UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN




# "ANÁLISIS PROACTIVO DEL TRÁFICO DE RED PARA LA DETECCIÓN Y RESPUESTAS DE INTRUSIONES UTILIZANDO MACHINE LEARNING."

TRABAJO FIN DE GRADO

Febrero - 2026

AUTOR: Juan Antonio Carlos Balsalobre

DIRECTOR/ES: Pablo José Piñol Peral

# ÍNDICE

## ÍNDICE DE FIGURAS

## ÍNDICE DE TABLAS

# RESUMEN

Debido al reciente auge de la inteligencia artificial y el mayor número de personas conectadas, las tecnologías web y métodos que emplean datos en la red, están teniendo cada vez un tráfico creciente, sin señal de que vaya a disminuir en ningún momento. De la misma forma, ha aumentado el número de ciberdelincuentes y es más común emplear métodos maliciosos para sacar provecho de todos los datos vulnerables que circulan en la red, por lo tanto, el papel de la ciberseguridad está en un plano de mayor importancia, siendo necesario ahora más que nunca.

En este trabajo se ha empleado el uso de inteligencia artificial para el análisis de datos en el tráfico de red, buscando patrones de comportamiento en los datos que puedan generar tráfico malicioso, empleando algoritmos que predicen la naturaleza de una conexión basándose en los datos que forman dichas conexiones. Para ello ha sido necesario el estudio del conjunto de datos (*dataset*) UNSW-NB15 y CIC-IDS2017, los cuales están constituido por un conjunto de conexiones como fuente principal de aprendizaje utilizada por los modelos de inteligencia artificial, seguido de la monitorización de los datos analizados a nivel de conexión para poder determinar si los valores que presentan constituyen un ataque o no.

Consiguientemente, se ha desarrollado un programa que a modo de detección de intrusiones tiene la función de monitorización, predicción de ataques e interpretación del tráfico de red, garantizando al usuario una serie de información de gran utilidad del tráfico que esté cursando en tiempo real, determinando, con ello, si ha habido una anomalía en las conexiones que haya procesado la interfaz de red que se esté empleando, pudiendo enviar esta información al usuario, permitiendo compartir los resultados de estas conexiones para su análisis o para una mayor transparencia en los datos que transcurren en la red.

Finalmente, se ha determinado cuáles son los parámetros más determinantes a la hora de detectar anomalías en el tráfico de red y cuál es la varianza que presentan los datos que forman dichos ataques conforme a conexiones normales, pudiendo clasificar el tráfico normal con el malicioso. También ha sido estudiado cuáles son los ataques más comunes

para poder establecer una serie de reglas en dichos parámetros de datos monitorizados y así evitar realizar conexiones no deseadas.

**Palabras clave**: Sistema de detección de intrusiones, ciberseguridad, inteligencia artificial, monitorización, predicciones.

**Abreviaciones:**

**IDS**: Sistema de Detección de Intrusiones

**TFG**: Trabajo de Final de Grado

**LAN**: Local Area Network

**AI:** Artificial Intelligence

**DL:** Deep Learning

**IDS**: Intrusion Detection System

**NN:** Neural Network

**SSH** Secure Shell

**TFG:** Trabajo Final de Grado

**TTL:** Time To Leave

**KNN**: K-Nearest Neighbors

**DT**: Decision Tree

**SVC:** Support Vector Classifier

**NV:** Naive Bayes

**RF:** Random Forest

# CAPÍTULO 1. INTRODUCCIÓN Y CONTEXTO

## 1.1 INTRODUCCIÓN

El uso de las tecnologías en el día a día es inevitable y cada vez son más personas las que necesitan utilizar algún tipo de tecnología que requiera conexión a internet, conllevando una exposición a ser atacado o simplemente que en el tráfico de red al que se haya conectado sufra una intrusión no deseada. Como la exposición a posibles conexiones maliciosas por parte de atacantes es cada vez mayor, para poder garantizar una seguridad en este medio es necesario para el funcionamiento de un sistema como es internet, apelar a la ciberseguridad, ya que juega un papel crucial para encargarse de que miles de usuarios puedan navegar libremente sin ser víctimas de tráfico malicioso.

Por lo tanto, para reforzar la forma en la que se pueda defender la red, entra el papel de la inteligencia artificial y la ciencia de datos, siendo su función principal analizar los patrones de datos que forman cada conexión y qué relación guardan entre ellos. La inteligencia artificial es capaz de generar modelos predictores que, en base a un gran volumen de datos, generen relaciones entre esos mismos datos para poder identificar si una conexión es maliciosa o no.

La inteligencia artificial se enfoca más en el apartado estadístico de análisis de los datos que componen conexiones tanto normales como de ataques. De esta forma se puede determinar que campos de una conexión son más determinantes que otros y cuáles son las diferencias entre los valores que tienen una conexión normal y una conexión maliciosa, por ejemplo, el número de paquetes enviados por el emisor y el número de paquetes recibidos, la tasa de bits del emisor o la duración de la conexión. Todos estos datos han sido estudiados generando una serie de patrones o normas que dan lugar a determinar la naturaleza de las conexiones.

Este trabajo se centra en el estudio de las conexiones y cómo se comportan los datos que los conforman, se emplea también la monitorización de conexiones, dando lugar a una mayor consciencia sobre el tráfico cursado y aumentando el nivel de seguridad una vez se hayan empleado los modelos de inteligencia artificial.

Se ha programado, mediante el lenguaje de programación Python, el sistema de detección de intrusiones IDS, que engloba y pone en práctica todos los conceptos mencionados anteriormente, garantizando al usuario de dicho programa una mayor seguridad mientras esté conectado, o siendo una herramienta para monitorizar y *loguear* el tráfico cursado y mantener un registro con el que pueda consultar para saber cómo ha sido el tráfico de red en las sesiones de uso donde haya habido conexión a internet.

## 1.2 JUSTIFICACIÓN DEL TRABAJO

La justificación o motivación principal del trabajo es investigar cómo puede la inteligencia artificial utilizar los datos que forman las conexiones para poder aplicarlos en el ámbito de ciberseguridad, generando una herramienta eficiente que pueda resultar de utilidad, proporcionando, con ello, mayor seguridad al recibir tráfico de internet.

Otra justificación surge de la necesidad de herramientas accesibles a nivel de usuario, pudiendo monitorizar el tráfico de red y aumentar la seguridad en el uso diario. La mayoría de las herramientas que permiten monitorizar conexiones no son accesibles para usuarios con conocimientos bajos sobre de informática, no siendo capaces de llevarlos a cabo ni de procesar la información que proporcionan.

La motivación final es la de generar un programa a modo de herramienta para que esté publicado online, para que otras personas puedan acceder a ella y utilizarla para uso de manera gratuita, de forma que puedan dar su enfoque de cómo ha sido su experiencia, dándole uso y matizando las mejoras que podría llegar a tener. Se ha empleado el uso de GitHub como medio de publicación online donde está disponible para la descarga y uso de manera gratuita.

## 1.3 OBJETIVOS GENERALES Y ESPECÍFICOS

El propósito fundamental de este trabajo consiste en la creación de una aplicación basada en un sistema de detección de intrusiones. Esta herramienta tendrá como objetivo principal la monitorización de los paquetes de red asociados a las conexiones que el usuario establezca, permitiendo diferenciar aquellos que representen una potencial amenaza de los que no. El desarrollo completo de la aplicación se llevará a cabo utilizando

el lenguaje de programación Python, aplicando los conocimientos adquiridos durante el transcurso del grado.

De esta forma, al ejecutar la aplicación, será posible observar en tiempo real el comportamiento y los parámetros generados durante una sesión de usuario, incluyendo actividades como la navegación web, recepción de archivos o el procesamiento de conexiones a un servidor, entre otras. La aplicación también facilitará el análisis de conexiones pertenecientes a archivos PCAP.

Estos son los objetivos generales del proyecto, que guiarán su desarrollo y serán alcanzados por la aplicación final. Estos objetivos mantienen un alto nivel de abstracción y se desarrollarán en detalle a lo largo del trabajo:

- **Monitorización y Análisis de Red:** proporcionar una herramienta para la monitorización del tráfico de red, incluyendo el análisis de las características de cada conexión.
- **Estudio de Conexiones:** investigar y entender los campos de cada conexión de red, identificando su relevancia y función específica.
- **Clasificación de conexiones mediante inteligencia artificial:** implementar un modelo de inteligencia artificial que clasifique el tráfico de red cursado.
- **Visualización de Datos:** Ofrecer un *dashboard* o *frontend* a modo de interfaz de usuario que visualice los datos de manera ordenada y accesible para el usuario.

A continuación, se detallan los objetivos específicos, derivados de los objetivos generales previamente establecidos:

- **Implementación y Análisis del Tráfico de Red:** puesta en marcha de software para el análisis de tráfico de paquetes en la red, con el fin de identificar y registrar cada conexión individual.
- **Desarrollo de Scripts de Edición:** creación de scripts especializados para el preprocesamiento y la transformación de los datos de red obtenidos.
- **Entrenamiento y Comparación de Modelos de IA:** entrenamiento de múltiples modelos de inteligencia artificial utilizando conjuntos de datos de ciberseguridad

basados en ataques reales simulados en entornos controlados, y comparar los resultados.

- **Montaje de la API de Predicciones:** desarrollo de una Interfaz de Programación de Aplicaciones (API) para centralizar la comunicación de los datos en diversos *endpoints* (puntos de acceso), y que será utilizada para la realización de predicciones.

- **Configuración del Servicio Cliente-Servidor:** establecimiento de un servicio que permita la transmisión de datos utilizando una arquitectura cliente-servidor.

## 1.4 ALCANCES Y LÍMITES DEL PROYECTO

Los límites de este trabajo se encuentran en los datos a emplear, principalmente datos de entrenamiento y datos etiquetados con conexiones de red donde constan ataques. Estos datos no puedo obtenerlos yo personalmente y me baso en el trabajo de otras personas o instituciones que lo hayan recopilado previamente.



## 1.5 ESTRUCTURA DEL DOCUMENTO

El capítulo uno, contiene información referente a la motivación del proyecto y al ámbito que abarca. Esta sección contribuye a garantizar una idea general de porqué se ha elegido esta temática para el proyecto y cuáles son los objetivos que se propone conseguir, junto a los límites y alcance que pueda llegar a tener, sirviendo como una guía para saber superficialmente el contenido del proyecto.

En el capítulo dos, Marco teórico, se describe toda la información teórica y conceptual acerca de los temas escogidos en este trabajo que son la inteligencia artificial y la ciberseguridad.

El capítulo tres, metodología, se centra en describir todo el marco tecnológico en el que se basa el proyecto. En este capítulo se describen todas las tecnologías que se han empleado para desarrollar la aplicación, su aspecto técnico y el papel que han desempeñado en el funcionamiento del sistema de intrusiones.

El capítulo 4, contiene el desarrollo del proyecto donde incluye la descripción detallada de todos los procesos, procedimientos y funciones, con sus diagramas de flujo. Se explica técnicamente la arquitectura del sistema, el papel de las tecnologías del capítulo 3 y el funcionamiento de los componentes.

El capítulo cinco, se dedica a la inteligencia artificial, describiendo el proceso de entrenamiento de modelos y los resultados obtenidos con distintos algoritmos.

El Capítulo seis, se dedica íntegramente a la ciberseguridad, poniendo a prueba el funcionamiento técnico previo. Aquí se ejecuta el programa completo, realizando diversas pruebas en un entorno simulado con tráfico real para evaluar su rendimiento y correcto funcionamiento.

En el Capítulo siete, se evalúan los resultados finales y se comparan con los objetivos propuestos en el capítulo uno. También se discuten los puntos positivos y negativos, junto con posibles mejoras orientadas al futuro.

En el capítulo ocho, se muestran todas las referencias sobre estudios, páginas webs, artículos revisados en el desarrollo de este TFG. Es el apartado correspondiente a la bibliografía.

