supervisados, pero en este caso son algo mejores que en el enfoque anterior con 12 parámetros. El que mejor rendimiento presenta es K-Medoids, con un 60 %; y el peor es K-Means, con 46 %. En este caso hay algo más de amplitud entre el valor mínimo y el máximo, pero de igual manera todos los resultados de rendimiento se encuentran muy cercanos al 50 %, por lo que no se pueden considerar como clasificadores realmente eficaces.

Capítulo 5

Conclusiones y Trabajo Futuro

Una vez expuesta toda la implementación, realizadas todas las pruebas, y presentado los resultados obtenidos, en este capítulo se recopila toda la información resultante para hallar las conclusiones pertinentes. Además, se presentan las posibles líneas de trabajo futuro que continúen la labor de este estudio.

5.1. Conclusiones

En este trabajo se ha implementado un sistema de detección de anomalías en redes utilizando diferentes algoritmos de aprendizaje automático, para comparar los resultados de cada uno de ellos e intentar decidir cuál resulta más útil en este propósito. Para ello se han utilizado a su vez dos datasets distintos, el CICIDS2017 y el UNSW-NB15, cada uno con una composición distinta y con una batería de ataques muy variada en ambos casos. De entre los 16 modelos de aprendizaje automático usados, 12 de ellos han sido de tipo supervisado, y 4 no supervisados.

La implementación del sistema de detección de anomalías se ha hecho a su vez desde dos perspectivas, una separando por cada tipo de ataque y otra usando el conjunto total de datos. En este último caso, las pruebas se han realizado con dos configuraciones de parámetros distintas, para poder así comprobar también cuál de ellas se comporta mejor.

En términos generales, y para ambos datasets, los algoritmos supervisados que mejor desempeño han mostrado a la hora de detectar los ataques son Random Forest, ID3, AdaBoost, KNN y Gradient Boost Classifier. En la inmensa mayoría de casos, para ambos datasets, y para todos los enfoques que se han probado, los resultados de rendimiento han sido superiores al 90 %. Estos datos son realmente buenos ya que muestran un nivel de efectividad muy alto en la detección de ataques. Otros algoritmos han mostrado un comportamiento más irregular, y dependiendo del enfoque o de los parámetros utilizados han proporcionado resultados de una gran amplitud, desde valores inferiores al 50 % hasta cercanos al 95 %. Es el caso de Gaussian Naïve Bayes, Logistic Regression, Support Vector Machine o Multi-Layer Perceptron.

Tal y como se ha destacado en el capítulo de Resultados y Evaluación, al realizar las pruebas separadamente para cada ataque, para algunos de ellos (FTP-Patator, en el caso de CICIDS2017, o Reconnaissance en UNSW-NB15) los resultados han sido extraordinariamente buenos con todos los algoritmos; mientras que no ha habido ningún ataque que haya presentado malos resultados en absolutamente todos los modelos supervisados. Esto es una buena noticia, ya que se entiende que cualquier tipo ataque tratado en este estudio podrá ser detectado al menos por un algoritmo supervisado de los que se han implementado aquí.

Comparando los resultados obtenidos para cada dataset, en general han sido mejores para el caso de UNSW-NB15 que para CICIDS2017. Si bien para ambos conjuntos de datos se han dado casos con resultados tanto inferiores al 50 % como del 100 %, en el caso del UNSW-NB15 los rendimientos de clasificación obtenidos con los modelos supervisados han sido ligeramente superiores que los recogidos con CICIDS2017. Pese a que la estrategia de implementación ha sido idéntica para ambos datasets, no es extraño que se den estas diferencias ya que cada dataset era notablemente distinto, con parámetros propios, distribuciones diferentes, y tipos de ataques también distintos.

En cuanto al enfoque realizado utilizando la totalidad del dataset, los resultados obtenidos con las diferentes configuraciones de parámetros han resultado muy similares. Tanto con CICIDS2017 como con UNSW-NB15, al construir los modelos a partir del dataset completo, los valores de rendimiento que han mostrado los distintos algoritmos han sido casi idénticos para el caso en el que se usaban 7 parámetros y en el que se tomaban 12 y 18 parámetros, respectivamente. Puesto que al utilizar un mayor número de parámetros las ejecuciones resultan sensiblemente más costosas, concluimos que es mucho más conveniente usar una selección de parámetros reducida que recoja la mayor importancia posible en vez de tomar una gran lista de parámetros.

En cuanto a los modelos no supervisados, es evidente que los resultados de rendimiento en la clasificación han sido sensiblemente peores que para el caso supervisado. Esto no resulta sorprendente ni preocupante ya que los modelos no supervisados no cuentan con la "ventaja"de conocer las etiquetas reales de los datos, como sí ocurre con los supervisados. Puede verse que tanto con el dataset CICIDS2017 como con UNSW-NB15, al trabajar sobre el conjunto total de datos, los valores alcanzados por los distintos modelos no supervisados rondan el 50 % de efectividad, llegando como máximo a un 64 % y en muchos casos presentando valores por debajo del 50 %. Distinguiendo por cada tipo de ataque, de nuevo se dan muchos valores de rendimiento por debajo del umbral del 50 %, pero en este caso también aparecen numerosos registros superiores al 80 %, incluso llegando hasta el 100 %.

Pese a sus resultados aparentemente bajos, este tipo de algoritmos resultan, aun así, más útiles que los de tipo supervisado, ya que en casos reales es difícil disponer de datos correctamente etiquetados de atemano para entrenar los modelos. Por tanto, son los algoritmos no supervisados los que en muchas ocasiones se utilizan para este tipo de cometidos, ya que no necesitan disponer previamente de una información tan detallada. Además, el hecho de que para algunos tipos de ataque los resultados de rendimiento obtenidos hayan sido superiores al 80 % e incluso al 90 %, muestran que la capacidad de estos clasificadores no está limitada al entorno del 50 %. Esto resulta

esperanzador ya que permite intuir que con distintos enfoques o afinando la construcción de los modelos se pueden conseguir sistemas de detección de anomalías no supervisados con una efectividad de clasificación realmente alta.

Otra conclusión importante se deriva del análisis realizado en el apartado 4.1.3, en el que se comparan los resultados obtenidos en este estudio con los de otro trabajo similar [3] que utiliza también el dataset CICIDS2017 y algunos algoritmos implementados aquí. Al realizar dicha comparación, se puede ver que, utilizando los mismos datos y los mismos algoritmos, dependiendo de qué precisiones se tomen los resultados pueden variar. Construir los modelos con ciertos hiperparámetros o preprocesar los datos de una determinada forma puede dar lugar a notables diferencias en el rendimiento del sistema construido. Viendo esto se puede concluir que, realizando un estudio más preciso de todas estas posibilidades y ajustándolas según convenga, sería posible obtener unos mejores resultados de rendimiento.

5.2. Trabajo futuro

Habiendo expuesto las conclusiones obtenidas a partir de todo el trabajo realizado, a continuación vamos a detallar algunas posibles líneas de trabajo futuro que continúen la labor realizada en este estudio:

- **Utilizar datasets distintos y más actuales:** Si bien los datasets utilizados en este estudio son completos y no están desactualizados, usar otros conjuntos de datos más modernos y amplios puede resultar muy interesante. Como se comentó en la Sección 2.2, existen en la actualidad datasets más recientes, con enfoques muy variados y una gama de ataques mayor, que pueden resultar muy útiles en la detección de anomalías. Si se dispone de máquinas con capacidad de cómputo suficiente, realizar pruebas similares a las hechas en este estudio con estos datasets más grandes y completos puede aportar resultados realmente útiles.
- **Ampliar la batería de algoritmos utilizados:** En este estudio se han realizado las pruebas con 16 algoritmos distintos, 12 supervisados y 4 no supervisados. Aunque es un buen número, existen muchos otros modelos de aprendizaje automático y estadísticos que pueden utilizarse en problemas de clasificación como el que nos ocupa. Algoritmos más recientes o que se adapten mejor a las necesidades del problema pueden ser interesantes de probar y comparar con el resto. De entre todos, los algoritmos no supervisados resultan especialmente interesantes ya que son de lo más útil frente a ataques reales, tanto conocidos como no. Por tanto, ampliar la gama de modelos no supervisados parece buena idea, teniendo en cuenta que en este trabajo sólo se han utilizado cuatro.
- **Estudiar los hiperparámetros de los modelos:** Como se ha podido ver en las fases de implementación y análisis de los resultados, modificar los hiperparámetros de los modelos de aprendizaje automático usados redunda en los resultados de rendimiento. Realizar un estudio en profundidad acerca de estos hiperparámetros y el efecto que tiene modificarlos en la detección de anomalías resultaría

enormemente útil a la hora de construir modelos mucho más precisos. Una posibilidad en esta línea es realizar barridos de hiperparámetros para los distintos algoritmos, buscando una configuración óptima que maximice el rendimiento de la clasificación (por ejemplo, con la función GridSearchCV de Sklearn).

- **Modificar la selección de parámetros:** De entre todos los parámetros disponibles en los datos, se han hecho varias selecciones para utilizarlas en la fase de implementación. En este caso el criterio para escoger unos u otros ha sido el valor de importancia que ha marcado el algoritmo Random Forest Regressor, y se han probado distintas configuraciones. Utilizar otros algoritmos, métodos o criterios de selección puede ser interesante para comparar distintas configuraciones de parámetros. Además, si se dispone de la suficiente capacidad de cómputo, se pueden utilizar muchas más variables en la implementación sin que esto suponga una limitación, como es el caso en este trabajo.

- **Preprocesar los datos de distintas formas:** Tal y como se ha podido ver en el análisis de resultados, el tratamiento previo de los datos tiene consecuencias directas en los resultados de rendimiento finales. La normalización o estandarización de los datos puede ser útil con algunos algoritmos y resultar contraproducente con otros. Estudiar qué resulta más conveniente en cada caso para optimizar la precisión de la clasificación puede proporcionar mejores resultados. Se puede también incidir en la transformación de variables categóricas a numéricas, paso imprescindible ya que los algoritmos trabajan con números, pero que dependiendo de cómo se haga se puede perder información importante en el proceso.

- **Extender el estudio a otros campos de aplicación:** Como ya se ha comentado en el apartado del Estado de la Cuestión, la detección de anomalías puede utilizarse en multitud de ámbitos. Una implementación similar a la de este trabajo, aplicada a otros campos de estudio, puede resultar realmente útil. La detección de anomalías en logs del sistema, en el uso de recursos (CPU, memoria, disco) por parte de los procesos o la predicción de fallos son algunos ejemplos de posibles aplicaciones de la detección de anomalías. Incluso otros sectores ajenos a la informática como la medicina, la industria, la sociología o las finanzas pueden verse beneficiadas de herramientas basadas en la detección de anomalías.

Introduction

5.3. Motivation

In the increasingly globalized and interconnected world we live in, the importance of the Internet is growing every day. Millions of users worldwide connect to the network daily to communicate with each other or with companies and institutions of all kinds. Over the last two decades, the number of Internet users globally has multiplied extraordinarily, surpassing 5.4 billion users in 2023, which accounts for 69% of the world's population [26].

In the face of this unstoppable reality, it can also be observed how, parallel to the growth of the Internet, the number of attacks on the network has multiplied, whether targeting individual users or corporations of any kind. These attacks can pose a significant risk in terms of economics, privacy, security, or even geopolitics.

In a global landscape where the vast majority of institutions and companies worldwide operate digitally, where the handling of massive data is becoming increasingly important, and where the emergence of new technologies in the field of computing is transforming reality, it is increasingly critical to defend against such cyber-attacks.

When it comes to preventing such attacks, the first line of defense for any network or system is usually the firewall, which performs an initial filter on incoming or outgoing traffic, discarding possible threats. However, many attacks manage to bypass the firewall, and this is where the second level of defense comes into play: the Intrusion Detection System (IDS). This system is based on the assumption that the behavior of the intruder differs quantifiably from that of a legitimate user. Thus, it can identify the traffic of an attacker, and if the intrusion is detected quickly enough, the threat can be identified and neutralized before any damage is done.

Currently, there are fundamentally two types of IDS, those based on rules and those based on anomaly detection.

Rule-based detection, also called signature detection, uses a database in which a set of rules or attack patterns (signatures) is defined to determine whether a given behavior is an intrusion or not. This method is quite effective, but it has the disadvantage that databases need to be constantly updated and process new information about attacks. Additionally, even if the databases are updated, they remain vulnerable to attacks of new types that have not been seen before. Since these attacks are not in

the database, they cannot be prevented. In this method, very specific rules of systems and attacks need to be described, requiring the detection of the specific vulnerability and not necessarily the attack, which can be complicated. The result is often a large number of rules to process, which may be inefficient when analyzing large amounts of traffic. Moreover, since a significant portion of current traffic travels encrypted and its content cannot be directly seen, it is particularly difficult for these systems to detect attacks among this type of traffic.

On the other hand, anomaly-based intrusion detection systems detect unusual behavior by examining traffic on the network, comparing it with the behavior of legitimate users. They detect deviations from this normal behavior, which could be attacks, by applying statistical tests. Nowadays, thanks to the advancement of machine learning algorithms, this methodology is advancing considerably, allowing for the automation of previously costly processes and obtaining very promising results. One of the advantages of anomaly detection is that it can indeed detect attacks never seen before, as it can compare them equally with normal traffic. Additionally, since these systems analyze general properties such as size or connection time, they can detect encrypted messages without needing to read their content. However, one of the major problems with anomaly detection is that large amounts of data, both normal traffic and attacks, are often needed to "train" the detection system, and they are not always easy to obtain. In any case, these IDS are increasingly used in attack detection and prevention.

Given the advantages of anomaly-based IDS, this work will study this type of systems in detail, specifically those that use machine learning algorithms. Furthermore, they will be tested with real traffic data using public datasets, and then their performance will be collected and analyzed.

5.4. Objectives

This work follows the path set by K. Kostas in his study "Anomaly Detection in Networks Using Machine Learning" [3], where various machine learning algorithms are studied and then applied to anomaly detection on a dataset, followed by an analysis of the results. The aim of this work is to expand on K. Kostas's study in multiple directions, with the goal of obtaining broader and more comprehensive results. Considering this, the objectives to be achieved by the end of this work are as follows:

- **Explore and select datasets for training the IDS.** Various available datasets will be explored, and two of them will be selected to train the intrusion detection system. Among all, those that seem most interesting, considering their characteristics and the objective of this study, will be chosen. Additionally, an effort will be made to ensure the datasets relate to other similar works (such as K. Kostas [3]) to compare and expand the results of those studies with this one.
- **Examine machine learning algorithms for anomaly detection.** Among all the algorithms, those used in K. Kostas's work [3] and others that appear promising or interesting for comparison will be selected. The idea here is to signifi-

cantly increase the number and variety of algorithms used, allowing for a broader comparison. In particular, both supervised and unsupervised algorithms will be used, enabling a very interesting comparison between these two approaches in machine learning.