Finalmente, el capítulo nueve, lo forma el anexo, con toda la información adicional, donde se muestran procesos técnicos o tablas.

# CAPÍTULO 2. MARCO TEÓRICO Y ESTADO DEL ARTE

## 2.1 MACHINE LEARNING (APRENDIZAJE AUTOMÁTICO) Y SU ENFOQUE PRÁCTICO

Gran parte del marco teórico de este proyecto se basa en el conocimiento sobre machine learning y la ciencia de datos, tanto en sus fundamentos como en el conocimiento sobre los algoritmos que lo componen.

El aprendizaje automático o Machine learning (ML) es un subconjunto perteneciente a la inteligencia artificial que se caracteriza principalmente por el aprendizaje autónomo de los computadores, o mejor dicho, la creación de reglas y/o patrones en un modelo o algoritmo de inteligencia artificial sin la necesidad de la intervención humana, mediante grandes volúmenes de datos. Como he mencionado, el aprendizaje automático significa que los algoritmos son capaces de generar sus propias reglas lógicas a diferencia de programas convencionales donde han de ser programados mediante algoritmos, reglas o subrutinas desarrollados explícitamente por una persona. Un ejemplo son los algoritmos llamados árboles de decisión, donde el algoritmo empieza vacío desde el nodo raíz y va generando reglas o ramas hasta llegar al nodo final. Esta serie de reglas se han generado autónomamente por el computador sin la necesidad de intervención humana, de ahí el nombre de aprendizaje automático. Se utiliza el término “modelo” para referirse a un algoritmo de aprendizaje automático ya entrenado. (Mitchell, 1997; Russell & Norvig, 2020)

En la siguiente figura se pueden apreciar los diferentes subconjuntos que forman la inteligencia artificial, este proyecto se centra más en aprendizaje automático, sin embargo, se realiza un comentario breve respecto a los otros.

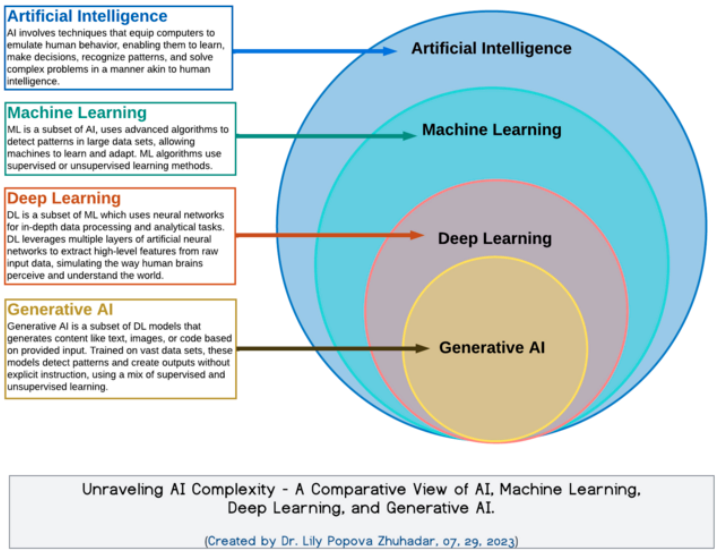



*Figura 2.1*. Diagrama representante de los diferentes conjuntos de inteligencia artificial.

*Nota*. De *Unraveling AI Complexity: A Comparative View of AI, Machine Learning, Deep Learning, and Generative AI* [Diagrama], por LiLy Popova Zhuhadar, 2023, Wikimedia Commons, (https://commons.wikimedia.org/wiki/File:Unraveling_AI_Complexity_-_A_Comparative_View_of_AI,_Machine_Learning,_Deep_Learning,_and_Generative_AI.png), CC BY-SA 4.0.

El proceso general de aprendizaje automático se basa en dos fases principales compuestas por entrenamiento y pruebas o *testeo*. Para poder realizar el proceso de aprendizaje se necesitan datos de *entrenamiento*, que puedan servir como referencia al modelo para generar reglas o patrones en base a este conjunto de datos dado. Una vez se haya pasado la fase de entrenamiento, dará lugar la fase de *testeo* o pruebas, donde se evaluará el rendimiento del algoritmo o se determinarán características en base los resultados obtenidos en esta fase. (Mitchell, 1997; Russell & Norvig, 2020)

### 2.1.1 TIPOS DE APRENDIZAJE

Existen diferentes metodologías o diferentes tipos de aprendizaje a la hora de entrenar modelos de inteligencia artificial, estos son los principales o más comunes:

- **Aprendizaje supervisado** (SL): Este tipo de aprendizaje se forma cuando se tienen etiquetas en los datos de entrenamiento. Las etiquetas son el resultado de una clasificación o regresión de una instancia y el valor a predecir que tiene que averiguar el modelo, es decir, el modelo se entrena sabiendo ya el resultado que

debe obtener para una serie de datos. Los atributos o características son las reglas que se ciñen en torno a esta serie de respuestas que son las etiquetas.

Por ejemplo, en un conjunto de datos que consiste en un listado de casas para su venta, las etiquetas son el valor del precio de cada casa en euros. El modelo creará una serie de reglas en base a los precios que tengan en el listado de casas a la venta, de forma que en el futuro pueda predecir el precio de una casa en base a unas características dadas como el número de habitaciones o los metros cuadrados de la vivienda.

El concepto o ventaja principal que supone utilizar un método de aprendizaje supervisado es la posibilidad de determinar cómo de preciso es el modelo que se esté entrenando, pudiendo determinar si es correcto para su uso o no. Las etiquetas proporcionan información real a comprobar frente al valor predicho en los *test*, esto se utiliza como métrica para determinar el número de instancias correctas que ha efectuado el modelo y si está listo o no para su uso. (Géron, 2019)

Dentro de los modelos aprendizaje supervisado se encuentra clasificación y regresión como los más populares y métodos efectivos:

- **Clasificación**: Es la función de determinar a qué tipo de categoría pertenece una serie o asignación de datos, según el tipo de etiquetas y siguiendo valores categóricos y discretos, los datos serán clasificados según venga especificado en dichas etiquetas, por ejemplo, en este trabajo los datos según el campo que equivale al tipo de ataque, cuando se realice una predicción podrán ser etiquetados según al ataque que corresponda.

- **Regresión logística**: Es un mecanismo alternativo a la clasificación para calcular probabilidades. Utilizado cuando se requiere una estimación de probabilidad como resultado. En términos prácticos se puede usar la probabilidad de forma binaria (*True*, *False*) o de forma numérica (un número). En este trabajo no se aborda en casos prácticos el uso de regresión logística, limitándose únicamente a algoritmos de clasificación. (Géron, 2019)

Ejemplos de algoritmos que emplean el aprendizaje supervisado:

- Árboles de decisión (*Decision Tree*).
- K-Nearest Neighbours.
- Máquinas de soporte vectorial (Support Vector Machines). (Géron, 2019)

- **Aprendizaje no supervisado** (UL): Los modelos encuentran patrones en datos sin etiquetar (*unsupervised methods*). En este caso es el propio modelo el encargado de designar una organización de los datos formantes. A la hora de su empleo existen dos categorías para su realización: agrupación (*clustering*) y reducción de dimensionalidad. El método de *clustering* se encarga de agrupar en *clusters* que son formados por datos similares. Por otro lado, la reducción global consiste en eliminar los atributos poco relevantes del conjunto, de forma que se evita cierta redundancia en los datos.

  Sin embargo, los algoritmos de aprendizaje no supervisado se entrenan o generan reglas en base a la estructura inherente que forman los datos y no dependen de ningún tipo de etiquetas o valores ya establecidos. Los algoritmos son adecuados para los conjuntos de datos donde no se puedan etiquetar las instancias que conforman dicho conjunto de datos, entonces el aprendizaje no supervisado realiza muy buen trabajo generando reglas y patrones en los datos que no tienen etiquetas.

  En métodos de aprendizaje no supervisado se determinan las agrupaciones o *clusters* mediante una medida de similitud, o la métrica que se determine para comparar muestras. (Bishop, 2006)

  Ejemplos que componen el aprendizaje no supervisado:

  - *K-means*,
  - Agrupamiento jerárquico. (Bishop, 2006)

- **Aprendizaje semisupervisado** (SSL): Este tipo de aprendizaje es una mezcla entre los dos mencionados anteriormente. Combina los atributos clasificados y no clasificados. Generalmente suelen presentar una cantidad pequeña de datos etiquetados, junto a una gran cantidad de datos no etiquetados. Al ser una mezcla

entre datos etiquetados y no etiquetados, su entrenamiento es más complicado dando lugar a una menor frecuencia de uso, siendo menos utilizados.

La razón por la que el aprendizaje semisupervisado es menos utilizado es debido a su falta de precisión respecto a sus alternativas. Sin embargo, los métodos que lo forman tienen un menor coste computacional, haciéndolo una opción viable cuando se considera oportuno y el volumen de datos no es muy grande o no se dispone de un equipo computacionalmente eficiente. (Zhu, 2005)

- **Aprendizaje por refuerzo** (RL): Es un modelo de aprendizaje donde los modelos aprenden a tomar decisiones a través de una metodología que imita *recompensas* y *penalizaciones*. De esta forma, aprenden sin la necesidad de que un usuario tenga que programar decisiones o reglas para alcanzar un objetivo determinado.

  Para realizar el proceso de aprendizaje se tiene al agente (se llama de esta forma al modelo de IA en este tipo de aprendizaje) donde toma decisiones. Si el agente realiza una acción que se considere como correcta o válida, este será recompensado obteniendo una recompensa o una puntuación positiva. Si realiza una acción incorrecta, será penalizado recibiendo una puntuación negativa. De esta forma, el modelo "aprenderá" en base a si realiza acciones consideradas correctas e incorrectas, y poco a poco irá refinando su precisión de esta forma. (Sutton & Barto, 2018)

- **Deep learning (DP) o aprendizaje profundo:** tipo de aprendizaje que tiene su propio subconjunto dentro del aprendizaje automático y se caracteriza por replicar el funcionamiento del sistema nervioso y el comportamiento neuronal conforme a su estructura biológica. El aprendizaje está formado por capas compuestas de neuronas, donde cada una de ellas se comunican entre sí formando relaciones entre cada neurona o nodo. Las redes neuronales tienen diferentes capas donde cada una de ellas tiene una función distinta y se relaciona con la capa respectiva de su siguiente nivel o anterior, de esta forma envían y reciben datos hasta llegar a la capa final y obtener el resultado de la predicción.

Se caracterizan por poder obtener un gran nivel de precisión en conjuntos grandes de datos donde otro tipo de modelos no podrían alcanzar, especialmente en conjuntos muy grandes, llegando poder clasificar diferentes tipos de formatos como imágenes, o incluso clasificar diferentes tipos de abstracciones. Se ha de destacar que su funcionamiento sobre datos pequeños no es eficiente y requieren de un gran poder computacional para su entrenamiento, comparado con otros algoritmos. (Goodfellow et al., 2016)

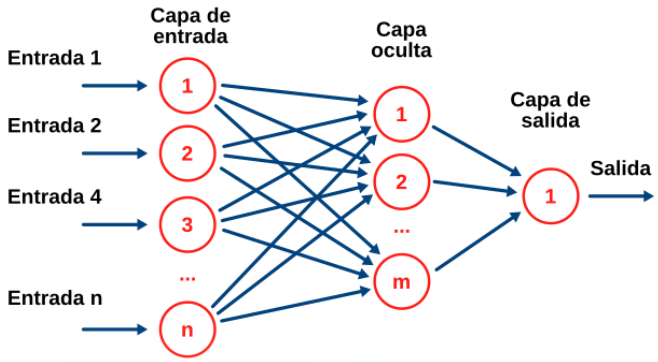



*Figura 2.2.* Diagrama ejemplificando una red neuronal.

*Nota*. De *Red neuronal artificial.svg* [Diagrama], por Antimundo, 2026, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Red_neuronal_artificial.svg). CC BY-SA 4.0.