- **Apply the selected machine learning algorithms.** Code will be implemented to detect attacks using both datasets and collect the results. This will be done, for both datasets, from two different approaches: separately for each type of attack present in the data, and on the entire dataset without distinguishing the type of attack. Additionally, various traffic parameter selections will be studied and the results compared for each of them. The code implemented in this work will be made publicly available along with the report through this GitHub repository.

- **Study the results obtained during the evaluation phase.** Once the anomaly detection models are built, they will be evaluated by collecting performance results of the classification. Given the variety of algorithms used, data, and types of attacks, a wealth of interesting output data will be obtained, which will be thoroughly studied and compared.

5.5. Workplan

Before beginning the specific development of the work, the most important general concepts about anomaly detection systems will be studied. This will provide fundamental prior knowledge for understanding and approaching the subsequent work.

Continuing with the preliminary documentation, the various datasets available for anomaly detection will be explored, studying and comparing the characteristics of each. In addition to the dataset used in K. Kostas's work [3], the CICIDS2017 dataset, another dataset will be chosen from among those analyzed based on its technical characteristics. This will allow for analogous work to be conducted with both datasets, enabling the combination and comparison of the results obtained, thereby providing a wide variety of data and attacks in the process.

Furthermore, available machine learning algorithms that can be used for anomaly detection will be explored and studied. Several of these algorithms will be selected for subsequent implementation within the IDS. In addition to those used in K. Kostas's work [3], promising or interesting algorithms will be selected for comparison, including unsupervised learning algorithms in this case. The idea here is to significantly increase the number and variety of algorithms used, allowing for a broader comparison.

Once the fundamental aspects of anomaly detection are understood, the final documentation task will involve examining previously published works in this field, with special emphasis on the most current ones. Since, as explained in the previous section, this work is largely an extension of the one carried out by K. Kostas [3], documentation surrounding this work will receive considerable attention.

Once all the necessary prior information has been studied and collected, the im-

plementation phase will commence. In this phase, both datasets will be analyzed and preprocessed, followed by the development of multiple Python programs that implement the anomaly detection system using the previously selected algorithms. These programs will read the records from the dataset, which will be used to train the detection model. Once the model is prepared, its performance will be tested by classifying traffic from the same dataset, and the performance results obtained will be collected.

Finally, the collected results from running the models will be compiled and analyzed to compare the different algorithms applied to each dataset. This will allow for comparative results to be concluded, shedding light on which models perform better or worse depending on the parameters, attacks, or data.

Conclusions and Future Work

Once all the implementation has been outlined, all tests have been conducted, and the results obtained have been presented, this chapter compiles all resulting information to draw relevant conclusions. Additionally, possible avenues for future work that continue the efforts of this study are presented.

5.6. Conclusion

In this work, an anomaly detection system in networks has been implemented using different machine learning algorithms to compare the results of each of them and attempt to decide which one is most useful for this purpose. To do this, two different datasets, CICIDS2017 and UNSW-NB15, have been used, each with a different composition and a varied range of attacks in both cases. Among the 16 machine learning models used, 12 of them have been supervised, and 4 unsupervised.

The implementation of the anomaly detection system has been carried out from two perspectives: one separating by each type of attack and the other using the total data set. In the latter case, tests have been performed with two different parameter configurations to also verify which one performs better.

In general terms, and for both datasets, the supervised algorithms that have shown the best performance in detecting attacks are Random Forest, ID3, AdaBoost, KNN, and Gradient Boost Classifier. In the vast majority of cases, for both datasets and for all approaches that have been tested, the performance results have been above 90%. These data are really good as they show a very high level of effectiveness in detecting attacks. Other algorithms have shown more irregular behavior, and depending on the approach or parameters used, they have provided results with a wide range, from values below 50% to close to 95%. This is the case for Gaussian Naïve Bayes, Logistic Regression, Support Vector Machine, or Multi-Layer Perceptron.

As highlighted in the Results and Evaluation chapter, when conducting tests separately for each attack, for some of them (such as FTP-Patator in the case of CICIDS2017 or Reconnaissance in UNSW-NB15), the results have been extraordinarily good with all algorithms. There hasn't been any attack that has performed poorly in absolutely all supervised models. This is good news, as it suggests that any type of attack addressed in this study can be detected by at least one supervised algorithm implemented here.

Comparing the results obtained for each dataset, they have generally been better for UNSW-NB15 than for CICIDS2017. Although for both datasets there have been cases with results both below 50% and 100%, in the case of UNSW-NB15, the classification performances obtained with the supervised models have been slightly higher than those collected with CICIDS2017. Despite the identical implementation strategy for both datasets, these differences are not surprising, as each dataset was notably different, with its own parameters, different distributions, and also different types of attacks.

Regarding the approach using the entire dataset, the results obtained with the different parameter configurations have been very similar. Both with CICIDS2017 and UNSW-NB15, when building the models from the complete dataset, the performance values shown by the different algorithms have been almost identical for the case of using 7 parameters and when taking 12 and 18 parameters, respectively. Since using a larger number of parameters makes the executions significantly more expensive, we conclude that it is much more convenient to use a reduced parameter selection that captures the greatest importance possible rather than taking a large list of parameters.

As for the unsupervised models, it is evident that the classification performance results have been significantly worse than for the supervised case. This is not surprising or concerning since unsupervised models do not have the "advantage" of knowing the real labels of the data, as supervised models do. It can be seen that both with the CICIDS2017 dataset and UNSW-NB15, when working on the entire dataset, the values reached by the different unsupervised models hover around 50% effectiveness, reaching a maximum of 64% and in many cases presenting values below 50%. Distinguishing by each type of attack, again, many performance values below the 50% threshold are observed, but in this case, numerous records above 80% also appear, even reaching 100%.

Despite their seemingly low results, these types of algorithms are still more useful than supervised ones because in real cases, it's difficult to have correctly labeled data in advance to train the models. Therefore, unsupervised algorithms are often used for these tasks since they do not require such detailed information beforehand. Additionally, the fact that for some types of attacks, the performance results obtained have been above 80% and even 90%, shows that the capability of these classifiers is not limited to around 50%. This is promising as it suggests that with different approaches or refining model construction, unsupervised anomaly detection systems with really high classification effectiveness can be achieved.

Another important conclusion is derived from the analysis conducted in Section 4.1.3, where the results obtained in this study are compared with those of another similar work [3] that also uses the CICIDS2017 dataset and some algorithms implemented here. When making this comparison, it can be seen that, using the same data and algorithms, depending on the precisions taken, the results can vary. Constructing the models with certain hyperparameters or preprocessing the data in a certain way can lead to notable differences in the performance of the constructed system. From this, it can be concluded that by conducting a more precise study of all these possibilities and adjusting them accordingly, it would be possible to obtain better performance results.

5.7. Future work

Having presented the conclusions drawn from all the work conducted, we will now detail some possible lines of future work that continue the efforts made in this study:

- **Utilizing different and more current datasets:** While the datasets used in this study are comprehensive and not outdated, using other more modern and extensive datasets can be very interesting. As mentioned in Section 2.2, there are currently more recent datasets with various approaches and a wider range of attacks, which can be very useful in anomaly detection. If machines with sufficient computing power are available, conducting similar tests to those carried out in this study with these larger and more comprehensive datasets can yield truly useful results.
- **Expanding the range of algorithms used:** In this study, tests were conducted with 16 different algorithms, 12 supervised and 4 unsupervised. Although this is a good number, there are many other machine learning and statistical models that can be used in classification problems like the one at hand. Newer algorithms or those that better suit the needs of the problem may be interesting to try and compare with the rest. Among all, unsupervised algorithms are particularly interesting as they are most useful against real attacks, both known and unknown. Therefore, expanding the range of unsupervised models seems like a good idea, considering that only four have been used in this work.
- **Studying model hyperparameters:** As seen in the implementation and results analysis phases, modifying the hyperparameters of the machine learning models used affects performance results. Conducting an in-depth study of these hyperparameters and the effect of modifying them on anomaly detection would be enormously useful in building much more precise models. One possibility in this line is to perform hyperparameter sweeps for the different algorithms, seeking an optimal configuration that maximizes classification performance (for example, using the GridSearchCV function from Sklearn).
- **Modifying parameter selection:** Among all the parameters available in the data, several selections have been made to use them in the implementation phase. In this case, the criterion for choosing one or the other has been the importance value marked by the Random Forest Regressor algorithm, and different configurations have been tested. Using other algorithms, methods, or selection criteria may be interesting to compare different parameter configurations. Additionally, if sufficient computing power is available, many more variables can be used in the implementation without limitation, as is the case in this work.
- **Preprocessing data in different ways:** As seen in the results analysis, preprocessing data has direct consequences on the final performance results. Normalization or standardization of data may be useful with some algorithms and counterproductive with others. Studying what is most convenient in each case to optimize classification accuracy can provide better results. Attention can also

be focused on transforming categorical variables into numeric ones, an essential step since algorithms work with numbers, but depending on how it is done, important information may be lost in the process.

- **Extending the study to other fields of application:** As already mentioned in the State of the Art section, anomaly detection can be utilized in numerous domains. An implementation similar to that of this work, applied to other fields of study, can be highly beneficial. Anomaly detection in system logs, resource usage (CPU, memory, disk) by processes, or failure prediction are some examples of potential applications for anomaly detection. Even sectors outside of IT, such as medicine, industry, sociology, or finance, can benefit from tools based on anomaly detection.

Bibliografía

[1] Scikit-learn: Machine learning in python. https://scikit-learn.org/stable/.

[2] P. Amangele, M. J. Reed, M. Al-Naday, N. Thomos, and M. Nowak. Hierarchical machine learning for iot anomaly detection in sdn. *International Conference on Information Technologies*, 2019.

[3] K Costas. Anomaly detection in networks using machine learning. *Heriott-Watt University*, 2018.

[4] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman. 1998 darpa intrusion detection evaluation dataset. *MIT Lincoln Laboratory*, 1998.

[5] R. Garreta and G. Monececchi. *Learning scikit-learn: Machine Learning in Python*. Pack Publishing, 2013.

[6] D. Hawkins. Identification of outliers (monographs on statistics and applied probability). *Springer*, 1980.

[7] Kate Highnam, Kai Arulkumaran, Zachary Hanif, and Nicholas R. Jennings. Beth dataset: Real cybersecurity data for anomaly detection research. 2021.

[8] M. Idhammad, K. Afdel, and M. Belouch. Semi-supervised machine learning approach for ddos detection. *Applied Intelligence*, 2018.

[9] MIT Lincoln Laboratory. 1998 darpa intrusion detection evaluation dataset. https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset.

[10] University College London. Real cybersecurity data for anomaly detection research. http://www.gatsby.ucl.ac.uk/~balaji/udl2021/accepted-papers/UDL2021-paper-033.pdf.

[11] Nour Moustafa. Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets. *IEEE Internet of Things Journal*, 2021.

[12] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *Military Communications and Information Systems Conference (MilCIS)*, 2015.

[13] Mukrimah Nawir, Amiza Amir, Naimah Yaakob, and Ong Bi Lynn. Effective and efficient network anomaly detection system using machine learning algorithm. *Bulletin of Electrical Engineering and Informatics*, 2019.

[14] Coburg University of Applied Sciences. Cidds - coburg intrusion detection data sets. `https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html`.

[15] University of New Brunswick. Cse-cic-ids2018 on aws. `https://www.unb.ca/cic/datasets/ids-2018.html`.

[16] University of New Brunswick. Intrusion detection evaluation dataset (cic-ids2017). `https://www.unb.ca/cic/datasets/ids-2017.html`.

[17] University of New Brunswick. Intrusion detection evaluation dataset (isc-xids2012). `https://www.unb.ca/cic/datasets/ids.html`.

[18] University of New Brunswick. Iscx nsl-kdd dataset 2009. `https://www.unb.ca/cic/datasets/nsl.html`.

[19] Daniel Pérez, Serafín Alonso, Antonio Morán, Miguel A. Prada, Juan José Fuertes, and Manuel Domínguez. Evaluation of feature learning for anomaly detection in network trafc. *Springer Nature*, 2020.

[20] Markus Ring, Sarah Wunderlich, Dominik Grüdl, Dieter Landes, and Andreas Hotho. Flow-based benchmark data sets for intrusion detection. *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)*, 2017.

[21] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2016.

[22] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani. Towards a reliable intrusion detection benchmark dataset. *Software Networking*, 2017.

[23] 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)*, 2018.

[24] 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)*, 2018.

[25] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers Security*, 2012.

[26] Internet World Stats. Internet growth statistics. Technical report, Internet World Stats, 2023.

[27] Salvatore J. Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, , and Philip K. Chan. Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection: Results from the jam project. *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, 1999.

[28] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani. A detailed analysis of the kdd cup 99 data set. *Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 2009.

[29] Irvine University of California. Kdd cup 1999 data. `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[30] Australia UNSW Canberra. The ton_iot datasets. `https://research.unsw.edu.au/projects/toniot-datasets`.

[31] Australia UNSW Canberra. The unsw-nb15 dataset. `https://research.unsw.edu.au/projects/unsw-nb15-dataset`.

Apéndice A

Lista de parámetros para cada dataset

En este apéndice se incluyen las listas de los parámetros de tráfico recogidos en cada dataset, junto con una breve descripción de cada uno.