### Enfoque utilizado en este proyecto

Como los conjuntos de datos que recopilan tráfico de red presentan etiquetas con el tipo de conexión establecida, se han entrenado modelos de aprendizaje supervisado basados en clasificación. Al tener etiquetas, se podrá obtener la matriz de confusión con la que se obtendrán las métricas de rendimiento sobre los datos de entrenamiento que se procesarán en la fase de pruebas, pudiendo ver cuántas clasificaciones correctas ha hecho el modelo. Esta parte se explicará en más profundidad en el capítulo 4.

### 2.1.2 ALGORITMOS DE MACHINE LEARNING EMPLEADOS EN ESTE PROYECTO

Existen diferentes tipos de modelos de inteligencia artificial dentro del aprendizaje supervisado, cada uno de ellos tiene sus propias reglas y características únicas que los convierten en mejores candidatos según el tipo de situación. En este trabajo se han escogido una serie de algoritmos que se puedan adaptar al contexto de ciberseguridad, que sean capaces de realizar predicciones que den como resultado un valor categórico, siendo este valor el tipo de conexión realizada. (Géron, 2019)

**Random Forest**

Random Forest es uno de los modelos de inteligencia artificial que ofrecen mejores resultados en cuanto a clasificación dentro del aprendizaje supervisado, Random Forest fue finalmente desarrollado por Leo Breiman y Adele Cutler. Este algoritmo pertenece a la familia de agrupamiento (*ensemble*). Esto quiere decir que su funcionamiento proviene de varios árboles de decisión generados del conjunto original, formando así muchos subconjuntos aleatorios llamados *bootstrap* en inglés, donde cada árbol se entrena individualmente respecto a los datos de entrenamiento para finalmente generar una solución que será sometida a votación por mayoría (para clasificación) o promedio (para regresión). Esto se conoce como *bagging* (*Bootstrap aggregating*). De las predicciones individuales de todos los árboles del agrupamiento. Esta estrategia de combinación reduce significativamente la varianza y el sobreajuste característicos de los árboles de decisión individuales. (Breiman, 2001)

En la siguiente figura se ve claramente el funcionamiento de Random Forest:

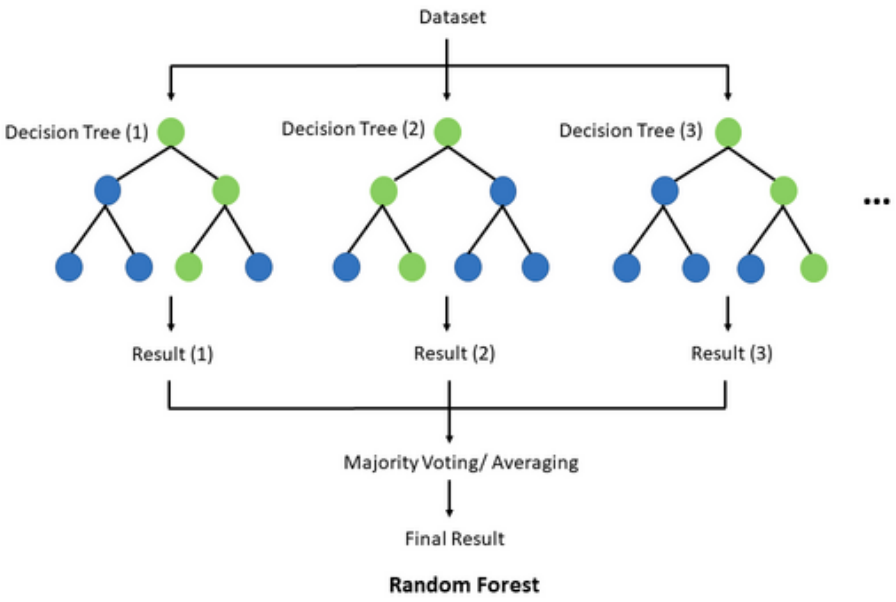



*Figura 2.3*. Diagrama sobre el funcionamiento de Random Forest.

*Nota*. De *Random forest explain* [Diagrama], por TseKiChun, 2017, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Random_forest_explain.png). CC BY-SA 4.0.

**Decision Tree**

Es un algoritmo de aprendizaje supervisado está conformado por una parte de Random Forest, su funcionamiento consiste en ir generando una serie de reglas en base a un criterio predefinido, como la entropía o desorden de clase o la impureza de gini. El funcionamiento de un árbol de decisión consiste en iniciar el nodo raíz por el atributo que mejor puntuación tenga respecto al criterio seleccionado. Una vez elegido el nodo raíz, se establecen condiciones por cada rama donde se han formado reglas por cada iteración, dando lugar a nodos condicionales. Estos nodos son atributos escogidos otra vez según su puntuación del criterio seleccionado (igual que el nodo raíz), y se va iterando, formando nuevas reglas hasta que no haya ningún atributo que satisfaga dicho criterio y se llega al nodo hoja (*Leaf node*). (Breiman et al., 1984)

Esto corresponde los criterios mencionados para seleccionar un atributo en cada iteración:

**Impureza de Gini**: Mide la probabilidad de clasificar incorrectamente un elemento elegido aleatoriamente:

- Su fórmula es la siguiente: $G = 1 - \sum_{i=0}^{n} \Sigma\,(p_i)^2$
- Donde p es la probabilidad de que aparezca un elemento en cada clase. (Breiman et al., 1984)

**Entropía**: Mide el grado de desorden o falta de homogeneidad de un atributo:

- Su fórmula es la siguiente: $H = -\sum_{i=0}^{n} \Sigma\, p_i * log^2(p_i)$
- Donde al igual que en gini, pi es la proporción de la clase, y se utiliza el logaritmo en base 2 porque la información se mide en bits. (Shannon, 1948)

En la siguiente figura se puede apreciar un árbol de decisión sobre si se debe cancelar un partido de fútbol o no:

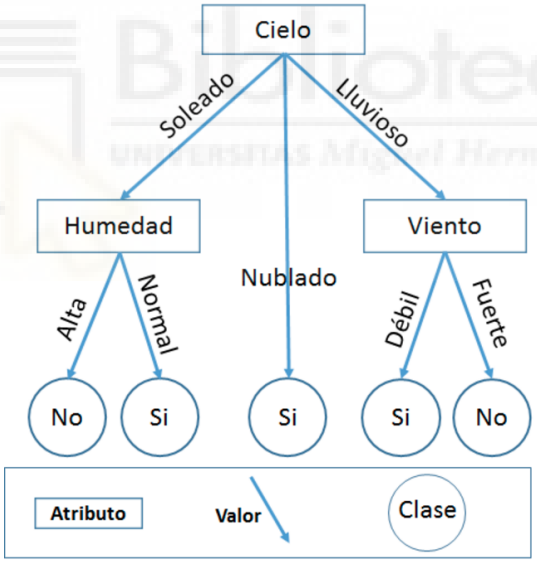



*Figura 2.4*. Diagrama de un árbol de decisión.

*Nota*. De ÁRBOL DE DECISIONES SIMPLE.png [Diagrama], por Evolutions algoritmos, 2019, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:%C3%81RBOL_DE_DECISIONES_SIMPLE.png). CC BY-SA 4.0.

**Máquinas de soporte vectorial (*Support Vector machines*):**

Las máquinas de soporte vectorial son algoritmos diferentes a los anteriores ya que no generan ningún tipo de reglas para llegar a una solución, sino que clasifican una instancia en base a una serie de valores y el resultado de la predicción de esa instancia dependerá de donde esté situada en el plano o en el hiperplano. Este tipo de modelo normalmente suele entrenarse para clasificación binaria, sin embargo, también se puede entrenar para clasificación multiclase donde su complejidad aumenta considerablemente.

El parámetro o elemento más importante que separa cada clase, se llama margen, representa la distancia más amplia posible entre las clases separadas en el hiperplano, delimitando los valores que las clasificaciones pueden llegar a tener. Los valores muy cercanos al margen se llaman valores extremos o *outliers*. Por ejemplo, un margen determina la distancia que separa una instancia de ser ataque o no. Es de crucial importancia determinar correctamente el margen para que se puedan delimitar correctamente los datos de cada atributo y que pueda tratar *outliers* correctamente. (Cortes & Vapnik, 1995).

En la siguiente figura, se puede apreciar un diagrama donde se aprecia el margen w, con b, que es el valor de la intersección de la recta de eje y cuando x=0 y x el valor del array de datos de entrenamiento.

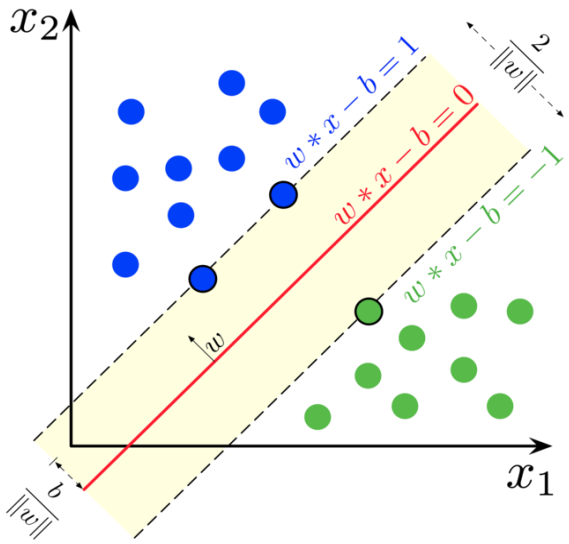



*Figura 2.5*. Diagrama sobre el funcionamiento de *Support Vector Machine.*

*Nota*. De *SVM margin.png* [Diagrama], por Larhmam, 2018, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:SVM_margin.png). CC BY-SA 4.0.

Se puede apreciar que se separan dos tipos de clase (verde y azul), dependiendo del valor que se haya dado dentro del plano, sin embargo, no todos los modelos tienen únicamente dos dimensiones, estos serían los casos cuando el modelo tenga más de dos atributos en el vector x:

- **En un espacio bidimensional** (dos características), el hiperplano es simplemente una línea recta que separa los puntos de las dos clases.
- **En un espacio n-dimensional** (n características), el hiperplano se convierte en un plano que corta el espacio en n-1 regiones.
- Cuando se trabaja con más de tres dimensiones, aunque se puede visualizar fácilmente, el hiperplano sigue siendo una superficie que divide el espacio de manera óptima. (Cortes & Vapnik, 1995).

La formulación matemática se corresponde con:

$$w^T x + b = 0$$

- **w**: vector de pesos (*weights*), es perpendicular al hiperplano al ser un vector transpuesto, representa la dirección del hiperplano).
- **b**: es el sesgo (*bias*) o término independiente, determina la distancia del hiperplano desde el origen.
- **x**: vector con muestra de características de datos del *dataset* (Cortes & Vapnik, 1995)

**K-Nearest Neighbors**

El algoritmo K-Nearest Neighbors (KNN) proviene de la familia de algoritmos que funcionan mediante agrupamiento por proximidad o vecinos (*neighbors*). Estos se caracterizan por tomar decisiones en base a puntos próximos en los datos de prueba o *test*. La predicción depende de la proximidad que tengan o la distancia que haya entre los puntos de los datos. Este algoritmo es la versión donde se configura manualmente el número de vecinos a emplear cuando se intente tomar valores próximos, de ahí viene la k en el nombre.

Cabe destacar que este algoritmo no realiza un proceso de entrenamiento explícito, ya que almacena los datos y efectúa el aprendizaje en el momento de la predicción. Debido a esta característica, KNN puede ser sensible al ruido y si se elige un valor de k demasiado pequeño, puede presentar sobreajuste (*overfitting*). Sin embargo, con una elección adecuada de k y una correcta normalización de los datos, el algoritmo puede generalizar correctamente (Cover & Hart, 1967).

En la siguiente figura se muestra una instancia que va a ser clasificada, representada por un punto verde con una interrogación, dependiendo del valor de k el modelo realizará una predicción según el número de vecinos más próximos al punto verde, en este caso si k=2, la predicción resultaría ser de la clase correspondiente a los triángulos rojos mientras que si k=1 o k=3, la predicción resultaría en la clase representada por cuadrados azules, debido al número de instancias más próximas a la actual.

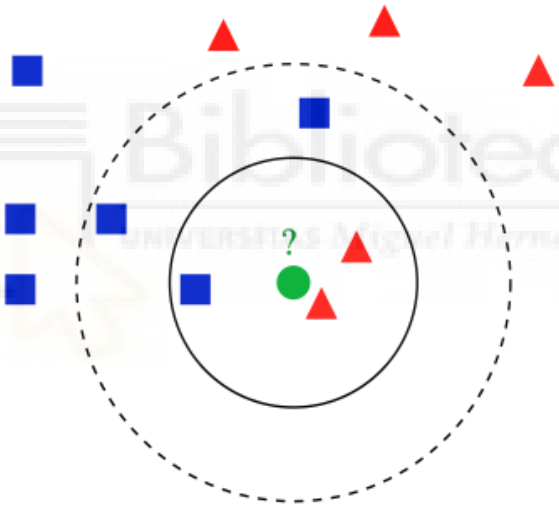



*Figura 2.6*. Diagrama ejemplificando el funcionamiento de KNN.

*Nota*. De *KnnClassification.svg* [Diagrama], por Antti Ajanki, 2016, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:KnnClassification.svg). CC BY-SA 4.0.

Para medir distancias y determinar cuál será el vecino más cercano se pueden utilizar diferentes formas de medición, sin embargo, estas son las más comunes, aquí está su formulación matemática (Cover & Hart, 1967)

Distancia euclídea (*Euclidean Distance*):

$$d_{euclidia(x,y)} = \sqrt{[\Sigma_{\mathrm{i}}{}^{1\mathrm{n}} (x_{\mathrm{i}} - y_{\mathrm{i}})^2]}$$

Distancia de Manhattan (*Manhattan Distance*)

$$d_{manhattan(x,y)} = \Sigma_{\mathrm{i}}{}^{=1\mathrm{n}} |x_{\mathrm{i}} - y_{\mathrm{i}}|$$

Distancia de Minkowski (*Minkowski Distance*)

$$d_{minkowski(x,y)} = (\Sigma_{\mathrm{i}}{}^{=1\mathrm{n}} |x_{\mathrm{i}} - y_{\mathrm{i}}|^{\mathrm{p}})^{\frac{1}{p}}$$

**Naive Bayes**

Finalmente, el último algoritmo empleado en este proyecto es Naive Bayes. Su funcionamiento se basa en cálculos de probabilidad estadística, específicamente en el teorema de Bayes, que trata sobre la probabilidad de que ocurra un evento, condicionada por la ocurrencia de otro evento previo. El término *naive* (ingenuo) se debe a que el algoritmo asume que todas las probabilidades son completamente independientes entre sí, sin generar reglas entre clases. Esta simplicidad resulta en un modelo de baja complejidad. A pesar de esto, Naive Bayes es eficaz en contextos específicos, como los detectores de *spam*, donde puede ofrecer un alto porcentaje de precisión. (Geron, 2019; Rish, 2001).

**Teorema de bayes original:**

$$P(A|B) = \frac{P(B|A)P(B)}{P(A)}$$

El suceso a predecir, denotado como A, representa la clase de la conexión. Este suceso A, está condicionado por la probabilidad del suceso B, que a su vez se corresponde con los valores de los atributos que constituyen los datos de dicho registro. En este proyecto, se calculará la probabilidad de ocurrencia de cada tipo de ataque en función de los valores de los atributos de la conexión. (Bayes & Price, 1763)

## 2.2 CIBERSEGURIDAD

Una vez analizado el fundamento teórico de la inteligencia artificial relevante para este trabajo, es imprescindible establecer una base teórica de ciberseguridad. Esto permitirá comprender el alcance de la ciberseguridad abordado en el estudio y su relación con la inteligencia artificial.

### 2.2.1 SISTEMAS DE DETECCIÓN DE INTRUSIONES (IDS)

Según el Instituto Nacional de Ciberseguridad (INCIBE, 2017), un sistema de detección de intrusiones (IDS) es un tipo de *software* cuya finalidad sea detectar instancias de *software* malicioso en una conexión o un archivo, mediante el análisis del tráfico de red. También comunica o envía alertas de esa información al supervisor correspondiente para alertar de un posible ataque.

Los IDS se implementan frecuentemente en redes de área local (LAN) con el fin de añadir una capa extra de seguridad en redes cuyo ámbito es privado, pertenecientes a instituciones como empresas o instituciones gubernamentales que contengan información confidencial (alternativamente, pueden usarse a modo de host para el uso doméstico). También ejercen la función de complementar otras medidas de seguridad, proporcionando un monitoreo continuo y detección temprana de amenazas.

Un IDS realiza dos tareas fundamentales:

- **Análisis y monitorización**: se realiza mediante *software* que permite registrar y analizar el tráfico de red.
- **Detección y alerta**: se intenta determinar patrones de intrusión en los datos recibidos por el *software* de análisis de red. Las funciones de *machine learning* enriquecen mucho esta función al poder detectar patrones en los datos analizados que no podrían ser detectados de otra forma. (INCIBE, 2017)

**Diferentes tipos de IDS**

Existen varios tipos de IDS principales dependiendo de su localización en topología de la red y según el método empleado para realizar detecciones.

**Tipos de IDS según su enfoque (anomalías o firmas)**

Según el Instituto Nacional de Ciberseguridad (INCIBE, 2017), existen diferentes tipos de IDS clasificados según cómo se procesan las conexiones y cómo se detectan las amenazas. Los dos métodos más comunes se centran en la detección de anomalías identificadas mediante inteligencia artificial o mediante conjuntos de reglas heurísticas basados en una serie de firmas (*signatures*) o reglas.

- **Detección de anomalías:** Es el tipo de este proyecto. El IDS se centra en detectar comportamientos que se salgan de la norma, tienen como referencia el tráfico de red normal y ataques. Para lograr detectar comportamiento sospechoso en los datos, se pueden aplicar métodos o utilizar modelos de inteligencia artificial que detecten patrones inusuales en los datos.
- **Detección firmas (*signature based*):** Los IDS basados en detección de firmas se caracterizan por utilizar patrones ya preestablecidos como referencia para detectar ataques. Se tiene como referencia ataques ya formados y se utilizan estos patrones para compararlos con el tráfico que va llegando a la red. (INCIBE, 2017)

  Un método de uso es cuando el tráfico llega del cortafuegos, se compara con un firma o patrón de ataque ya establecido y se determina si puede acceder a la red o si es rechazado.

**Según su alcance (de host o red)**

- **HIDS (*Host-based Intrusion Detection Systems*):** Como indica su nombre, un HIDS opera sobre un dispositivo individual (*host*), como un servidor o una estación de trabajo. Su análisis se centra en el tráfico efectuado en ese sistema. Se encarga de monitorizar archivos de registro, como los generados por *syslog*. Un HIDS es eficaz para detectar ataques que tienen como objetivo un host específico y no son propagados por la red.

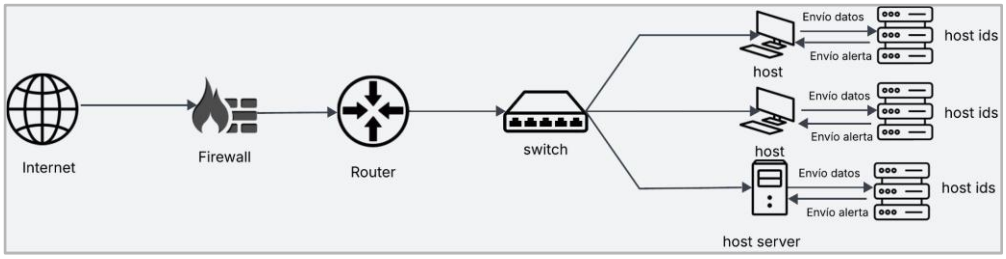



*Figura 2.7*: Diagrama con Topología de un host IDS.

- **NIDS (*Network Intrusion Detection System*):** Menos precisos ya que se encargan de monitorizar el tráfico que circula a través de la red global y no de un host específico. Son instalados de forma que monitorizan las conexiones del tráfico y no de un único host, analizando gran parte del tráfico que circula por la red, ofreciendo un mayor índice de seguridad global, pero siendo menos efectivo que los enfocados a host. (INCIBE, 2017)

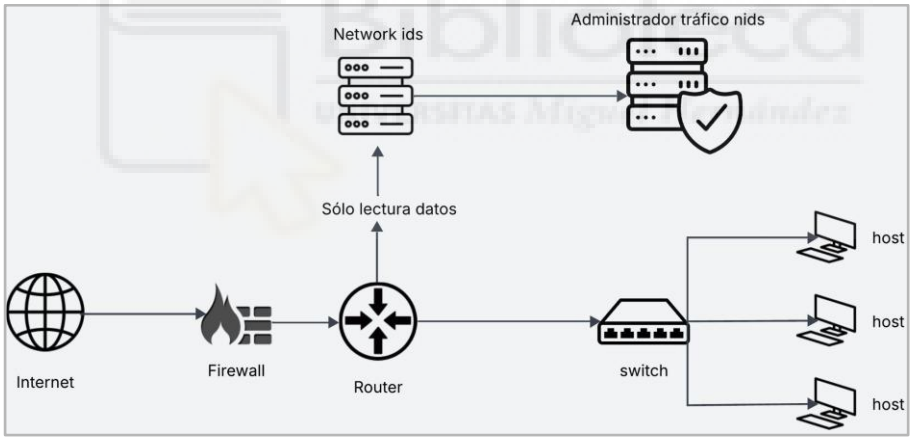



*Figura 2.8*: Diagrama con topología de un NIDS.

### 2.2.2 CONJUNTOS DE DATOS EMPLEADOS

Los conjuntos de datos o *datasets* en inglés, son la fuente principal de información que utilizan los algoritmos de aprendizaje automático para poder aprender y generar nuevas reglas o patrones. En este trabajo se han utilizado dos, UNSW-NB15 y CIC-IDS2017.

## UNSW-NB15

El conjunto de datos UNSW-NB15, desarrollado por el Australian Centre for Cyber Security (ACCS) en la Universidad de Nueva Gales del Sur (UNSW) en 2015, es una recolección fundamental en la investigación sobre detección de intrusiones en redes. Este conjunto de datos representa una mejora significativa respecto a otros conjuntos enfocados en ciberseguridad como KDD'99 y NSL-KDD al estar más actualizado que el resto.

### Características de UNSW-NB15

Como indican Moustafa y Slay (2015), el conjunto de datos UNSW-NB15 está compuesto por 6 apartados diferentes (*flow*, *basic*, *content*, *time*, *additional* y *labelled)*, formando un total de 49 atributos o características. Los apartados se dividen según lo que caracterice su contenido, por ejemplo, el apartado *Basic* está constituido por atributos que forman la parte de metadatos de la conexión. Por otro lado, apartados como *Content* aportan información centrada en los paquetes que forman el "cuerpo" de la conexión. (Moustafa y Slay, 2015)

El conjunto de datos presenta un desbalance considerable en varios apartados:

- Está compuesto en su mayoría por conexiones orientadas a protocolos TCP y UDP, con 771,488 y 301,528 registros respectivamente, mientras que conexiones del protocolo ICMP tan solo tiene 150 registros, quedando en el resto de protocolos tienen únicamente 150 conexiones.