A.1. CICIDS2017

№	Feature Name	Feature Description
1	Flow ID	Flow ID
2	Source IP	Source IP
3	Source Port	Source Port
4	Destination IP	Destination IP
5	Destination Port	Destination Port
6	Protocol	Protocol
7	Timestamp	Timestamp
8	Flow Duration	Duration of the flow in Microsecond
9	Total Fwd Packets	Total packets in the forward direction
10	Total Backward Packets	Total packets in the backward direction
11	Total Length of Fwd Packets	Total size of packet in forward direction
12	Total Length of Bwd Packets	Total size of packet in backward direction
13	Fwd Packet Length Max	Maximum size of packet in forward direction
14	Fwd Packet Length Min	Minimum size of packet in forward direction
15	Fwd Packet Length Mean	Mean size of packet in forward direction
16	Fwd Packet Length Std	Standard deviation size of packet in forward direction
17	Bwd Packet Length Max	Maximum size of packet in backward direction
18	Bwd Packet Length Min	Minimum size of packet in backward direction
19	Bwd Packet Length Mean	Mean size of packet in backward direction
20	Bwd Packet Length Std	Standard deviation size of packet in backward direction
21	Flow Bytes/s	Number of flow bytes per second
22	Flow Packets/s	Number of flow packets per second
23	Flow IAT Mean	Mean length of a flow
24	Flow IAT Std	Standard deviation length of a flow
25	Flow IAT Max	Maximum length of a flow
26	Flow IAT Min	Minimum length of a flow
27	Fwd IAT Total	Total time between two packets sent in the forward direction
28	Fwd IAT Mean	Mean time between two packets sent in the forward direction
29	Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
30	Fwd IAT Max	Maximum time between two packets sent in the forward direction
31	Fwd IAT Min	Minimum time between two packets sent in the forward direction
32	Bwd IAT Total	Total time between two packets sent in the backward direction
33	Bwd IAT Mean	Mean time between two packets sent in the backward direction
34	Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
35	Bwd IAT Max	Maximum time between two packets sent in the backward direction
36	Bwd IAT Min	Minimum time between two packets sent in the backward direction
37	Fwd PSH Flags	Number of packets with PUSH
38	Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
39	Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
40	Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
41	Fwd Header Length	Total bytes used for headers in the forward direction
42	Bwd Header Length	Total bytes used for headers in the backrward direction
43	Fwd Packets/s	Number of forward packets per second
44	Bwd Packets/s	Number of backward packets per second

Figura A.1: Detalle de los parámetros recogidos en el dataset CICIDS2017 (I)

№	Feature Name	Feature Description
45	Min Packet Length	Minimum inter-arrival time of packet
46	Max Packet Length	Maximum inter-arrival time of packet
47	Packet Length Mean	Mean inter-arrival time of packet
48	Packet Length Std	Standard deviation inter-arrival time of packet
49	Packet Length Variance	Packet Length Variance
50	FIN Flag Count	Number of packets with FIN
51	SYN Flag Count	Number of packets with SYN
52	RST Flag Count	Number of packets with RST
53	PSH Flag Count	Number of packets with PUSH
54	ACK Flag Count	Number of packets with ACK
55	URG Flag Count	Number of packets with URG
56	CWE Flag Count	Number of packets with CWE
57	ECE Flag Count	Number of packets with ECE
58	Down/Up Ratio	Download and upload ratio
59	Average Packet Size	Average size of packet
60	Avg Fwd Segment Size	Average size observed in the forward direction
61	Avg Bwd Segment Size	Average size observed in the backward direction
62	Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
63	Fwd Avg Packets/Bulk	Average number of packets bulk rate in the forward direction
64	Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
65	Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
66	Bwd Avg Packets/Bulk	Average number of packets bulk rate in the backward direction
67	Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
68	Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
69	Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
70	Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
71	Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
72	Init_Win_bytes_forward	The total number of bytes sent in initial window in the forward direction
73	Init_Win_bytes_backward	The total number of bytes sent in initial window in the backward direction
74	act_data_pkt_fwd	Count of packets with at least 1 byte of TCP data payload in the forward direction
75	min_seg_size_forward	Minimum segment size observed in the forward direction
76	Active Mean	Mean time a flow was active before becoming idle
77	Active Std	Standard deviation time a flow was active before becoming idle
78	Active Max	Maximum time a flow was active before becoming idle
79	Active Min	Minimum time a flow was active before becoming idle
80	Idle Mean	Mean time a flow was idle before becoming active
81	Idle Std	Standard deviation time a flow was idle before becoming active
82	Idle Max	Maximum time a flow was idle before becoming active
83	Idle Min	Minimum time a flow was idle before becoming active
84	Label	Label
85	External IP	External IP

Figura A.2: Detalle de los parámetros recogidos en el dataset CICIDS2017 (II)

(Imagen tomada del trabajo *Anomaly Detection in Networks Using Machine Learning*, de Kahraman Kostas)

A.2. UNSW-NB15

SNo.	Name	Type	Description
36	is_sm_ips_ports	Binary	"If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal thenthis variable takes value 1 else 0"
39	is_ftp_login		If the ftp session is accessed by user and password then 1 else 0.
49	Label		0 for normal and 1 for attack records
7	dur	Float	Record total duration
15	Sload		Source bits per second
16	Dload		Destination bits per second
27	Sjit		Source jitter (mSec)
28	Djit		Destination jitter (mSec)
31	Sintpkt		Source interpacket arrival time (mSec)
32	Dintpkt		Destination interpacket arrival time (mSec)
33	tcprtt		"TCP connection setup round-trip time, the sum of synack and ackdat."
34	synack		"TCP connection setup time the time between the SYN and the SYN_ACK packets."
35	ackdat		"TCP connection setup time the time between the SYN_ACK and the ACK packets."
2	sport	Integer	Source port number
4	dsport		Destination port number
8	sbytes		Source to destination transaction bytes
9	dbytes		Destination to source transaction bytes
10	sttl		Source to destination time to live value
11	dttl		Destination to source time to live value
12	sloss		Source packets retransmitted or dropped
13	dloss		Destination packets retransmitted or dropped
17	Spkts		Source to destination packet count
18	Dpkts		Destination to source packet count
19	swin		Source TCP window advertisement value
20	dwin		Destination TCP window advertisement value
21	stcpb		Source TCP base sequence number
22	dtcpb		Destination TCP base sequence number
23	smeansz		Mean of the ?ow packet size transmitted by the src
24	dmeansz		Mean of the ?ow packet size transmitted by the dst
25	trans_depth		Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len		Actual uncompressed content size of the data transferred from the servers http service.
37	ct_state_ttl		No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
38	ct_flw_http_mthd		No. of flows that has methods such as Get and Post in http service.
40	ct_ftp_cmd		No of flows that has a command in ftp session.
41	ct_srv_src		No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
42	ct_srv_dst		No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
43	ct_dst_ltm		No. of connections of the same destination address (3) in 100 connections according to the last time (26).
44	ct_src_ ltm		No. of connections of the same source address (1) in 100 connections according to the last time (26).
45	ct_src_dport_ltm		No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
46	ct_dst_sport_ltm		No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
47	ct_dst_src_ltm		No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).
1	srcip	Nominal	Source IP address
3	dstip		Destination IP address
5	proto		Transaction protocol
6	state		Indicates to the state and its dependent protocol
14	service		"http ftp smtp ssh dns ftp-data
48	attack_cat		"The name of each attack category. In this data set nine categories *eg* Fuzzers Analysis Backdoors DOS exploits Generic Reconnaissance Shellcode and Worms"
29	Stime	Timestamp	record start time
30	Ltime		record last time

Figura A.3: Detalle de los parámetros recogidos en el dataset UNSW-NB15

(Imagen tomada del trabajo *Benchmark Datasets for Network Intrusion Detection: A Review*, de Yasir Hamid, V. R. Balasaraswathi, Ludovic Journaux y M. Sugumaran)

Apéndice B

Importancia de los parámetros por cada tipo de ataque

En este apéndice se incluyen los valores de importancia para cada tipo de ataque en cada dataset, resultantes de la ejecución del algoritmo Random Forest Regressor. Se incluye también, para cada ataque, una representación gráfica de dichos valores de importancia de los parámetros. Estos resultados son los que se utilizarán para la selección de parámetros en cada enfoque de la implementación.

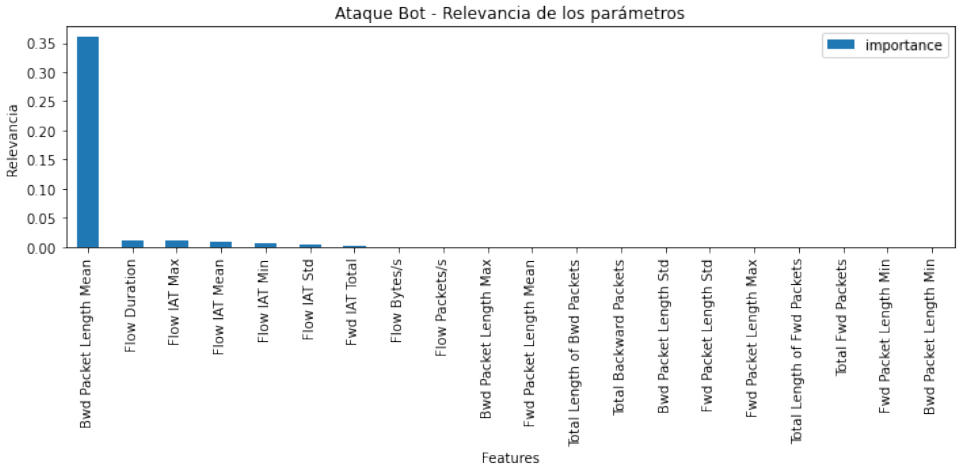

Figura B.1: Importancia de los parámetros para el ataque Bot (CICIDS2017)

Parámetro	**Importancia**
Bwd Packet Length Mean	0.360206
Flow Duration	0.011118
Flow IAT Max	0.010515
Flow IAT Mean	0.008170
Flow IAT Min	0.006642
Flow IAT Std	0.004260
Fwd IAT Total	0.001397
Flow Bytes/s	0.000804
Flow Packets/s	0.000666
Bwd Packet Length Max	0.000394
Fwd Packet Length Mean	0.000381
Total Length of Bwd Packets	0.000318
Total Backward Packets	0.000210
Bwd Packet Length Std	0.000122
Fwd Packet Length Std	0.000070
Fwd Packet Length Max	0.000055
Total Length of Fwd Packets	0.000055
Total Fwd Packets	0.000044
Fwd Packet Length Min	0.000010
Bwd Packet Length Min	0.000003

Tabla B.1: Importancia de los parámetros para el ataque Bot (CICIDS2017)

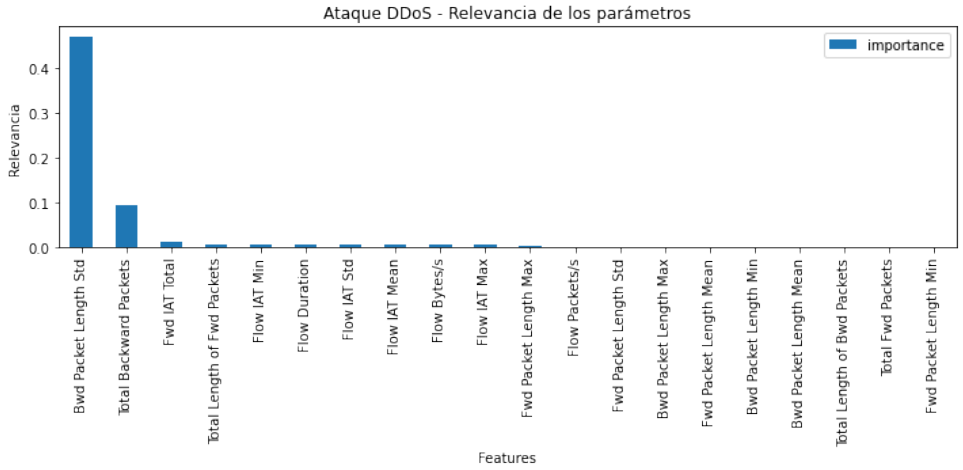

Figura B.2: Importancia de los parámetros para el ataque DDoS (CICIDS2017)

Parámetro	**Importancia**
Bwd Packet Length Std	0.470069
Total Backward Packets	0.093654
Fwd IAT Total	0.013253
Total Length of Fwd Packets	0.006444
Flow IAT Min	0.006028
Flow Duration	0.005928
Flow IAT Std	0.005522
Flow IAT Mean	0.005416
Flow Bytes/s	0.004743
Flow IAT Max	0.004642
Fwd Packet Length Max	0.002123
Flow Packets/s	0.001030
Fwd Packet Length Std	0.000728
Bwd Packet Length Max	0.000647
Fwd Packet Length Mean	0.000479
Bwd Packet Length Min	0.000449
Bwd Packet Length Mean	0.000194
Total Length of Bwd Packets	0.000099
Total Fwd Packets	0.000035
Fwd Packet Length Min	0.000028

Tabla B.2: Importancia de los parámetros para el ataque DDoS (CICIDS2017)

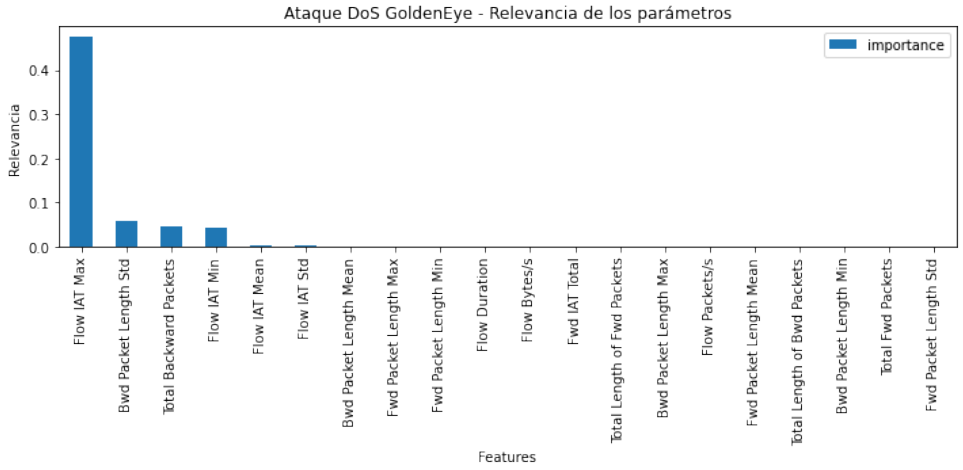

Figura B.3: Importancia de los parámetros para el ataque DoS-Goldeneye (CICIDS2017)

Parámetro	**Importancia**
Flow IAT Max	0.474878
Bwd Packet Length Std	0.058527
Total Backward Packets	0.046780
Flow IAT Min	0.043043
Flow IAT Mean	0.003036
Flow IAT Std	0.002148
Bwd Packet Length Mean	0.001270
Fwd Packet Length Max	0.001244
Fwd Packet Length Min	0.001199
Flow Duration	0.001182
Flow Bytes/s	0.000922
Fwd IAT Total	0.000871
Total Length of Fwd Packets	0.000442
Bwd Packet Length Max	0.000276
Flow Packets/s	0.000211
Fwd Packet Length Mean	0.000168
Total Length of Bwd Packets	0.000146
Bwd Packet Length Min	0.000050
Total Fwd Packets	0.000033
Fwd Packet Length Std	0.000010

Tabla B.3: Importancia de los parámetros para el ataque DoS-Goldeneye (CICIDS2017)

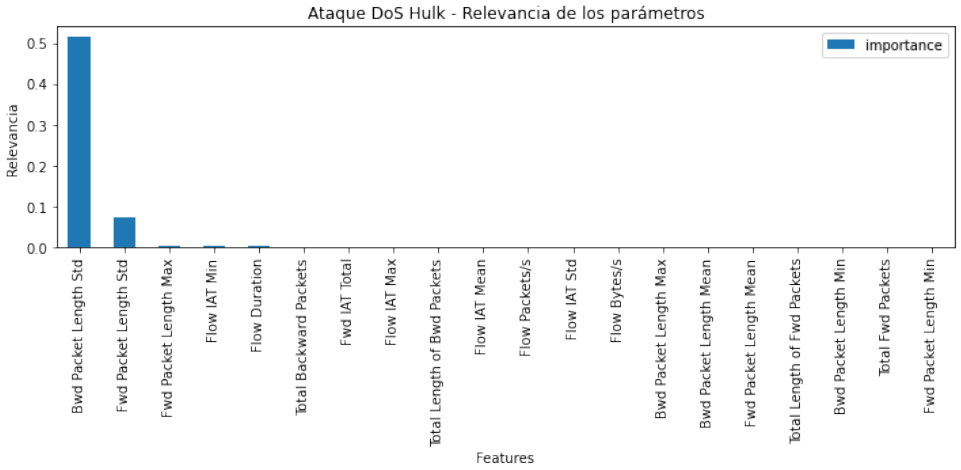

Figura B.4: Importancia de los parámetros para el ataque DoS-Hulk (CICIDS2017)