- La relación entre el número de registros pertenecientes a ataques y conexiones normales es considerablemente grande, habiendo una cantidad de ataques mucho menor frente a conexiones normales. Del total de conexiones, solo 321,283 representan ataques frente a 2,218,761 de conexiones normales. (Moustafa y Slay, 2015)

En la siguiente tabla se puede apreciar el conjunto de características que forman el conjunto de datos, divididos en las dos tomas que se llevaron a cabo:

| Statistical features | | 16 hours | 15 hours |
|---|---|---|---|
| No._of_flows | | 987,627 | 976,882 |
| Src_bytes | | 4,860,168,866 | 5,940,523,728 |
| Des_bytes | | 44,743,560,943 | 44,303,195,509 |
| Src_Pkts | | 41,168,425 | 41,129,810 |
| Dst_pkts | | 53,402,915 | 52,585,462 |
| Protocol types | TCP | 771,488 | 720,665 |
| | UDP | 301,528 | 688,616 |
| | ICMP | 150 | 374 |
| | Others | 150 | 374 |
| Label | Normal | 1,064,987 | 1,153,774 |
| | Attack | 22,215 | 299,068 |
| Unique | Src_ip | 40 | 41 |
| | Dst_ip | 44 | 45 |

*Tabla 2.1*. Estadísticas del dataset UNSW-NB15.

*Nota*. Obtenido del dataset UNSW-NB16 (Moustafa y Slay, 2015)

**CIC-IDS2017**

Para no depender únicamente del conjunto de datos UNSW-NB15 también se ha realizado un estudio generando diferentes modelos de inteligencia artificial utilizando el conjunto de datos CIC-IDS2017.

CIC-IDS2017 es un *dataset* más reciente y completo en comparación con UNSW-NB15. Fue desarrollado por el Instituto Canadiense de Ciberseguridad (Canadian Institute of Cybersecurity [CIC]) y proporciona una muestra centrada en ataques modernos y más conocidos. La principal ventaja que ofrece este conjunto de datos frente al anterior es un amplio número de registros de ataques y la especificación de cada uno de dichos ataques. Los ataques se han recopilado a lo largo de 5 días, de lunes a viernes, quedando registrados en varios archivos CSV dependiendo el día de registro.

El conjunto de datos comprende 79 atributos superando significativamente los 49 de UNSW-NB15. Esto resulta en una mayor cantidad de información que pueden utilizar los modelos de inteligencia artificial para generar más reglas y/o patrones. (Sharafaldin et al., 2018).

### 2.2.3 CIBERATAQUES

En este trabajo se han estudiado las características o atributos de los datos para determinar cuáles de ellos varían en cada tipo de ciberataque, para así poder determinar qué forma una intrusión.

**Ciberataques de UNSW-NB15**

UNSW-NB15 tiene 9 tipos de ataques y se agrupan en categorías comunes, es decir, no se enfocan en tipos de ataques individuales, sino que están englobados en conjuntos. Esto genera más ambigüedad al no saber qué tipo de ataque específico se está analizando en concreto. En el siguiente listado se describe cada tipo de ataque:

| Tipo | Descripción | Funcionamiento |
|---|---|---|
| **Generic** | Intrusión mediante debilidades en cifrado. | Conexiones TCP cortas (ms), bajo volumen de paquetes. |
| **Exploits** | Explotación de vulnerabilidades (*Ransomware*, *malware*). | Flujos variables destinados a ganar privilegios. |
| **Fuzzers** | Envío de datos aleatorios para forzar errores. | Alto volumen de datos aleatorios y ráfagas de tráfico. |
| **DoS** | Saturación de recursos o ancho de banda. | Elevado número de conexiones y/o paquetes de gran tamaño. |
| **Reconnaissance** | Identificación de puertos y servicios activos. | Flujos de control breves, escaneo de puertos/servicios. |
| **Scan** | Escaneo de puertos y envío de contenido malicioso. | Conexiones TCP/UDP con mayor carga útil que *Reconnaissance*. |
| **Backdoors** | Acceso remoto persistente y oculto a la víctima. | Tráfico sigiloso, conexiones periódicas de control. |
| **Shellcode** | Inyección de código para ejecución de comandos. | Carga útil (*payload*) con instrucciones de terminal/*shell*. |
| **Worm** | Propagación por la red y contagio. | Uso de TCP (HTTP/SMTP) para infección. |

*Tabla 2.2*. Listado de ataques del conjunto de datos UNSW-NB15 con su descripción.

*Nota.* Adaptado de UNSW-NB15: A comprehensive data set for network intrusion detection systems, por N. Moustafa & J. Slay, 2015, Military Communications and Information Systems Conference (MilCIS). IEEE.

**Ciberataques de CIC-IDS2017:**

A diferencia de UNSW-NB15, este conjunto de datos se enfoca en ataques específicos en lugar de conjuntos de ataques más ambiguos.

| Categoría | Ataques Específicos | Comportamiento Técnico Sugerido |
|---|---|---|
| **Botnet** | Bot | Malware que envía información al atacante. |
| **DDoS / DoS** | Hulk, GoldenEye, Slowloris, Slowhttptest | Peticiones HTTP masivas, evasión de caché y mantenimiento de conexiones abiertas. |
| **Vulnerabilidades** | Heartbleed | Explotación de fallos en SSL/TLS. |
| **Infiltration** | Infiltration | Acceso no autorizado tras explotación inicial. |
| **Web Attacks** | Brute Force, inyección SQL, XSS | Inyección de código (SQL/JS) y búsqueda de credenciales en formularios web. |

*Tabla 2.3*. Listado de ataques del conjunto de datos UNSW-NB15 con su descripción.

Nota. Adaptado de CIC-IDS2017 Dataset, por Canadian Institute for Cybersecurity, s.f. (https://www.unb.ca/cic/datasets/ids-2017.html).

# CAPÍTULO 3. METODOLOGÍA

Tras entender todo el marco teórico que abarca este proyecto, en este capítulo se describen todas las herramientas de *software* utilizadas para su desarrollo. El *software* empleado es completamente gratuito, perteneciendo a la categoría de *software* libre.

Para cada herramienta se va a describir la función principal de cada componente de software, dependiendo de la función que le corresponda dentro del sistema de detección de intrusiones.

## 3.1 CAPTURA DE TRÁFICO RED Y PROCESAMIENTO DE DATOS

Para poder analizar las conexiones que efectúa un usuario es necesario una herramienta que pueda analizar el tráfico de red. Para ello, se han empleado programas capaces de transformar datos procedentes de paquetes de red y adaptarlos a las conexiones. Es importante diferenciar entre paquetes de red y conexiones debido a que los modelos de IA se han entrenado en base a conexiones.

**Zeek**

Zeek es una herramienta que analiza el tráfico, convirtiendo los archivos PCAP en formato binario a logs con formato con valores numéricos o categóricos. El programa guarda toda la información en archivos de tipo log dependiendo del protocolo utilizado (*conn.log, dns.log, http.log, file.log, ftp.log, ...*). En el marco de este proyecto, la función principal de zeek es extraer características a partir de paquete de red y transformarlas en conexiones.

Del archivo *conn.log* se obtienen 14 características esenciales para el *dataset*: *id.orig_h, id.orig_p, id.resp_h, id.resp_p, proto, service, duration, orig_bytes, resp_bytes, conn_state, orig_pkts, orig_ip_bytes, resp_pkts, resp_ip_bytes*. (Zeek Community, 2026)

**CICFlowmeter**

Es la herramienta encargada de analizar el tráfico para el dataset CIC-IDS2017, a diferencia de zeek no necesita leer de un archivo PCAP, sino que puede generar un

archivo CSV directamente de analizar el tráfico red de una interfaz determinada. También puede leer un archivo PCAP si es deseado.

El proceso restante es similar al de zeek, donde genera una serie de características en base a un *flow*. Los *flows* se forman por cinco campos que delimitan la conexión IP origen, IP destino, puerto origen, puerto destino y protocolo. Todos los *flows* que se vayan registrando los irá almacenando en un archivo CSV del cual el modelo de inteligencia artificial irá realizando predicciones. (Lashkari, 2026)

**Apache2**

Para poder realizar una simulación de ataques es necesario disponer de un servidor web para poder darle el papel de víctima y ser atacado por los ataques que forman los conjuntos de datos. En este caso, he utilizado Apache2 como servidor web hospedado en el puerto 80, para que se puedan ejecutar los ataques desde una máquina atacante en Kali Linux. (Apache Software Foundation, 2026)

## 3.2 COMUNICACIÓN CLIENTE-SERVIDOR MEDIANTE API RESTFUL

Para poder establecer un control entre la funcionalidad que ofrece el sistema de detección de intrusiones, he optado por el uso de una API, pudiendo así desplegar la interfaz de usuario mediante el navegador y ceñirme a una aplicación web convencional.

Una API (*Applicaction Progamming Inteface*) es una interfaz de aplicaciones que se encarga de garantizar la comunicación entre diferentes partes de una aplicación. En el contexto de aplicaciones web, permite la interacción estructurada entre el cliente (*frontend*) y el servidor (*backend*) mediante peticiones y respuestas HTTP principalmente, devolviendo datos en formato JSON. La API también coordina todo el uso de los componentes desarrollados en Python.

Esto garantiza una funcionalidad más sencilla de utilizar que utilizar la línea de comandos convencional, permitiendo una funcionalidad mejor estructurada, seguido de una mayor facilidad a la hora de visualizar datos mediante la interfaz de usuario.

**SQLITE3**

He utilizado SQLite3 como motor de base de datos para la persistencia de estos frente a otros motores de bases debido a su sencillez y bajo tamaño. Hay una tabla de conexiones y otra de amenazas (THREATS) donde se va guardando las predicciones realizadas en la captura de tráfico en tiempo real. (Hipp, 2026)

**Fast API**

Fast API es un framework enfocado en tecnologías web, escrito en Python, cuyo enfoque se basa en una alta reactividad y bajos tiempos de respuesta, de ahí el nombre fast en inglés.

En este trabajo se ha optado por FastAPI, debido a su facilidad de uso y rapidez a la hora de enviar y recibir datos mediante peticiones http y https. (Ramírez, 2026)


*Figura 3.1.* Logotipo del *framework* FastAPI.

*Nota***.** De *FastAPI logo.svg* [Logotipo], por Tiangolo, 2018, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:FastAPI_logo.svg). Licencia MIT.

**Streamlit**

Es un *framework* escrito en Python que forma la interfaz de usuario de este proyecto. Se comunica directamente con FastAPI, siendo esta la parte que ve el usuario y la encargada de mandar peticiones HTTP/S a FastAPI, coordinando, con ello, todas las funciones que forman la aplicación.

Streamlit proporciona una interfaz de usuario orientada a aplicaciones de análisis de datos y de inteligencia artificial, garantizando mayor funcionalidad para visualizar archivos CSV respecto a otras alternativas convencionales como HTML, CSS y JavaScript. (Streamlit Inc., 2026)


*Figura 3.2.* Logotipo del *framework* Streamlit.

*Nota*. De *Streamlit logo primary colormark darktext* [Logotipo], por Streamlit, 2022, Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Streamlit-logo-primary-colormark-darktext.png). Licencia CC BY-SA 4.0.

## 3.3 DESARROLLO EN PYTHON

El trabajo se ha escrito íntegramente en Python a excepción del estilo para la interfaz de usuario que ha sido escrito en CSS. El motivo principal de la elección de Python como lenguaje principal del proyecto es su alta efectividad procesando datos en formato CSV, librerías de ciencia de datos, herramientas para visualizar datos y librerías para ciberseguridad.

Python tiene un funcionamiento diferente a la hora de añadir librerías frente a lenguajes convencionales como C. Esto se debe a que utiliza módulos administrados por un instalador de paquetes en vez del funcionamiento clásico de librerías implementadas. Los paquetes se instalan mediante PIP (*Python Package Instaler*) mediante la línea de comandos.

### Pandas

Pandas es una de las librerías más conocidas de Python cuya finalidad consiste en proporcionar funcionalidad para manejar datos en formato CSV, mediante su formato lógico llamado *DataFrame* (Contenedor de datos).