Parámetro	**Importancia**
Bwd Packet Length Std	5.146679e-01
Fwd Packet Length Std	7.388719e-02
Fwd Packet Length Max	4.256529e-03
Flow IAT Min	2.005605e-03
Flow Duration	1.638425e-03
Total Backward Packets	3.508506e-04
Fwd IAT Total	2.067530e-04
Flow IAT Max	1.372391e-04
Total Length of Bwd Packets	1.282891e-04
Flow IAT Mean	1.155876e-04
Flow Packets/s	9.161050e-05
Flow IAT Std	3.582877e-05
Flow Bytes/s	3.101402e-05
Bwd Packet Length Max	2.143281e-05
Bwd Packet Length Mean	1.869880e-05
Fwd Packet Length Mean	9.285329e-06
Total Length of Fwd Packets	9.093455e-06
Bwd Packet Length Min	6.424531e-06
Total Fwd Packets	4.946972e-06
Fwd Packet Length Min	3.084385e-07

Tabla B.4: Importancia de los parámetros para el ataque DoS-Hulk (CICIDS2017)

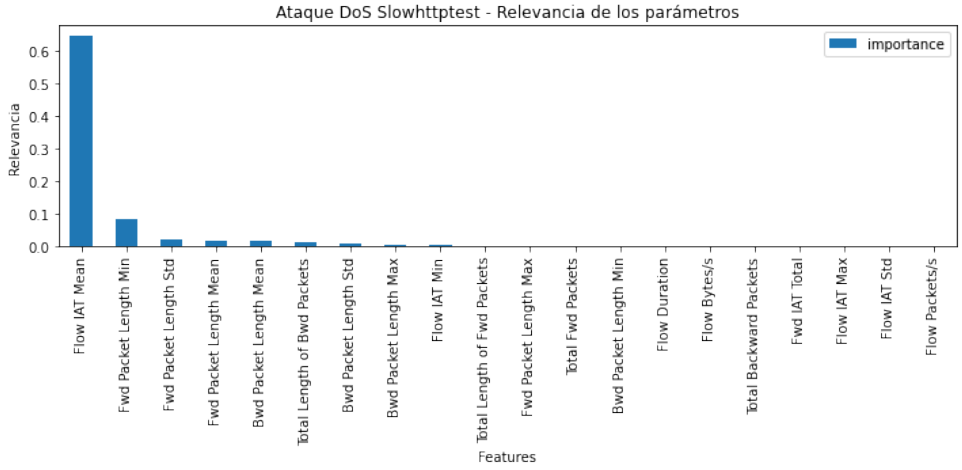

Figura B.5: Importancia de los parámetros para el ataque DoS-Slowhttptest (CI-CIDS2017)

Parámetro	**Importancia**
Flow IAT Mean	0.645387
Fwd Packet Length Min	0.082083
Fwd Packet Length Std	0.018954
Fwd Packet Length Mean	0.017246
Bwd Packet Length Mean	0.016647
Total Length of Bwd Packets	0.012343
Bwd Packet Length Std	0.009863
Bwd Packet Length Max	0.004496
Flow IAT Min	0.002060
Total Length of Fwd Packets	0.001039
Fwd Packet Length Max	0.000919
Total Fwd Packets	0.000772
Bwd Packet Length Min	0.000573
Flow Duration	0.000510
Flow Bytes/s	0.000474
Total Backward Packets	0.000417
Fwd IAT Total	0.000342
Flow IAT Max	0.000274
Flow IAT Std	0.000255
Flow Packets/s	0.000138

Tabla B.5: Importancia de los parámetros para el ataque DoS-Slowhttptest (CI-CIDS2017)

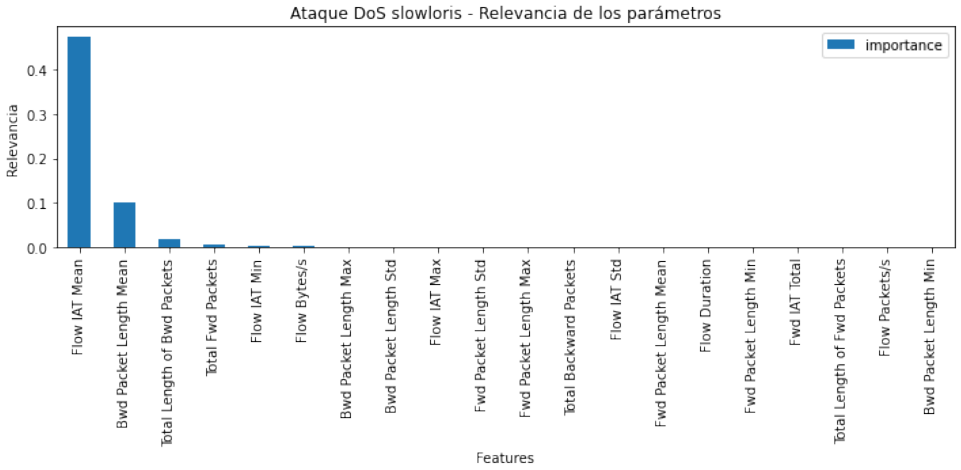

Figura B.6: Importancia de los parámetros para el ataque DoS-Slowloris (CICIDS2017)

Parámetro	**Importancia**
Flow IAT Mean	0.474426
Bwd Packet Length Mean	0.100013
Total Length of Bwd Packets	0.019652
Total Fwd Packets	0.006356
Flow IAT Min	0.002424
Flow Bytes/s	0.001758
Bwd Packet Length Max	0.000968
Bwd Packet Length Std	0.000884
Flow IAT Max	0.000807
Fwd Packet Length Std	0.000699
Fwd Packet Length Max	0.000694
Total Backward Packets	0.000666
Flow IAT Std	0.000539
Fwd Packet Length Mean	0.000433
Flow Duration	0.000319
Fwd Packet Length Min	0.000245
Fwd IAT Total	0.000226
Total Length of Fwd Packets	0.000182
Flow Packets/s	0.000135
Bwd Packet Length Min	0.000003

Tabla B.6: Importancia de los parámetros para el ataque DoS-Slowloris (CICIDS2017)

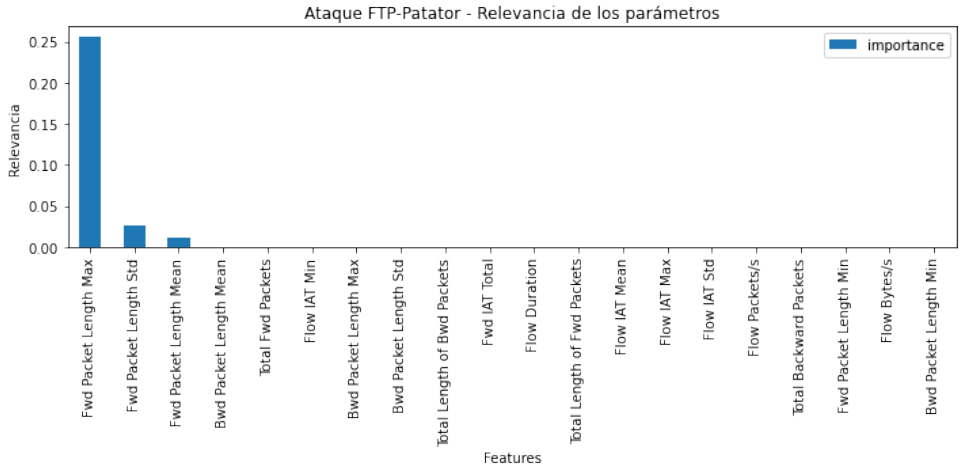

Figura B.7: Importancia de los parámetros para el ataque FTP-Patator (CICIDS2017)

Parámetro	**Importancia**
Fwd Packet Length Max	0.255835
Fwd Packet Length Std	0.027022
Fwd Packet Length Mean	0.010657
Bwd Packet Length Mean	0.000637
Total Fwd Packets	0.000458
Flow IAT Min	0.000294
Bwd Packet Length Max	0.000261
Bwd Packet Length Std	0.000191
Total Length of Bwd Packets	0.000154
Fwd IAT Total	0.000131
Flow Duration	0.000124
Total Length of Fwd Packets	0.000121
Flow IAT Mean	0.000094
Flow IAT Max	0.000079
Flow IAT Std	0.000068
Flow Packets/s	0.000047
Total Backward Packets	0.000043
Fwd Packet Length Min	0.000016
Flow Bytes/s	0.000007
Bwd Packet Length Min	0.000002

Tabla B.7: Importancia de los parámetros para el ataque FTP-Patator (CICIDS2017)

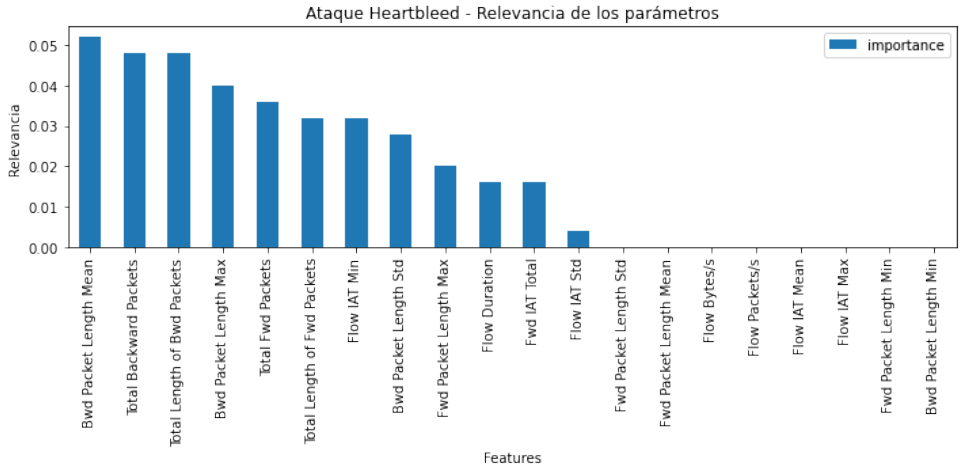

Figura B.8: Importancia de los parámetros para el ataque Heartbleed (CICIDS2017)

Parámetro	**Importancia**
Bwd Packet Length Mean	0.052
Total Backward Packets	0.048
Total Length of Bwd Packets	0.048
Bwd Packet Length Max	0.040
Total Fwd Packets	0.036
Total Length of Fwd Packets	0.032
Flow IAT Min	0.032
Bwd Packet Length Std	0.028
Fwd Packet Length Max	0.020
Flow Duration	0.016
Fwd IAT Total	0.016
Flow IAT Std	0.004
Fwd Packet Length Std	0.000
Fwd Packet Length Mean	0.000
Flow Bytes/s	0.000
Flow Packets/s	0.000
Flow IAT Mean	0.000
Flow IAT Max	0.000
Fwd Packet Length Min	0.000
Bwd Packet Length Min	0.000

Tabla B.8: Importancia de los parámetros para el ataque Heartbleed (CICIDS2017)

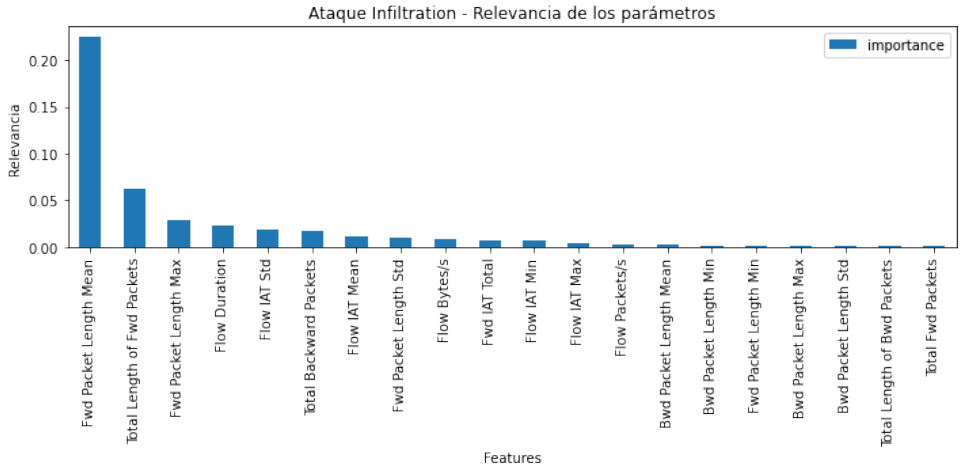

Figura B.9: Importancia de los parámetros para el ataque Infiltration (CICIDS2017)

Parámetro	**Importancia**
Fwd Packet Length Mean	0.224772
Total Length of Fwd Packets	0.062421
Fwd Packet Length Max	0.028231
Flow Duration	0.022988
Flow IAT Std	0.018528
Total Backward Packets	0.017390
Flow IAT Mean	0.011898
Fwd Packet Length Std	0.009862
Flow Bytes/s	0.008832
Fwd IAT Total	0.007862
Flow IAT Min	0.006825
Flow IAT Max	0.003532
Flow Packets/s	0.003299
Bwd Packet Length Mean	0.002072
Bwd Packet Length Min	0.001923
Fwd Packet Length Min	0.001691
Bwd Packet Length Max	0.001300
Bwd Packet Length Std	0.001246
Total Length of Bwd Packets	0.001151
Total Fwd Packets	0.000971

Tabla B.9: Importancia de los parámetros para el ataque Infiltration (CICIDS2017)