Su funcionalidad es muy amplia y permite realizar muchas operaciones con los datos de los que se disponga, como leer datos de archivos CSV, modificar filas, modificar columnas, generar nuevos archivos CSV, eliminar datos nulos o duplicados, etc. Pandas

se utiliza en varias fases del desarrollo de este proyecto y de la ciencia de datos e inteligencia artificial.

**Scikit-learn**

Scikit learn es una de las librerías o módulos más conocidos y utilizadas en el ámbito de la inteligencia artificial, contiene cualquier tipo de elemento necesario para entrenar modelos de inteligencia artificial. Dispone de las funciones necesarias enfocadas al aprendizaje automático que se ha aplicado en este trabajo. (Pedregosa et al., 2011)

**Matplotlib y Seaborn**

Estas librerías pertenecen a la rama de la ciencia de datos, más que al aprendizaje automático o a la inteligencia artificial. He agrupado a Matplotlib y Seaborn conjuntamente debido a que comparten la misma funcionalidad, que es proporcionar herramientas como gráficos y tablas para poder visualizar datos. Los gráficos y tablas que se han utilizado en este trabajo han sido generados mediante código de estos módulos. (Caswell et al., 2026; Waskom, 2026)



## 3.4 MÉTRICAS DE RENDIMIENTO Y EVALUACIÓN PARA MODELOS DE INTELIGENCIA ARTIFICIAL

**Matriz de confusión**

Para poder determinar si el modelo entrenado presenta un rendimiento adecuado se recurre a la matriz de confusión. Esta es la herramienta con la que se visualizan los resultados obtenidos tras ejecutar algoritmos de clasificación sobre un conjunto de pruebas y determinar, con ello, las predicciones que se han realizado correcta o incorrectamente tras compararlo con sus etiquetas.

La matriz de confusión permite determinar el número de verdaderos positivos (VP), verdaderos negativos (VN), falsos positivos (FP) y falsos negativos (FN). Estos valores permiten evaluar el rendimiento del modelo mediante métricas como la exactitud (*Accuracy*), la sensibilidad (*recall*), puntuación F1 (*f1-score*) y la precisión (*precision*). (Géron, 2019)

Ejemplo de una matriz de confusión:

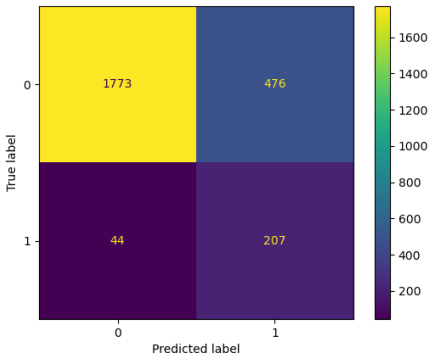



*Figura 3.3*. Ejemplo de una Matriz de confusión.

**Medidas de rendimiento en reporte de clasificación**

Una vez se han obtenido los resultados de la matriz de confusión se pueden generar más métricas para comprobar el rendimiento que ha tenido el modelo en más profundidad.

**Exactitud (*Accuracy*)**

Proporción total de predicciones correctas sobre todas las predicciones que se han realizado. Mide el rendimiento general del modelo.

Se calcula dividiendo los verdaderos positivos y negativos entre todos los demás datos:

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

**Precisión (*Precision*)**

Indica cuántas de las predicciones positivas hechas por el modelo fueron correctas.

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

**Sensibilidad (*Recall*)**

Mide la capacidad del modelo para encontrar todos los casos positivos reales. Es útil cuando el número de falsos negativos afecta considerablemente a cómo rinde el modelo.

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

**Puntuación F1 (*F1-score*)**

Es la media armónica entre la precisión y la sensibilidad. Proporciona un balance entre ambas métricas, especialmente útil en conjuntos de datos desbalanceados.

$$F1 - Score = 2 \times \frac{precisión \times sensibilidad}{precisión + sensibilidad}$$

## 3.5. PROCEDIMIENTO EXPERIMENTAL PARA LA DETECCIÓN DE INTRUSIONES EN TIEMPO REAL

Para determinar si todo el sistema está integrado correctamente se realizarán *test* sobre diferentes ataques con el sistema de detección de intrusiones capturando tráfico en tiempo real. Para ello será necesario disponer de una máquina virtual que simule el papel de atacante. En este trabajo se ha escogido una máquina virtual con el sistema operativo Kali Linux para desempeñar esa función. Se ha escogido ese sistema operativo en concreto debido al gran número de aplicaciones preinstaladas relacionadas con ciberseguridad y también debido a que son las mismas que se utilizaron al simular los ataque de CIC-IDS2017.

# 4. DESARROLLO DEL PROYECTO

Este capítulo contiene todos los procesos y funcionalidades aplicadas del capítulo anterior. Para ello se muestran diversos aspectos técnicos con diagramas y flujos de trabajo para cada caso práctico que componen el trabajo.

## 4.1 CAPTURA DE TRÁFICO Y ANÁLISIS DEL TRÁFICO RED

El sistema de detección de intrusiones es capaz de generar dos funciones principales:

- Analizar archivos PCAP y clasificar cada conexión, mostrar sus parámetros con el tráfico clasificado.
- Analizar el tráfico de una interfaz de red en tiempo real y mostrar el resultado de esas clasificaciones generando alertas.



**Análisis de un archivo PCAP**

Para este método se utiliza la herramienta Zeek para transformar los datos en formato binario de paquetes de red a archivos tipo LOG. Posteriormente se transforman a formato CSV para que el modelo de IA lo pueda entender, y finalmente se pueden visualizar en formato CSV. Para realizar predicciones en este caso, se utiliza el modelo UNSW-NB15 frente al de CIC-IDS2017 debido a que CICFlowmeter procesa mejor el tráfico en tiempo real.

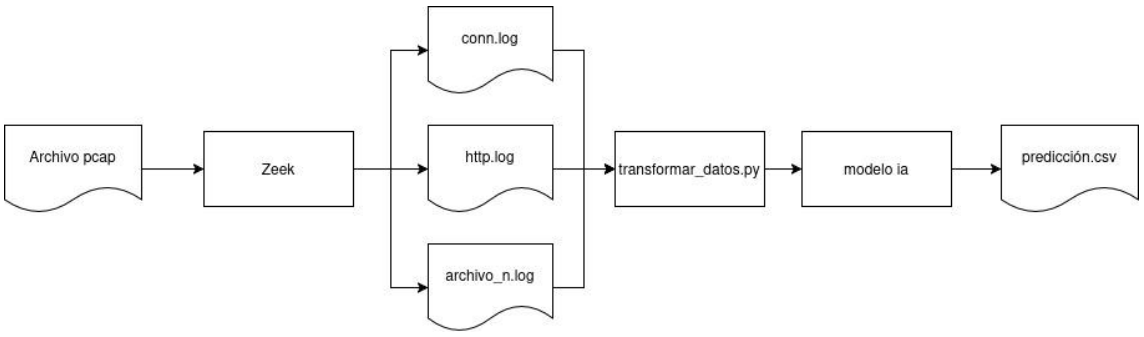



*Figura 4.1*. Diagrama de flujo para analizar un archivo PCAP.

### Captura y análisis de tráfico de red en tiempo real

Para capturar datos en tiempo real se utiliza la herramienta CICFlowmeter sobre una interfaz de red seleccionada por el usuario. El proceso de transformar datos de paquetes de red a CSV lo lleva a cabo CICFlowmeter, de tal forma que únicamente se ha de predecir ese archivo CSV obtenido. Para mantener este proceso de una manera constante, cada 10 segundos se publica un archivo CSV que podrá visualizarlo el usuario mediante la interfaz de usuario. Finalmente, se almacenan los datos en la base de datos de SQLite y si hay algún ataque clasificado como malicioso se muestra por pantalla al usuario.

En la siguiente figura se puede ver el flujo de datos que sigue el programa desde escuchar una interfaz de datos hasta visualizar las predicciones por el usuario.

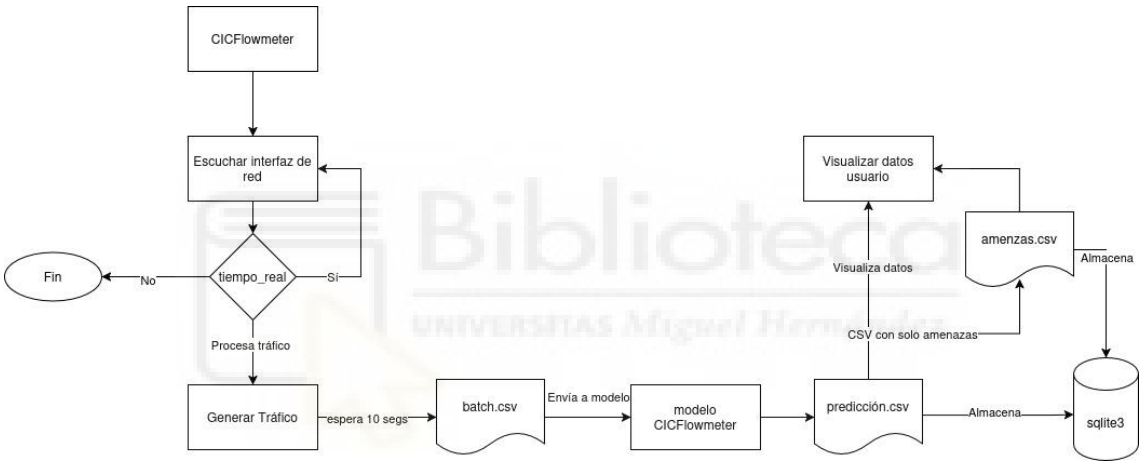



*Figura 4.2*. Diagrama de flujo para realizar captura continua.

## 4.2 ENTRENAMIENTO Y OPTIMIZACIÓN DE MODELOS DE INTELIGENCIA ARTIFICIAL

Este es el proceso que se ha llevado a cabo para entrenar los modelos de inteligencia artificial, pudiendo crear, con ello, un modelo clasificador. En el siguiente diagrama se muestra el proceso iterativo mencionado:

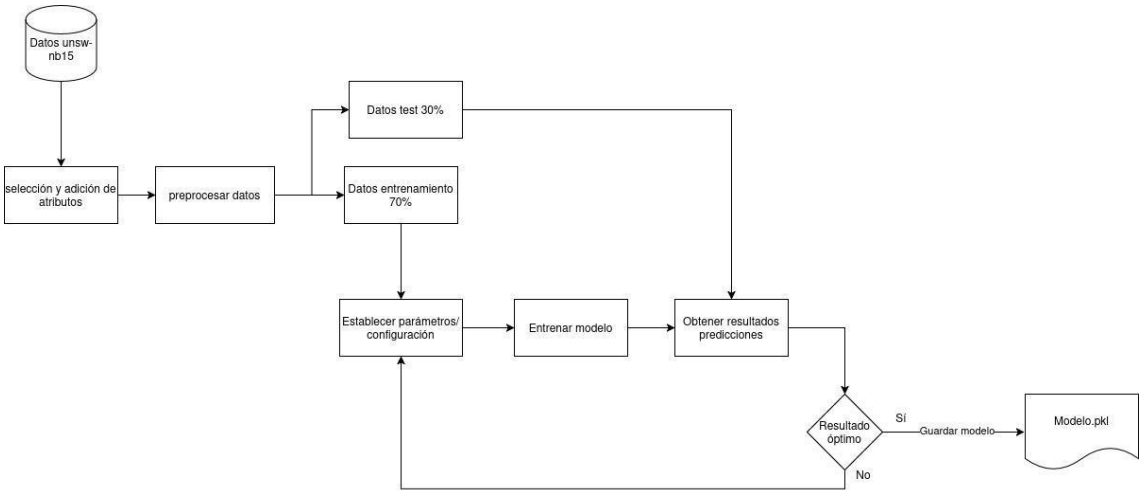



*Figura 4.3*. Diagrama de flujo del proceso de creación de modelos de inteligencia artificial.

Para poder generar modelos de inteligencia artificial correctamente, se ha seguido el siguiente proceso:

- Preprocesar datos: se cambian los valores de categórico a numérico, se eliminan todos los valores nulos, se eliminan todos los valores duplicados y, finalmente, se asegura de que no haya ningún dato que pudiera impedir el correcto entrenamiento del modelo.

- Selección y cálculo de atributos: se determina una serie de atributos para entrenar el modelo, no todo el conjunto. También se calculan algunos atributos adicionales para mejorar el rendimiento de dicho modelo.

- Se divide el conjunto de datos inicial en datos en entrenamiento (70%) y test (30%), cada subconjunto se emplea en su fase determinada.

- Se establecen los valores de los parámetros que el algoritmo usará para entrenar el modelo.

- Se realizan predicciones sobre el conjunto de datos de prueba y se obtiene el reporte de clasificación para determinar si el modelo presenta un rendimiento adecuado, en caso contrario, se vuelve a configurar y entrenar.

- Se comprobarán los resultados de predicciones realizadas tanto en el dataset de entrenamiento como el de pruebas para determinar si se produce sobreajuste y

comprobar, de este modo, que el modelo no ha “memorizado” los datos de entrenamiento.

- Una vez terminado el proceso, se serializa el modelo en un archivo con formato PKL para su uso futuro.

## 4.3 ARQUITECTURA DEL SISTEMA E INTEGRACIÓN DE LA API

La aplicación se divide en dos partes principales:

- *Frontend*: interfaz de usuario, desarrollado en Streamlit.
- *Backend*: lógica de la aplicación, coordinada por FastAPI.

Mientras que Streamlit permite desplegar todos los componentes que forman el sistema de detección de intrusiones por navegador, FastAPI realiza las llamadas correspondientes entre base de datos y módulo de Python programados.

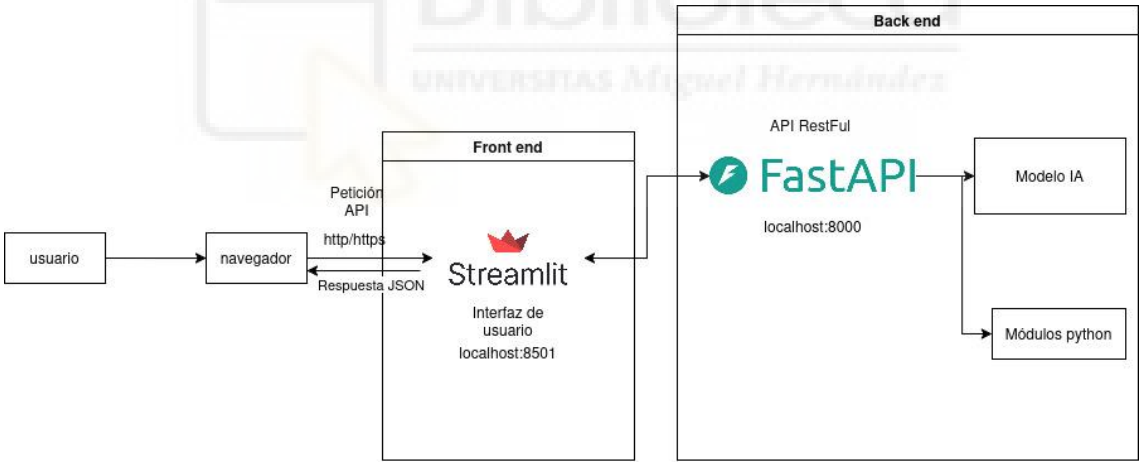



*Figura 4.4*. Diagrama de flujo para analizar un archivo PCAP.

Cada uno de estos *endpoints*, representa una llamada http que se realiza desde el navegador para comunicarse con la API y de esa forma poder coordinar el código escrito en Python y ponerlo todo en marcha.

La API cuenta con un número de *endpoints* donde se realizar peticiones http y se busca obtener datos con peticiones GET y/o enviar datos con peticiones POST:

| Método | Endpoint | Descripción |
|---|---|---|
| **GET** | / | Información general de la API y lista de endpoints disponibles |
| **GET** | /health | Estado del sistema |
| **POST** | /start_escaner | Inicia la captura de tráfico en tiempo real sobre la interfaz activa |
| **POST** | /stop_escaner | Detiene la captura y cierra la sesión registrando métricas finales |
| **POST** | /analizar_pcap | Recibe un fichero PCAP y devuelve las predicciones del modelo |
| **POST** | /predict | Recibe flujos de red en JSON y devuelve predicciones (integración externa) |
| **GET** | /api/v1/last | Últimas predicciones generadas en el CSV de tiempo real |
| **GET** | /api/v1/threats | Amenazas detectadas en el último batch procesado |
| **GET** | /api/v1/logs | Últimas N entradas del log del sistema (?last=50) |
| **GET** | /api/v1/interfaces | Lista las interfaces de red disponibles en el sistema |
| **POST** | /api/v1/interfaces/{nombre} | Cambia la interfaz de red activa del escáner |
| **GET** | /api/v1/sesiones | Lista todas las sesiones de escaneo con sus métricas |
| **GET** | /api/v1/sesiones/{id} | Detalle de una sesión: distribución de ataques, IPs, tiempos |
| **GET** | /api/v1/sesiones/{id}/trafico_por_ip | Flujos agrupados por IP origen/destino y tipo de predicción |
| **GET** | /api/v1/export | Exporta los datos de amenazas en formato CSV o JSON (?format=csv&sesion_id=X) |

*Tabla 4.1*. Endpoints de la API.

*Nota*. Elaboración propia, obtenida a partir de los *endpoints* utilizados en la API.

## 4.4 DISEÑO E IMPLEMENTACIÓN DE LA INTERFAZ DE USUARIO

La interfaz de usuario se divide en dos páginas dependiendo de la función deseada, la funcionalidad ofrecida es la siguiente:

- Captura de tráfico de red en tiempo real.
    - Registrar todos los datos respecto a una sesión. Una sesión se establece al iniciar la captura en tiempo real y finaliza al parar dicha captura.

- Análisis de un archivo PCAP y CSV.

**Página monitorización en tiempo real**

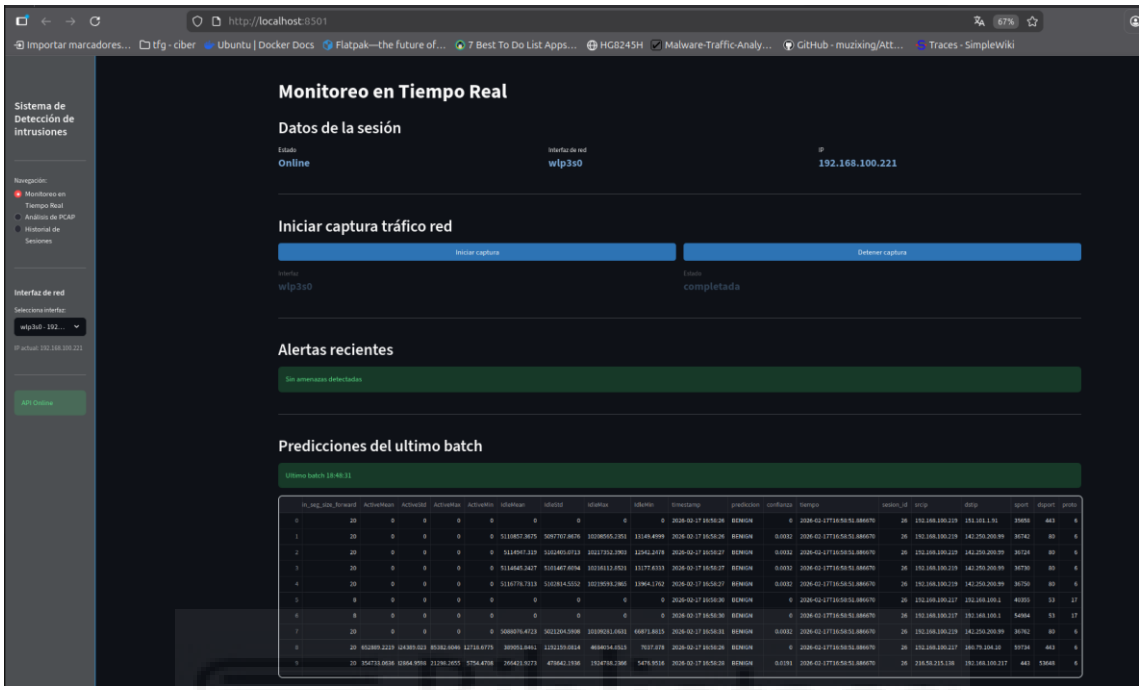



*Figura 4.5.* Interfaz de usuario frontal de la aplicación.

La parte de control navegacional situada a la izquierda, siempre presente, permite cambiar de página y de interfaz de usuario.

Se puede visualizar el estado de la sesión "*online*" u "*offline*", la interfaz de red seleccionada y la IP asociada a esa interfaz.

La interfaz de red se puede seleccionar desde la barra lateral izquierda. Se permite seleccionar de las interfaces disponibles, también se puede visualizar la dirección IP asociada a dicha interfaz.

La pantalla principal "Monitoreo de tráfico red" permite las siguientes funciones:

- El botón "iniciar captura" pone en marcha, en bucle, el proceso de captura de tráfico y, cada vez que se realiza una predicción, aparece en pantalla.
- El botón "Detener captura" finaliza el bucle iniciado de captura de tráfico red, terminando todos los procesos que intervinieran en la captura de tráfico red.

El resultado de la predicción es un archivo CSV que se puede encontrar bajo el título "Predicciones del último *batch*", este hace referencia a un archivo CSV que se irá actualizando con el tráfico de red periódicamente. Dependiendo del tráfico cursado, el tiempo será de 15 segundos, si no ha habido suficiente tráfico detectado por la herramienta de captura de tráfico no se mostrará el archivo.

Streamlit permite, al estar diseñado con una orientación a temáticas basadas en ciencias de datos, visualizar completamente un archivo CSV, independientemente del número de filas que tenga. Esto posibilita que se puedan desglosar todas las conexiones que haya realizado el usuario.

**LOG de tiempo real**

He utilizado un LOG más abajo de la parte donde se visualiza cada conexión para que el usuario tenga conocimiento sobre todo lo suceda internamente en la aplicación.

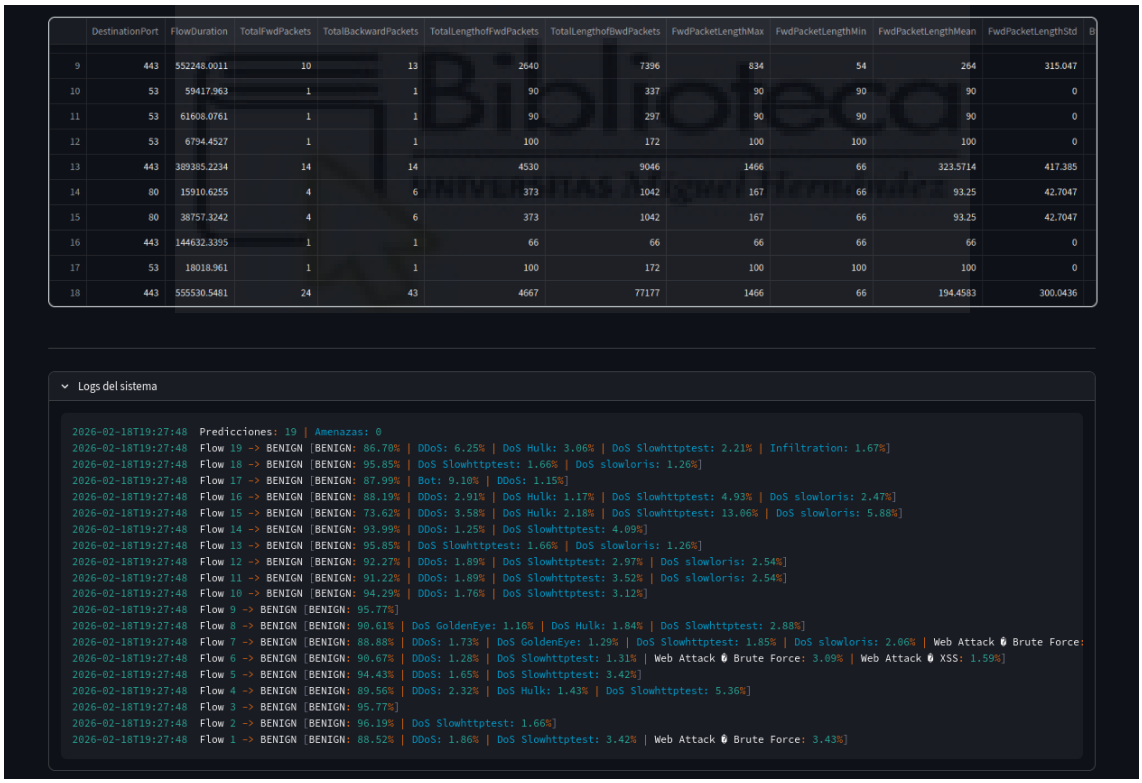



*Figura 4.6*. Interfaz de usuario mostrando más información.