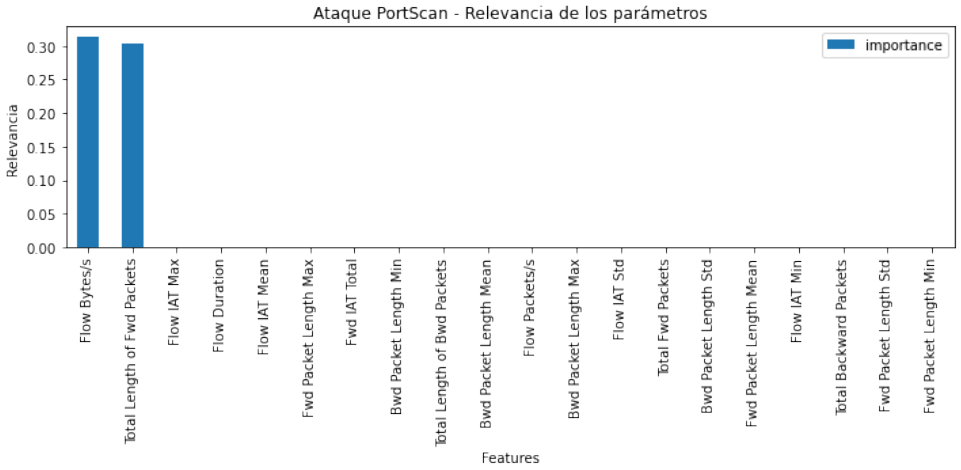

Figura B.10: Importancia de los parámetros para el ataque PortScan (CICIDS2017)

Parámetro	**Importancia**
Flow Bytes/s	0.313358
Total Length of Fwd Packets	0.304442
Flow IAT Max	0.000255
Flow Duration	0.000252
Flow IAT Mean	0.000231
Fwd Packet Length Max	0.000177
Fwd IAT Total	0.000130
Bwd Packet Length Min	0.000071
Total Length of Bwd Packets	0.000068
Bwd Packet Length Mean	0.000055
Flow Packets/s	0.000046
Bwd Packet Length Max	0.000038
Flow IAT Std	0.000027
Total Fwd Packets	0.000019
Bwd Packet Length Std	0.000017
Fwd Packet Length Mean	0.000014
Flow IAT Min	0.000014
Total Backward Packets	0.000014
Fwd Packet Length Std	0.000008
Fwd Packet Length Min	0.000001

Tabla B.10: Importancia de los parámetros para el ataque PortScan (CICIDS2017)

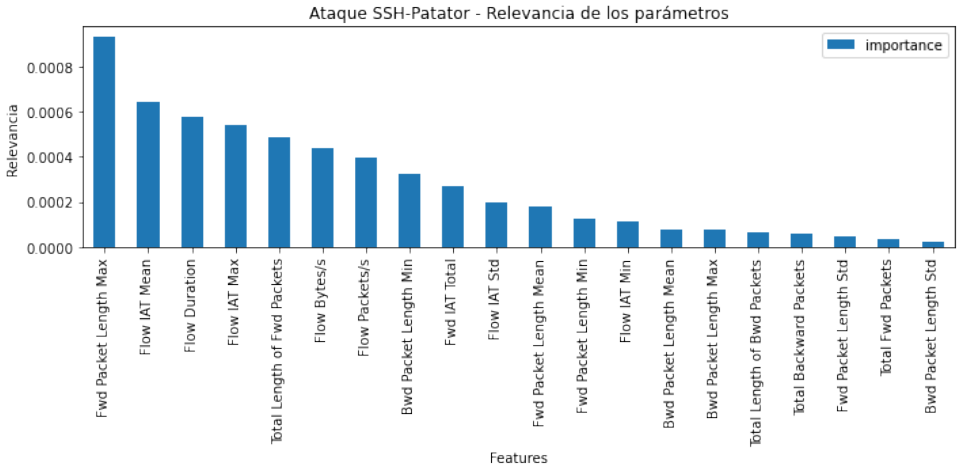

Figura B.11: Importancia de los parámetros para el ataque SSH-Patator (CICIDS2017)

Parámetro	**Importancia**
Fwd Packet Length Max	0.000931
Flow IAT Mean	0.000642
Flow Duration	0.000580
Flow IAT Max	0.000544
Total Length of Fwd Packets	0.000484
Flow Bytes/s	0.000441
Flow Packets/s	0.000399
Bwd Packet Length Min	0.000324
Fwd IAT Total	0.000268
Flow IAT Std	0.000198
Fwd Packet Length Mean	0.000180
Fwd Packet Length Min	0.000127
Flow IAT Min	0.000116
Bwd Packet Length Mean	0.000078
Bwd Packet Length Max	0.000075
Total Length of Bwd Packets	0.000067
Total Backward Packets	0.000057
Fwd Packet Length Std	0.000048
Total Fwd Packets	0.000036
Bwd Packet Length Std	0.000026

Tabla B.11: Importancia de los parámetros para el ataque SSH-Patator (CICIDS2017)

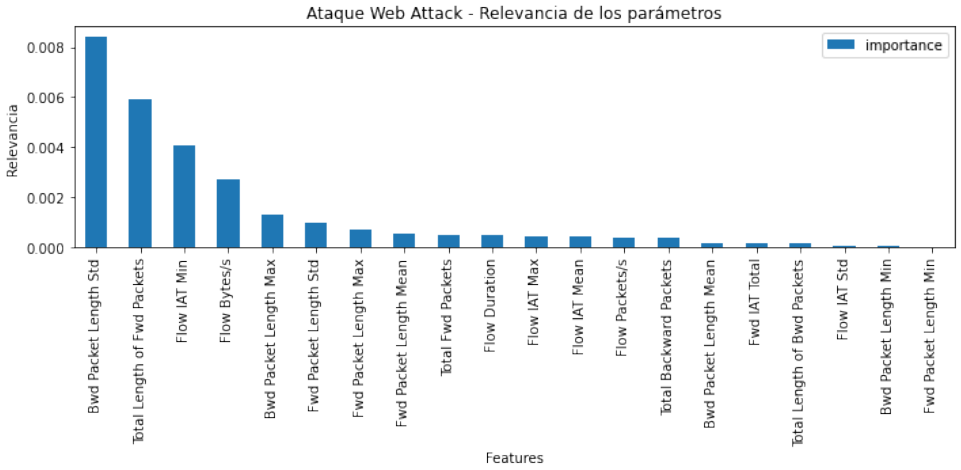

Figura B.12: Importancia de los parámetros para el ataque Web Attack (CICIDS2017)

Parámetro	**Importancia**
Bwd Packet Length Std	0.008407
Total Length of Fwd Packets	0.005940
Flow IAT Min	0.004051
Flow Bytes/s	0.002704
Bwd Packet Length Max	0.001307
Fwd Packet Length Std	0.000968
Fwd Packet Length Max	0.000693
Fwd Packet Length Mean	0.000525
Total Fwd Packets	0.000481
Flow Duration	0.000480
Flow IAT Max	0.000450
Flow IAT Mean	0.000422
Flow Packets/s	0.000369
Total Backward Packets	0.000368
Bwd Packet Length Mean	0.000146
Fwd IAT Total	0.000145
Total Length of Bwd Packets	0.000135
Flow IAT Std	0.000057
Bwd Packet Length Min	0.000033
Fwd Packet Length Min	0.000000

Tabla B.12: Importancia de los parámetros para el ataque Web Attack (CICIDS2017)

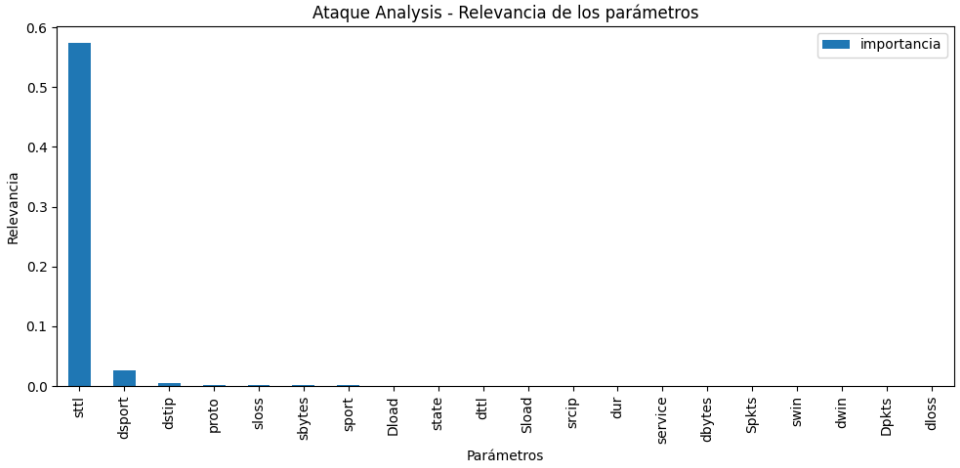

Figura B.13: Importancia de los parámetros para el ataque Analysis (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.573390
dsport	0.026739
dstip	0.004418
proto	0.001617
sloss	0.001478
sbytes	0.001425
sport	0.001121
Dload	0.000656
state	0.000575
dttl	0.000561
Sload	0.000533
srcip	0.000383
dur	0.000355
service	0.000189
dbytes	0.000163
Spkts	0.000160
swin	0.000059
dwin	0.000037
Dpkts	0.000017
dloss	0.000012

Tabla B.13: Importancia de los parámetros para el ataque Analysis (UNSW-NB15)

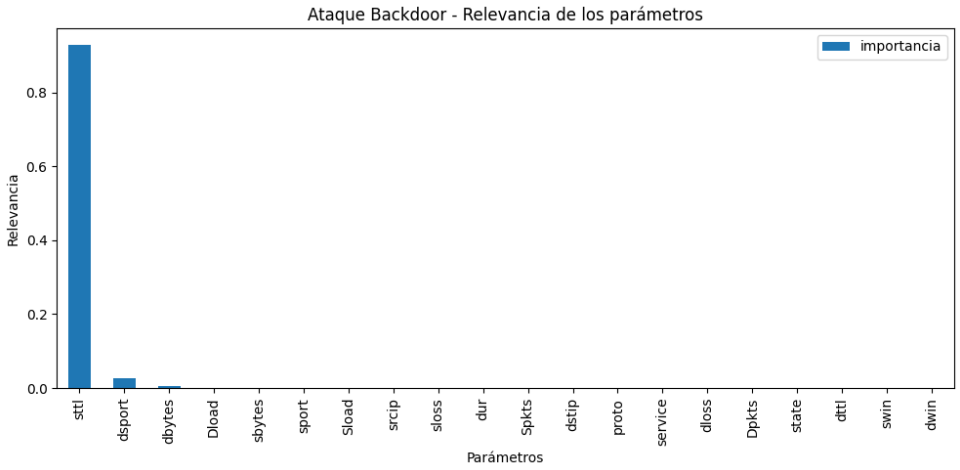

Figura B.14: Importancia de los parámetros para el ataque Backdoor (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.927527
dsport	0.025598
dbytes	0.004252
Dload	0.000637
sbytes	0.000541
sport	0.000445
Sload	0.000363
srcip	0.000337
sloss	0.000317
dur	0.000251
Spkts	0.000222
dstip	0.000169
proto	0.000143
service	0.000130
dloss	0.000114
Dpkts	0.000093
state	0.000056
dttl	0.000012
swin	0.000002
dwin	0.000000

Tabla B.14: Importancia de los parámetros para el ataque Backdoor (UNSW-NB15)

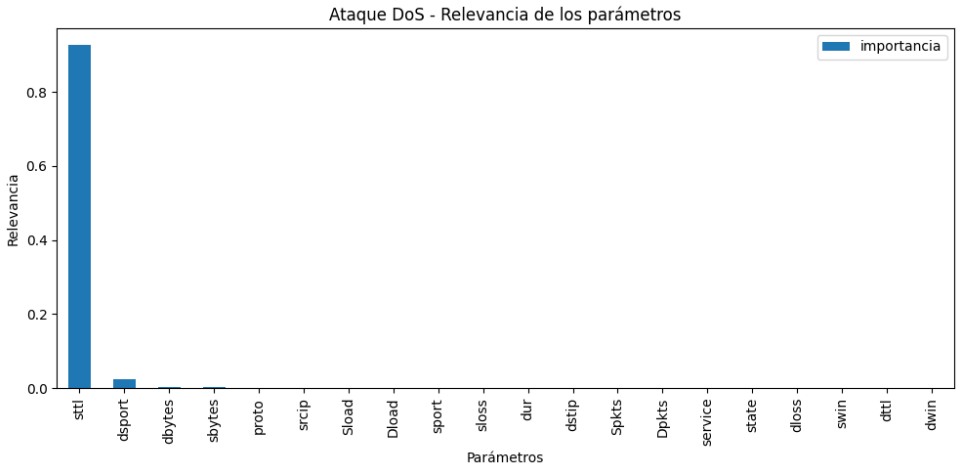

Figura B.15: Importancia de los parámetros para el ataque DoS (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.926416
dsport	0.025332
dbytes	0.003720
sbytes	0.001647
proto	0.001311
srcip	0.001094
Sload	0.000596
Dload	0.000555
sport	0.000555
sloss	0.000447
dur	0.000334
dstip	0.000288
Spkts	0.000253
Dpkts	0.000252
service	0.000244
state	0.000117
dloss	0.000063
swin	0.000034
dttl	0.000012
dwin	0.000003

Tabla B.15: Importancia de los parámetros para el ataque DoS (UNSW-NB15)

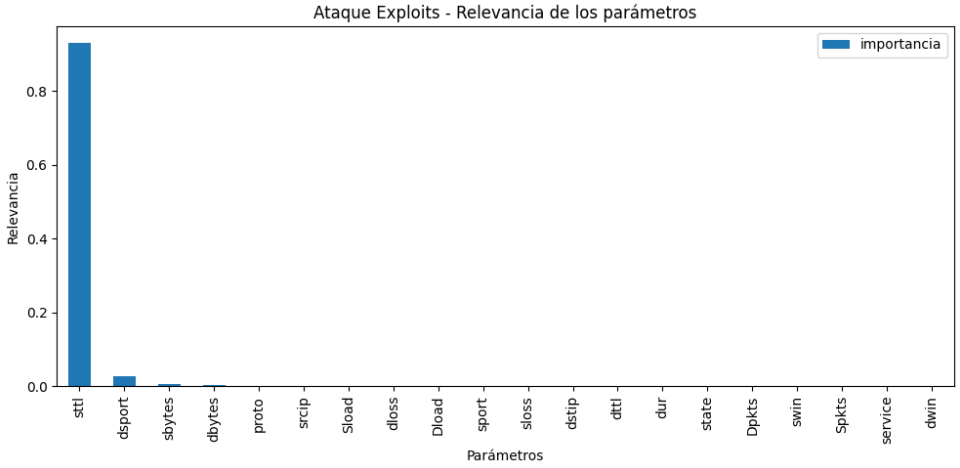

Figura B.16: Importancia de los parámetros para el ataque Exploits (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.929542
dsport	0.027404
sbytes	0.004697
dbytes	0.001776
proto	0.001072
srcip	0.000999
Sload	0.000856
dloss	0.000803
Dload	0.000756
sport	0.000629
sloss	0.000596
dstip	0.000465
dttl	0.000387
dur	0.000338
state	0.000312
Dpkts	0.000221
swin	0.000182
Spkts	0.000108
service	0.000061
dwin	0.000009

Tabla B.16: Importancia de los parámetros para el ataque Exploits (UNSW-NB15)

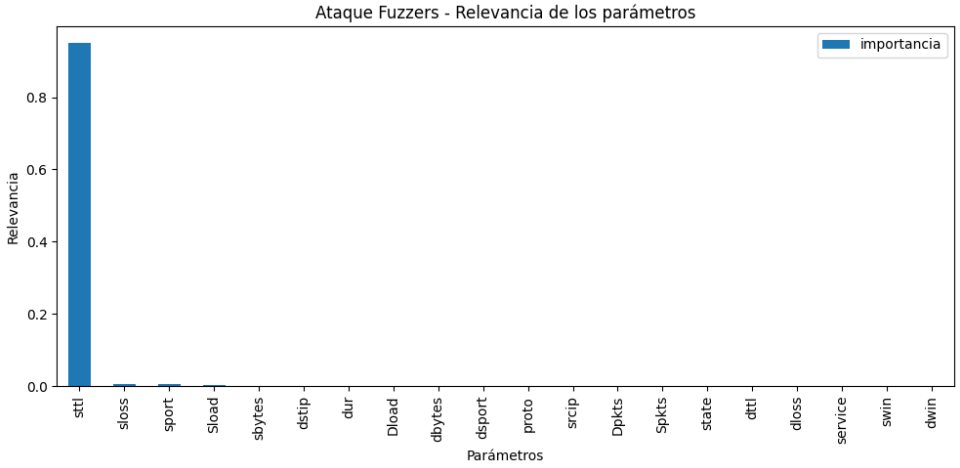

Figura B.17: Importancia de los parámetros para el ataque Fuzzers (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.949261
sloss	0.004627
sport	0.004603
Sload	0.002090
sbytes	0.001503
dstip	0.001408
dur	0.001376
Dload	0.001167
dbytes	0.000922
dsport	0.000714
proto	0.000639
srcip	0.000631
Dpkts	0.000212
Spkts	0.000199
state	0.000192
dttl	0.000153
dloss	0.000104
service	0.000087
swin	0.000032
dwin	0.000004

Tabla B.17: Importancia de los parámetros para el ataque Fuzzers (UNSW-NB15)

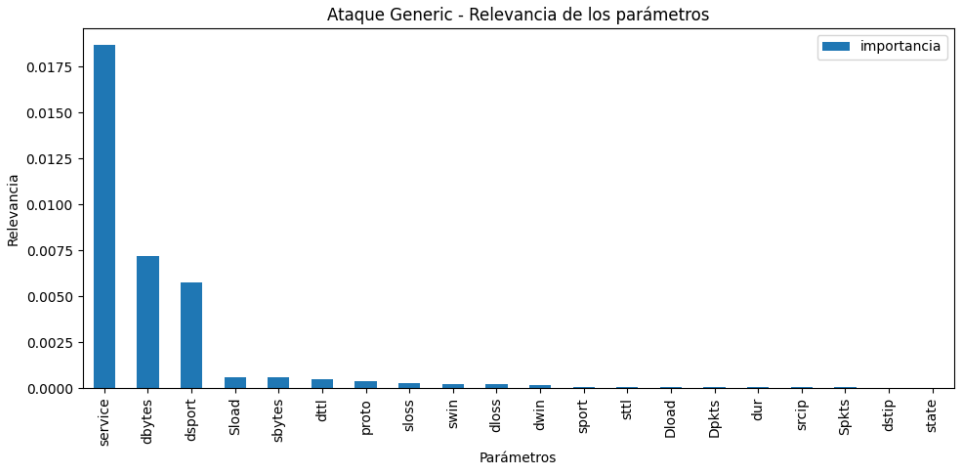

Figura B.18: Importancia de los parámetros para el ataque Generic (UNSW-NB15)

Parámetro	**Importancia**
service	0.018651
dbytes	0.007203
dsport	0.005746
Sload	0.000601
sbytes	0.000569
dttl	0.000495
proto	0.000374
sloss	0.000264
swin	0.000220
dloss	0.000210
dwin	0.000144
sport	0.000078
sttl	0.000071
Dload	0.000069
Dpkts	0.000059
dur	0.000044
srcip	0.000042
Spkts	0.000040
dstip	0.000027
state	0.000010

Tabla B.18: Importancia de los parámetros para el ataque Generic (UNSW-NB15)

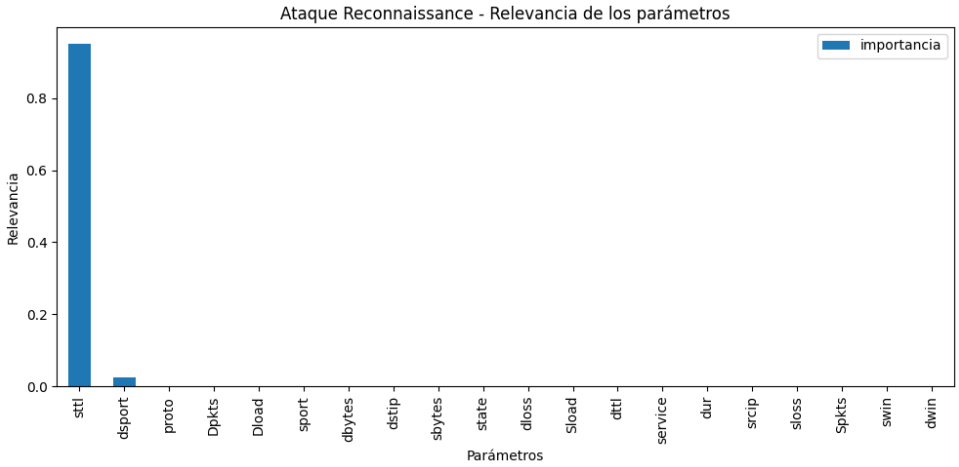

Figura B.19: Importancia de los parámetros para el ataque Reconnaissance (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.949993
dsport	0.023367
proto	0.001591
Dpkts	0.001574
Dload	0.001566
sport	0.001166
dbytes	0.000880
dstip	0.000786
sbytes	0.000492
state	0.000134
dloss	0.000079
Sload	0.000076
dttl	0.000061
service	0.000051
dur	0.000041
srcip	0.000030
sloss	0.000018
Spkts	0.000016
swin	0.000003
dwin	0.000000

Tabla B.19: Importancia de los parámetros para el ataque Reconnaissance (UNSW-NB15)

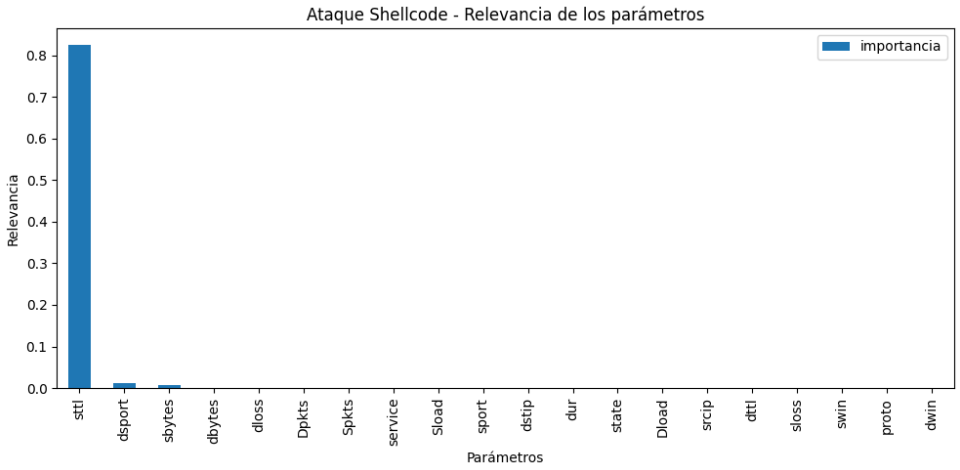

Figura B.20: Importancia de los parámetros para el ataque Shellcode (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.823958
dsport	0.012884
sbytes	0.008264
dbytes	0.001093
dloss	0.000918
Dpkts	0.000666
Spkts	0.000612
service	0.000437
Sload	0.000338
sport	0.000209
dstip	0.000179
dur	0.000112
state	0.000073
Dload	0.000052
srcip	0.000011
dttl	0.000008
sloss	0.000008
swin	0.000003
proto	0.000000
dwin	0.000000

Tabla B.20: Importancia de los parámetros para el ataque Shellcode (UNSW-NB15)

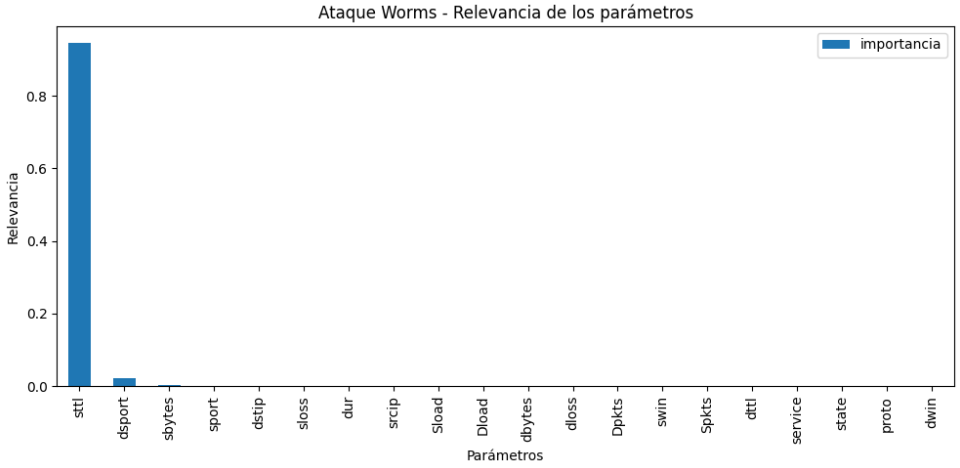

Figura B.21: Importancia de los parámetros para el ataque Worms (UNSW-NB15)

Parámetro	**Importancia**
sttl	0.944970
dsport	0.022880
sbytes	0.003631
sport	0.000298
dstip	0.000252
sloss	0.000204
dur	0.000187
srcip	0.000167
Sload	0.000163
Dload	0.000110
dbytes	0.000075
dloss	0.000035
Dpkts	0.000022
swin	0.000000
Spkts	0.000000
dttl	0.000000
service	0.000000
state	0.000000
proto	0.000000
dwin	0.000000

Tabla B.21: Importancia de los parámetros para el ataque Worms (UNSW-NB15)

Apéndice C

Resultados de la implementación - CICIDS2017

En este apéndice se incluyen todos los resultados obtenidos en las ejecuciones con el dataset CICIDS2017. Para cada caso se detallan los valores de accuracy, precision, recall, f-score y tiempo de ejecución (expresado en segundos). Las tablas siguientes corresponden a las pruebas realizadas por cada tipo de ataque y para el conjunto total de datos, distinguiendo a su vez para 18 y 7 parámetros.

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.45	0.67	0.62	0.45	0.0122
QDA	0.69	0.74	0.78	0.68	0.0139
Random Forest	0.95	0.94	0.95	0.94	0.0447
ID3	0.96	0.94	0.97	0.95	0.0156
AdaBoost	0.97	0.96	0.98	0.97	0.271
MLP	0.74	0.68	0.65	0.66	0.6638
K Nearest Neighbours	0.97	0.96	0.97	0.96	0.1836
Logistic Regression	0.66	0.73	0.76	0.66	0.0452
Support Vector Machine	0.71	0.35	0.5	0.41	0.7677
Multinomial Naïve Bayes	0.52	0.58	0.59	0.52	0.0137
SGD Classifier	0.62	0.55	0.55	0.55	0.3147
Gradient Boosting Classifier	0.98	0.96	0.98	0.97	0.3019
K-Means	0.35	0.34	0.45	0.39	0.0342
K-Medoids	0.36	0.34	0.44	0.38	3.0096
Isolation Forest	0.43	0.44	0.44	0.44	0.0768
Local Outlier Factor	0.4	0.24	0.44	0.43	0.0996

Tabla C.1: Resultados para el ataque Bot

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.35	0.66	0.53	0.3	0.0675
QDA	0.82	0.81	0.87	0.81	0.1126
Random Forest	0.97	0.96	0.98	0.97	0.4987
ID3	0.97	0.96	0.98	0.97	0.4643
AdaBoost	0.97	0.96	0.98	0.97	2.959
MLP	0.82	0.82	0.78	0.77	36.238
K Nearest Neighbours	0.93	0.91	0.93	0.92	4.0651
Logistic Regression	0.85	0.84	0.79	0.81	0.3707
Support Vector Machine	0.87	0.89	0.8	0.83	201.885
Multinomial Naïve Bayes	0.75	0.7	0.67	0.68	0.0721
SGD Classifier	0.67	0.61	0.62	0.62	0.5657
Gradient Boosting Classifier	0.97	0.96	0.98	0.97	3.6809
K-Means	0.6	0.64	0.59	0.59	0.1766
K-Medoids	0.64	0.64	0.58	0.58	5.0856
Isolation Forest	0.61	0.61	0.61	0.61	0.3465
Local Outlier Factor	0.47	0.47	0.48	0.47	4.5939

Tabla C.2: Resultados para el ataque DDoS

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.82	0.82	0.74	0.77	0.0409
QDA	0.67	0.73	0.76	0.67	0.062
Random Forest	0.99	0.98	0.99	0.99	0.1263
ID3	0.99	0.99	0.99	0.99	0.0471
AdaBoost	0.98	0.98	0.98	0.98	0.7146
MLP	0.71	0.74	0.76	0.7	4.6151
K Nearest Neighbours	0.98	0.97	0.98	0.97	0.7494
Logistic Regression	0.93	0.91	0.92	0.92	0.116
Support Vector Machine	0.94	0.94	0.93	0.93	8.2287
Multinomial Naïve Bayes	0.73	0.74	0.79	0.72	0.0272
SGD Classifier	0.8	0.76	0.78	0.77	0.2927
Gradient Boosting Classifier	0.99	0.99	0.99	0.99	1.1168
K-Means	0.61	0.68	0.57	0.56	0.0594
K-Medoids	0.85	0.84	0.89	0.84	4.9092
Isolation Forest	0.79	0.79	0.79	0.79	0.1267
Local Outlier Factor	0.46	0.47	0.48	0.47	0.6062

Tabla C.3: Resultados para el ataque DoS Gondeneye

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.34	0.65	0.54	0.3	0.4615
QDA	0.41	0.66	0.59	0.4	0.6533
Random Forest	0.94	0.94	0.9	0.92	2.5766
ID3	0.96	0.94	0.95	0.95	1.012
AdaBoost	0.96	0.95	0.95	0.95	15.7894
MLP	0.95	0.94	0.94	0.94	74.1242
K Nearest Neighbours	0.96	0.95	0.96	0.95	35.741
Logistic Regression	0.87	0.86	0.81	0.83	4.1771
Support Vector Machine	0.89	0.89	0.82	0.85	844.4542
Multinomial Naïve Bayes	0.85	0.82	0.8	0.81	0.6841
SGD Classifier	0.71	0.66	0.67	0.66	242.3775
Gradient Boosting Classifier	0.96	0.95	0.95	0.95	28.5785
K-Means	0.42	0.36	0.5	0.42	1.1706
K-Medoids	0.8	0.86	0.81	0.83	3.6214
Isolation Forest	0.57	0.57	0.57	0.57	2.1782
Local Outlier Factor	0.5	0.5	0.5	0.5	149.7249

Tabla C.4: Resultados para el ataque DoS Hulk

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.94	0.94	0.9	0.92	0.0283
QDA	0.94	0.94	0.92	0.93	0.0218
Random Forest	0.97	0.98	0.96	0.97	0.0744
ID3	0.97	0.98	0.94	0.96	0.0281
AdaBoost	0.97	0.98	0.96	0.97	0.4445
MLP	0.74	0.78	0.73	0.66	1.9783
K Nearest Neighbours	0.97	0.97	0.97	0.97	0.5074
Logistic Regression	0.95	0.94	0.93	0.93	0.1469
Support Vector Machine	0.95	0.94	0.93	0.94	9.0385
Multinomial Naïve Bayes	0.8	0.76	0.81	0.77	0.0229
SGD Classifier	0.4	0.49	0.49	0.4	2.5843
Gradient Boosting Classifier	0.98	0.98	0.97	0.98	0.6567
K-Means	0.9	0.92	0.87	0.89	0.0425
K-Medoids	0.93	0.92	0.88	0.9	4.4357
Isolation Forest	0.59	0.59	0.59	0.59	0.0934
Local Outlier Factor	0.41	0.41	0.43	0.42	0.7886

Tabla C.5: Resultados para el ataque DoS Slowhttptest

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.42	0.67	0.58	0.4	0.015
QDA	0.48	0.67	0.62	0.48	0.0155
Random Forest	0.94	0.94	0.93	0.93	0.0676
ID3	0.96	0.97	0.94	0.95	0.0282
AdaBoost	0.96	0.96	0.94	0.95	0.4316
MLP	0.81	0.77	0.78	0.74	2.3008
K Nearest Neighbours	0.94	0.93	0.93	0.93	0.3858
Logistic Regression	0.9	0.88	0.89	0.89	0.1469
Support Vector Machine	0.85	0.87	0.77	0.79	9.127
Multinomial Naïve Bayes	0.88	0.86	0.91	0.87	0.023
SGD Classifier	0.77	0.76	0.67	0.69	2.0468
Gradient Boosting Classifier	0.96	0.97	0.94	0.95	0.6687
K-Means	0.75	0.8	0.6	0.59	0.0398
K-Medoids	0.77	0.83	0.61	0.61	5.1326
Isolation Forest	0.52	0.52	0.52	0.52	0.111
Local Outlier Factor	0.43	0.41	0.44	0.42	0.594

Tabla C.6: Resultados para el ataque DoS Slowloris

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	1.0	1.0	1.0	1.0	0.0189
QDA	1.0	1.0	0.99	1.0	0.021
Random Forest	1.0	1.0	1.0	1.0	0.0793
ID3	1.0	1.0	1.0	1.0	0.0281
AdaBoost	1.0	1.0	1.0	1.0	0.6382
MLP	1.0	1.0	1.0	1.0	5.5263
K Nearest Neighbours	1.0	1.0	1.0	1.0	0.7549
Logistic Regression	0.75	0.78	0.82	0.75	0.18
Support Vector Machine	0.77	0.79	0.83	0.77	11.5659
Multinomial Naïve Bayes	0.76	0.78	0.82	0.75	0.0268
SGD Classifier	0.64	0.59	0.6	0.59	0.6506
Gradient Boosting Classifier	1.0	1.0	1.0	1.0	0.8596
K-Means	0.41	0.34	0.49	0.41	0.0704
K-Medoids	0.23	0.24	0.2	0.22	4.7277
Isolation Forest	0.28	0.28	0.29	0.28	0.1047
Local Outlier Factor	0.49	0.48	0.49	0.48	1.253

Tabla C.7: Resultados para el ataque FTP-Patator

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.86	0.83	0.9	0.84	0.0085
QDA	1.0	1.0	1.0	1.0	0.0085
Random Forest	1.0	1.0	1.0	1.0	0.0251
ID3	1.0	1.0	1.0	1.0	0.0082
AdaBoost	1.0	1.0	1.0	1.0	0.0108
MLP	0.57	0.56	0.64	0.54	0.0448
K Nearest Neighbours	1.0	1.0	1.0	1.0	0.0109
Logistic Regression	1.0	1.0	1.0	1.0	0.0151
Support Vector Machine	1.0	1.0	1.0	1.0	0.0109
Multinomial Naïve Bayes	0.86	0.83	0.9	0.84	0.0114
SGD Classifier	1.0	1.0	1.0	1.0	0.0112
Gradient Boosting Classifier	1.0	1.0	1.0	1.0	0.0592
K-Means	1.0	1.0	1.0	1.0	0.0204
K-Medoids	1.0	1.0	1.0	1.0	0.0087
Isolation Forest	0.53	0.54	0.55	0.53	0.0499
Local Outlier Factor	0.3	0.3	0.3	0.3	0.0521

Tabla C.8: Resultados para el ataque Heartbleed

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.92	0.89	0.89	0.89	0.0085
QDA	0.79	0.74	0.81	0.76	0.0079
Random Forest	0.9	0.87	0.87	0.87	0.0205
ID3	0.92	0.89	0.89	0.89	0.0075
AdaBoost	0.92	0.89	0.89	0.89	0.0848
MLP	0.35	0.51	0.54	0.3	0.0313
K Nearest Neighbours	0.88	0.83	0.86	0.84	0.0117
Logistic Regression	0.79	0.74	0.81	0.76	0.0141
Support Vector Machine	0.92	0.89	0.89	0.89	0.0126
Multinomial Naïve Bayes	0.75	0.64	0.56	0.55	0.0134
SGD Classifier	0.78	0.8	0.8	0.75	0.0119
Gradient Boosting Classifier	0.92	0.89	0.89	0.89	0.0663
K-Means	0.72	0.73	0.75	0.74	0.0234
K-Medoids	0.85	0.87	0.88	0.87	0.0088
Isolation Forest	0.62	0.63	0.64	0.63	0.0445
Local Outlier Factor	0.95	0.97	0.92	0.94	0.0425

Tabla C.9: Resultados para el ataque Infiltration

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.44	0.66	0.6	0.43	0.2653
QDA	0.8	0.79	0.86	0.79	0.5029
Random Forest	1.0	1.0	0.99	0.99	2.272
ID3	1.0	1.0	1.0	1.0	1.0001
AdaBoost	1.0	1.0	0.99	0.99	13.914
MLP	0.73	0.76	0.57	0.49	48.998
K Nearest Neighbours	1.0	1.0	1.0	1.0	17.3794
Logistic Regression	0.47	0.68	0.63	0.46	0.7815
Support Vector Machine	0.85	0.83	0.89	0.84	314.7009
Multinomial Naïve Bayes	0.46	0.67	0.62	0.45	0.0677
SGD Classifier	0.64	0.61	0.63	0.61	15.8212
Gradient Boosting Classifier	1.0	1.0	0.99	0.99	4.01
K-Means	0.42	0.35	0.45	0.39	0.7936
K-Medoids	0.41	0.34	0.45	0.39	4.9826
Isolation Forest	0.3	0.3	0.3	0.3	1.8234
Local Outlier Factor	0.34	0.33	0.36	0.34	37.7851

Tabla C.10: Resultados para el ataque PortScan

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.43	0.67	0.6	0.42	0.0172
QDA	0.57	0.7	0.69	0.57	0.0178
Random Forest	0.96	0.94	0.96	0.95	0.1006
ID3	0.96	0.95	0.97	0.96	0.0395
AdaBoost	0.96	0.95	0.96	0.96	0.5823
MLP	0.88	0.87	0.85	0.85	2.418
K Nearest Neighbours	0.95	0.94	0.95	0.95	0.463
Logistic Regression	0.46	0.68	0.62	0.45	0.2424
Support Vector Machine	0.7	0.35	0.5	0.41	23.0391
Multinomial Naïve Bayes	0.4	0.44	0.43	0.39	0.0233
SGD Classifier	0.42	0.45	0.44	0.41	0.2752
Gradient Boosting Classifier	0.96	0.95	0.96	0.96	0.9435
K-Means	0.4	0.34	0.45	0.39	0.0475
K-Medoids	0.6	0.66	0.65	0.65	6.5801
Isolation Forest	0.5	0.5	0.5	0.5	0.338
Local Outlier Factor	0.4	0.38	0.41	0.39	0.2466

Tabla C.11: Resultados para el ataque SSH-Patator

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.78	0.79	0.83	0.77	0.0182
QDA	0.84	0.83	0.88	0.83	0.0187
Random Forest	0.95	0.95	0.94	0.95	0.0699
ID3	0.96	0.95	0.95	0.95	0.0253
AdaBoost	0.96	0.96	0.95	0.95	0.2441
MLP	0.91	0.89	0.9	0.88	1.6921
K Nearest Neighbours	0.95	0.94	0.95	0.94	0.1717
Logistic Regression	0.47	0.64	0.6	0.47	0.04
Support Vector Machine	0.7	0.79	0.52	0.46	1.319
Multinomial Naïve Bayes	0.56	0.38	0.42	0.39	0.0156
SGD Classifier	0.63	0.49	0.52	0.49	0.0517
Gradient Boosting Classifier	0.96	0.96	0.95	0.95	0.3052
K-Means	0.37	0.34	0.5	0.41	0.037
K-Medoids	0.37	0.33	0.41	0.36	2.7704
Isolation Forest	0.34	0.34	0.35	0.35	0.1982
Local Outlier Factor	0.34	0.33	0.38	0.35	0.0831

Tabla C.12: Resultados para el ataque Web Attack

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.78	0.62	0.64	0.63	4.3063
QDA	0.61	0.58	0.58	0.62	8.6075
Random Forest	0.94	0.97	0.84	0.89	22.916
ID3	0.96	0.94	0.9	0.96	25.1759
AdaBoost	0.95	0.94	0.88	0.9	296.3007
MLP	0.84	0.75	0.73	0.83	180.3331
K Nearest Neighbours	0.97	0.94	0.94	0.94	1483.5885
Logistic Regression	0.7	0.67	0.8	0.65	84.2076
Support Vector Machine	0.88	0.89	0.66	0.7	352.7382
Multinomial Naïve Bayes	0.82	0.68	0.66	0.67	3.4888
SGD Classifier	0.75	0.66	0.76	0.67	58.2526
Gradient Boosting Classifier	0.95	0.96	0.87	0.91	18.5087
K-Means	0.81	0.65	0.63	0.64	11.1541
K-Medoids	0.64	0.57	0.62	0.55	3.3586
Isolation Forest	0.68	0.56	0.58	0.55	9.4715
Local Outlier Factor	0.7	0.49	0.49	0.49	32.5694

Tabla C.13: Resultados para todo el dataset con 18 parámetros

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.82	0.66	0.63	0.64	1.2446
QDA	0.86	0.76	0.68	0.7	3.7565
Random Forest	0.94	0.96	0.82	0.87	10.5677
ID3	0.95	0.93	0.89	0.96	6.7232
AdaBoost	0.95	0.93	0.86	0.89	92.7308
MLP	0.94	0.95	0.82	0.87	1859.9979
K Nearest Neighbours	0.97	0.93	0.95	0.94	211.0688
Logistic Regression	0.85	0.73	0.68	0.7	17.1998
Support Vector Machine	0.87	0.79	0.66	0.7	298.9161
Multinomial Naïve Bayes	0.79	0.63	0.64	0.64	1.0744
SGD Classifier	0.44	0.52	0.53	0.41	1047.5976
Gradient Boosting Classifier	0.95	0.95	0.87	0.9	198.9961
K-Means	0.81	0.65	0.63	0.64	6.8428
K-Medoids	0.81	0.65	0.63	0.64	4.8439
Isolation Forest	0.68	0.55	0.58	0.55	10.2262
Local Outlier Factor	0.7	0.5	0.5	0.5	4.8384

Tabla C.14: Resultados para todo el dataset con 7 parámetros

Apéndice D

Resultados de la implementación - UNSW-NB15

En este apéndice se incluyen todos los resultados obtenidos en las ejecuciones con el dataset UNSW-NB15. Para cada caso se detallan los valores de accuracy, precision, recall, f-score y tiempo de ejecución (expresado en segundos). Las tablas siguientes corresponden a las pruebas realizadas por cada tipo de ataque y para el conjunto total de datos, distinguiendo a su vez para 12 y 7 parámetros.

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.85	0.83	0.83	0.83	0.0153
QDA	0.99	0.98	0.98	0.98	0.0171
Random Forest	1.0	1.0	1.0	1.0	0.0316
ID3	1.0	1.0	0.99	0.99	0.0116
AdaBoost	1.0	1.0	1.0	1.0	0.1863
MLP	0.94	0.95	0.93	0.93	0.7534
K Nearest Neighbours	0.99	0.99	0.99	0.99	0.27
Logistic Regression	0.92	0.93	0.88	0.9	0.0437
Support Vector Machine	0.92	0.94	0.88	0.9	1.2762
Multinomial Naïve Bayes	0.76	0.78	0.83	0.76	0.0134
SGD Classifier	0.7	0.69	0.72	0.68	0.1572
Gradient Boosting Classifier	0.99	0.98	0.98	0.98	0.1718
K-Means	0.37	0.34	0.47	0.4	0.093
K-Medoids	0.28	0.29	0.28	0.28	10.0029
Isolation Forest	0.36	0.36	0.36	0.36	0.191
Local Outlier Factor	0.5	0.5	0.5	0.49	0.115

Tabla D.1: Resultados para el ataque Analysis

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.69	0.56	0.53	0.5	0.0133
QDA	0.97	0.96	0.97	0.96	0.0127
Random Forest	0.99	0.98	0.99	0.98	0.0337
ID3	0.99	0.98	0.99	0.99	0.0134
AdaBoost	0.98	0.98	0.99	0.98	0.2047
MLP	0.71	0.74	0.63	0.57	0.543
K Nearest Neighbours	0.96	0.94	0.95	0.95	0.133
Logistic Regression	0.98	0.97	0.97	0.97	0.0376
Support Vector Machine	0.98	0.97	0.97	0.97	0.4457
Multinomial Naïve Bayes	0.9	0.87	0.91	0.88	0.0155
SGD Classifier	0.95	0.93	0.94	0.93	0.5542
Gradient Boosting Classifier	0.99	0.99	0.99	0.99	0.2079
K-Means	0.59	0.68	0.57	0.56	0.0415
K-Medoids	0.82	0.84	0.82	0.83	4.4366
Isolation Forest	0.53	0.53	0.53	0.53	0.1631
Local Outlier Factor	0.52	0.53	0.52	0.52	0.1102

Tabla D.2: Resultados para el ataque Backdoor

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.94	0.93	0.93	0.93	0.0348
QDA	0.94	0.93	0.93	0.93	0.032
Random Forest	0.99	0.98	0.99	0.99	0.1134
ID3	0.99	0.98	0.99	0.99	0.0373
AdaBoost	0.99	0.98	0.99	0.99	0.8045
MLP	0.94	0.93	0.93	0.93	5.9237
K Nearest Neighbours	0.99	0.99	0.99	0.99	1.6144
Logistic Regression	0.95	0.96	0.93	0.94	0.1198
Support Vector Machine	0.96	0.96	0.94	0.95	43.4807
Multinomial Naïve Bayes	0.89	0.86	0.9	0.88	0.0363
SGD Classifier	0.93	0.92	0.91	0.91	0.2507
Gradient Boosting Classifier	0.99	0.98	0.99	0.98	0.8304
K-Means	0.42	0.4	0.49	0.41	0.1003
K-Medoids	0.4	0.36	0.42	0.38	16.5475
Isolation Forest	0.52	0.52	0.52	0.52	0.3786
Local Outlier Factor	0.49	0.48	0.49	0.48	1.7987

Tabla D.3: Resultados para el ataque DoS

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.85	0.85	0.79	0.81	0.0804
QDA	0.88	0.9	0.81	0.84	0.0876
Random Forest	0.99	0.98	0.99	0.98	0.3478
ID3	0.99	0.99	0.98	0.99	0.108
AdaBoost	0.99	0.98	0.99	0.99	2.7252
MLP	0.96	0.96	0.95	0.96	13.7761
K Nearest Neighbours	0.99	0.99	0.99	0.99	5.2117
Logistic Regression	0.96	0.96	0.95	0.95	0.1152
Support Vector Machine	0.87	0.89	0.79	0.82	58.2061
Multinomial Naïve Bayes	0.62	0.63	0.65	0.6	0.0914
SGD Classifier	0.79	0.75	0.73	0.74	0.6082
Gradient Boosting Classifier	0.98	0.98	0.99	0.98	2.5647
K-Means	0.56	0.85	0.5	0.42	0.327
K-Medoids	0.35	0.38	0.37	0.38	3.6924
Isolation Forest	0.69	0.69	0.69	0.69	0.839
Local Outlier Factor	0.56	0.57	0.55	0.55	7.1877

Tabla D.4: Resultados para el ataque Exploits

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.7	0.35	0.5	0.41	0.0559
QDA	0.99	0.98	0.99	0.99	0.0648
Random Forest	0.99	0.98	0.99	0.99	0.231
ID3	0.99	0.98	0.99	0.99	0.0841
AdaBoost	0.99	0.98	0.99	0.99	1.6642
MLP	0.65	0.77	0.67	0.61	4.0331
K Nearest Neighbours	0.77	0.72	0.7	0.71	1.9364
Logistic Regression	0.99	0.98	0.99	0.99	0.177
Support Vector Machine	0.99	0.98	0.99	0.99	14.5126
Multinomial Naïve Bayes	0.7	0.73	0.77	0.69	0.0487
SGD Classifier	0.94	0.93	0.91	0.92	7.7626
Gradient Boosting Classifier	0.99	0.98	0.99	0.99	1.9743
K-Means	0.5	0.5	0.5	0.5	0.1999
K-Medoids	0.49	0.49	0.49	0.48	7.9071
Isolation Forest	0.65	0.65	0.65	0.65	0.516
Local Outlier Factor	0.59	0.6	0.58	0.58	2.2916

Tabla D.5: Resultados para el ataque Fuzzers

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.7	0.61	0.58	0.58	0.4574
QDA	0.65	0.72	0.75	0.65	0.5571
Random Forest	0.99	1.0	0.99	0.99	3.5781
ID3	0.99	1.0	0.99	0.99	1.0917
AdaBoost	0.99	1.0	0.99	0.99	22.6866
MLP	0.72	0.52	0.5	0.42	34.7862
K Nearest Neighbours	1.0	1.0	1.0	0.99	1.0
Logistic Regression	0.89	0.85	0.92	0.87	0.0853
Support Vector Machine	0.94	0.91	0.95	0.92	22.322
Multinomial Naïve Bayes	0.92	0.89	0.94	0.91	0.0395
SGD Classifier	0.79	0.78	0.85	0.78	4.9902
Gradient Boosting Classifier	0.99	0.99	0.99	0.99	1.1161
K-Means	0.59	0.65	0.57	0.57	1.2662
K-Medoids	0.9	0.89	0.93	0.91	6.6764
Isolation Forest	0.36	0.36	0.35	0.35	1.8096
Local Outlier Factor	0.34	0.34	0.37	0.36	252.4371

Tabla D.6: Resultados para el ataque Generic

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.86	0.84	0.9	0.85	0.0273
QDA	0.99	0.98	0.99	0.99	0.0266
Random Forest	0.99	0.99	0.99	0.99	0.1104
ID3	1.0	0.99	1.0	0.99	0.0436
AdaBoost	0.99	0.99	0.99	0.99	0.793
MLP	0.98	0.98	0.98	0.98	2.0645
K Nearest Neighbours	0.99	0.99	0.99	0.99	0.8713
Logistic Regression	0.99	0.98	0.99	0.99	0.1011
Support Vector Machine	0.99	0.98	0.99	0.99	9.8577
Multinomial Naïve Bayes	0.88	0.86	0.92	0.87	0.0805
SGD Classifier	0.98	0.97	0.98	0.97	3.7164
Gradient Boosting Classifier	0.99	0.98	0.99	0.98	2.4968
K-Means	0.4	0.33	0.41	0.37	0.0971
K-Medoids	0.35	0.33	0.41	0.37	15.83
Isolation Forest	0.4	0.4	0.4	0.4	0.1271
Local Outlier Factor	0.41	0.4	0.42	0.41	2.1827

Tabla D.7: Resultados para el ataque Reconnaissance

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	1.0	0.99	1.0	1.0	0.0097
QDA	0.29	0.15	0.5	0.23	0.0089
Random Forest	1.0	1.0	1.0	1.0	0.0235
ID3	1.0	1.0	1.0	1.0	0.0091
AdaBoost	1.0	1.0	1.0	1.0	0.131
MLP	1.0	1.0	1.0	1.0	1.162
K Nearest Neighbours	1.0	1.0	1.0	1.0	0.0808
Logistic Regression	0.99	0.99	1.0	0.99	0.0506
Support Vector Machine	1.0	0.99	1.0	1.0	0.0446
Multinomial Naïve Bayes	0.78	0.78	0.84	0.77	0.0109
SGD Classifier	0.98	0.98	0.97	0.98	0.0145
Gradient Boosting Classifier	0.99	0.99	1.0	0.99	0.124
K-Means	0.37	0.35	0.47	0.4	0.0344
K-Medoids	0.27	0.27	0.28	0.27	1.0194
Isolation Forest	0.66	0.66	0.66	0.66	0.0517
Local Outlier Factor	0.44	0.42	0.44	0.42	0.1106

Tabla D.8: Resultados para el ataque Shellcode

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.71	0.35	0.5	0.42	0.0073
QDA	0.98	0.98	0.98	0.98	0.0075
Random Forest	0.99	0.99	0.99	0.99	0.0186
ID3	0.99	0.99	0.99	0.99	0.0073
AdaBoost	0.99	0.99	0.99	0.99	0.0782
MLP	0.59	0.63	0.54	0.45	0.1012
K Nearest Neighbours	0.65	0.56	0.55	0.55	0.0174
Logistic Regression	0.98	0.97	0.97	0.97	0.0462
Support Vector Machine	0.98	0.97	0.97	0.97	0.0265
Multinomial Naïve Bayes	0.69	0.71	0.75	0.68	0.0433
SGD Classifier	0.98	0.97	0.97	0.97	0.0636
Gradient Boosting Classifier	1.0	1.0	1.0	1.0	0.2659
K-Means	0.61	0.6	0.62	0.59	0.0315
K-Medoids	0.63	0.64	0.69	0.62	0.0191
Isolation Forest	0.63	0.62	0.63	0.63	0.0401
Local Outlier Factor	0.56	0.59	0.57	0.58	0.0367

Tabla D.9: Resultados para el ataque Worms

Algoritmo	Accuracy	Precision	Recall	F-Score	Tiempo
Gaussian Naïve Bayes	0.81	0.58	0.58	0.58	2.7677
QDA	0.97	0.92	0.96	0.94	4.1358
Random Forest	0.98	0.97	0.95	0.96	16.1722
ID3	0.99	0.98	0.97	0.97	8.1971
AdaBoost	0.99	0.98	0.98	0.98	225.1423
MLP	0.87	0.77	0.5	0.48	599.7932
K Nearest Neighbours	0.99	0.98	0.97	0.97	268.961
Logistic Regression	0.98	0.94	0.96	0.95	10.9124
Support Vector Machine	0.98	0.95	0.96	0.95	474.8378
Multinomial Naïve Bayes	0.97	0.92	0.96	0.94	3.5226
SGD Classifier	0.88	0.74	0.89	0.79	7.279
Gradient Boosting Classifier	0.97	0.92	0.98	0.95	12.9346
K-Means	0.85	0.44	0.48	0.46	8.4687
K-Medoids	0.61	0.6	0.72	0.54	5.2122
Isolation Forest	0.68	0.54	0.57	0.53	8.452
Local Outlier Factor	0.77	0.56	0.59	0.57	39.608

Tabla D.10: Resultados para todo el dataset con 12 parámetros

Algoritmo	**Accuracy**	**Precision**	**Recall**	**F-Score**	**Tiempo**
Gaussian Naïve Bayes	0.86	0.59	0.52	0.51	2.2354
QDA	0.87	0.66	0.55	0.56	2.4654
Random Forest	0.98	0.98	0.91	0.94	19.0587
ID3	0.98	0.98	0.95	0.96	9.4779
AdaBoost	0.99	0.97	0.97	0.97	165.1903
MLP	0.87	0.92	0.5	0.47	133.4903
K Nearest Neighbours	0.99	0.97	0.97	0.97	132.4598
Logistic Regression	0.75	0.61	0.71	0.62	3.6722
Support Vector Machine	0.87	0.94	0.5	0.47	12034.2238
Multinomial Naïve Bayes	0.7	0.57	0.64	0.56	2.4967
SGD Classifier	0.68	0.56	0.62	0.54	903.1595
Gradient Boosting Classifier	0.94	0.97	0.74	0.81	4.1431
K-Means	0.85	0.44	0.48	0.46	5.7211
K-Medoids	0.76	0.59	0.65	0.6	3.2946
Isolation Forest	0.67	0.53	0.55	0.51	7.7038
Local Outlier Factor	0.77	0.57	0.61	0.58	6.3967

Tabla D.11: Resultados para todo el dataset con 7 parámetros

Apéndice E

Fichero README sobre la implementación

En este anexo se incluye el documento "README", en el que se detalla la implementación realizada en Python y los ficheros generados.

El código se ha implementado con Python 3.11, y el entorno de desarrollo utilizado es Jupyter Notebook. Todos los scripts de este trabajo tienen este formato, e incluyen comentarios acerca del código.

Los ficheros Jupyter Notebook se dividen en dos carpetas, uno por cada dataset utilizado (CICIDS2017 y UNSW-NB15). En cada una de estas carpetas el código se organiza en ficheros de forma completamente análoga, aunque para cada dataset el código implementado es sensiblemente distinto.

En los ficheros **CICIDS_preproc.ipynb** y **UNSW-NB15_preproc.ipynb** se incluye el código necesario para el preprocesamiento inicial de los datos, el análisis estadístico de los mismos, y el estudio de la importancia de los parámetros en cada caso. Tras la ejecución de este código, se habrán generado los ficheros CSV (tanto para cada tipo de ataque como para el conjunto total de datos, en ambos casos ambos ya preprocesados) que se utilizarán en las siguientes fases de ejecución.

En los ficheros **CICIDS_attacks_sup.ipynb** y **UNSW-NB15_attacks_sup.ipynb** se incluye el código para la implementación de los sistemas de detección de anomalías distinguiendo por cada tipo de ataque, utilizando 12 algoritmos distintos de aprendizaje automático supervisado. Para poder ejecutar este código, es necesario que se hayan generado los ficheros CSV por cada tipo de ataque. Al ejecutar las celdas de código de este fichero, se obtendrán los resultados de rendimiento por cada tipo de ataque y cada algoritmo.

En los ficheros **CICIDS_attacks_unsup.ipynb** y **UNSW-NB15_attacks_unsup.ipynb** se incluye el código para la implementación de los sistemas de detección de anomalías distinguiendo por cada tipo de ataque, utilizando 4 algoritmos distintos de aprendizaje automático no supervisado. Para poder ejecutar este código, es necesario que se hayan generado los ficheros CSV por cada tipo de ataque. Al ejecutar las celdas de código de

este fichero, se obtendrán los resultados de rendimiento por cada tipo de ataque y cada algoritmo.

En los ficheros **CICIDS_all_sup.ipynb** y **UNSW-NB15_all_sup.ipynb** se incluye el código para la implementación de los sistemas de detección de anomalías sobre el conjunto total de datos, utilizando 12 algoritmos distintos de aprendizaje automático supervisado. Para poder ejecutar este código, es necesario que se hayan generado los ficheros CSV con todos los datos preprocesados. Se distinguen a su vez dos configuraciones distintas de parámetros, una combinando los parámetros más importantes por cada tipo de ataque (18 parámetros para CICIDS2017 y 12 para UNSW-NB15), y otra seleccionando los 7 más importantes sobre el conjunto total. Al ejecutar las celdas de código de este fichero, se obtendrán los resultados de rendimiento para ambos enfoques y cada algoritmo.

En los ficheros **CICIDS_all_unsup.ipynb** y **UNSW-NB15_all_unsup.ipynb** se incluye el código para la implementación de los sistemas de detección de anomalías sobre el conjunto total de datos, utilizando 4 algoritmos distintos de aprendizaje automático no supervisado. Para poder ejecutar este código, es necesario que se hayan generado los ficheros CSV con todos los datos preprocesados. Se distinguen a su vez dos configuraciones distintas de parámetros, una combinando los parámetros más importantes por cada tipo de ataque (18 parámetros para CICIDS2017 y 12 para UNSW-NB15), y otra seleccionando los 7 más importantes sobre el conjunto total. Al ejecutar las celdas de código de este fichero, se obtendrán los resultados de rendimiento para ambos enfoques y cada algoritmo.

Todos los ficheros aquí descritos pueden encontrarse en el siguiente enlace de GitHub:
https://github.com/PabloHG01/Anomaly-detection-in-networks-using-machine-learning.git