He utilizado esta captura para visualizar un error que tenía al insertar filas en la base de datos. De esta forma se puede tener un conocimiento sobre el funcionamiento interno de la aplicación.

### Página de visualización de sesiones

En esta página se permite ver todas las sesiones que ha registrado el usuario, pudiendo visualizar todos los datos de conexiones red registrados. También he introducido un apartado de estadísticas que describen todas las clasificaciones de dicha sesión, así se puede consultar qué amenazas han transcurrido en un periodo de tiempo predeterminado. El usuario tiene la opción de exportar esos datos a formato CSV. Cada sesión se almacena en la base de datos del sistema de detección de intrusiones.

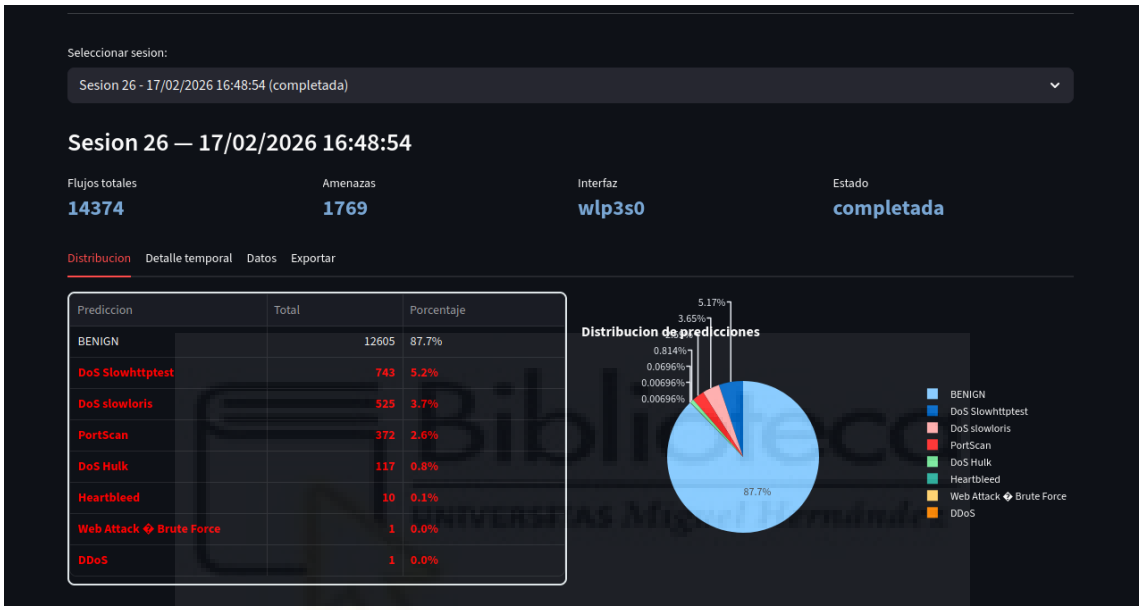



*Figura 4.7.* Gráfico de la sesión 26 que realicé para probar varios ataques desde una máquina virtual de Kali Linux.

### Página para el análisis de archivos PCAP

Esta página permite subir un archivo PCAP o CSV para poder visualizar todas las conexiones que lo forman.

El archivo PCAP viene procesado con su predicción asociada. Esta predicción se realiza con el modelo de inteligencia artificial entrenado en UNSW-NB15. En la siguiente figura se muestra la interfaz de usuario asociada a esa parte:

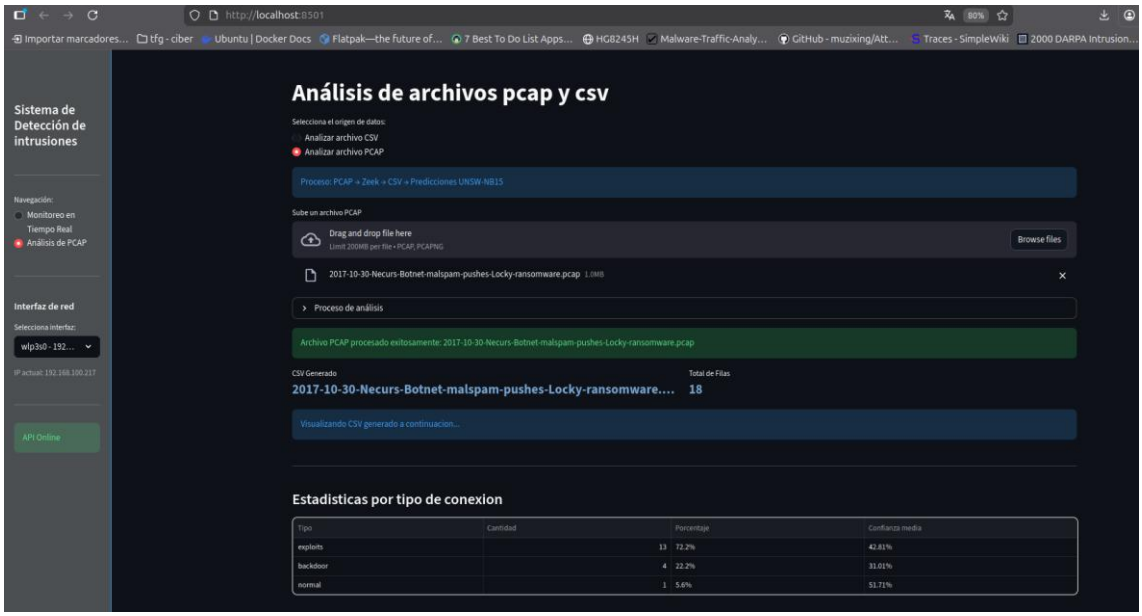
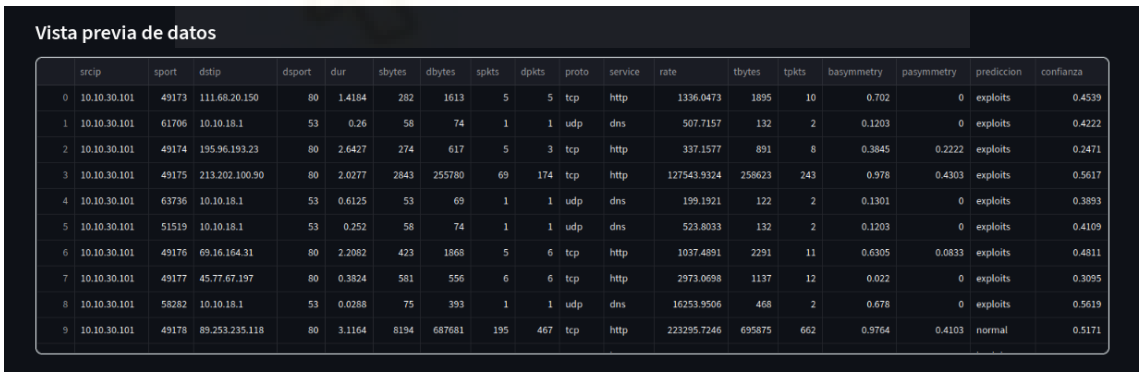



*Figura 4.8*. Interfaz de usuario del apartado analizar PCAPs.

En la tabla Estadísticas por tipo de conexión aparecen todas las predicciones del archivo PCAP. Más abajo se pueden visualizar los datos de conexiones en una tabla. En la siguiente figura aparece la tabla que he mencionado ya que no me ha sido posible capturarlo todo. En la tabla están todas las filas, únicamente aparecen 9 por pantalla porque no he podido capturarlas todas.




| | srcip | sport | dstip | dsport | dur | sbytes | dbytes | spkts | dpkts | proto | service | rate | tbytes | tpkts | basymmetry | pasymmetry | prediccion | confianza |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 10.10.30.101 | 49173 | 111.68.20.150 | 80 | 1.4184 | 282 | 1613 | 5 | 5 | tcp | http | 1336.0473 | 1895 | 10 | 0.702 | 0 | exploits | 0.4539 |
| 1 | 10.10.30.101 | 61706 | 10.10.18.1 | 53 | 0.26 | 58 | 74 | 1 | 1 | udp | dns | 507.7157 | 132 | 2 | 0.1203 | 0 | exploits | 0.4222 |
| 2 | 10.10.30.101 | 49174 | 195.96.193.23 | 80 | 2.6427 | 274 | 617 | 5 | 3 | tcp | http | 337.1577 | 891 | 8 | 0.3845 | 0.2222 | exploits | 0.2471 |
| 3 | 10.10.30.101 | 49175 | 213.202.100.90 | 80 | 2.0277 | 2843 | 255780 | 69 | 174 | tcp | http | 127543.9324 | 258623 | 243 | 0.978 | 0.4303 | exploits | 0.5617 |
| 4 | 10.10.30.101 | 63736 | 10.10.18.1 | 53 | 0.6125 | 53 | 69 | 1 | 1 | udp | dns | 199.1921 | 122 | 2 | 0.1301 | 0 | exploits | 0.3893 |
| 5 | 10.10.30.101 | 51519 | 10.10.18.1 | 53 | 0.252 | 58 | 74 | 1 | 1 | udp | dns | 523.8033 | 132 | 2 | 0.1203 | 0 | exploits | 0.4109 |
| 6 | 10.10.30.101 | 49176 | 69.16.164.31 | 80 | 2.2082 | 423 | 1868 | 5 | 6 | tcp | http | 1037.4891 | 2291 | 11 | 0.6305 | 0.0833 | exploits | 0.4811 |
| 7 | 10.10.30.101 | 49177 | 45.77.67.197 | 80 | 0.3824 | 581 | 556 | 6 | 6 | tcp | http | 2973.0698 | 1137 | 12 | 0.022 | 0 | exploits | 0.3095 |
| 8 | 10.10.30.101 | 58282 | 10.10.18.1 | 53 | 0.0288 | 75 | 393 | 1 | 1 | udp | dns | 16253.9506 | 468 | 2 | 0.678 | 0 | exploits | 0.5619 |
| 9 | 10.10.30.101 | 49178 | 89.253.235.118 | 80 | 3.1164 | 8194 | 687681 | 195 | 467 | tcp | http | 223295.7246 | 695875 | 662 | 0.9764 | 0.4103 | normal | 0.5171 |

## 4.5 FUNCIONAMIENTO DE TEST DE ATAQUES CON KALI LINUX

Para poder probar el correcto funcionamiento del sistema de detección de intrusiones se han realizado varios ataques sobre un servidor web Apache2 con el programa de ejecutándose de fondo. En la siguiente figura se muestra un diagrama con todo el proceso que sucede desde la captura de ese ataque hasta que el usuario puede visualizar la información en el *frontend*.

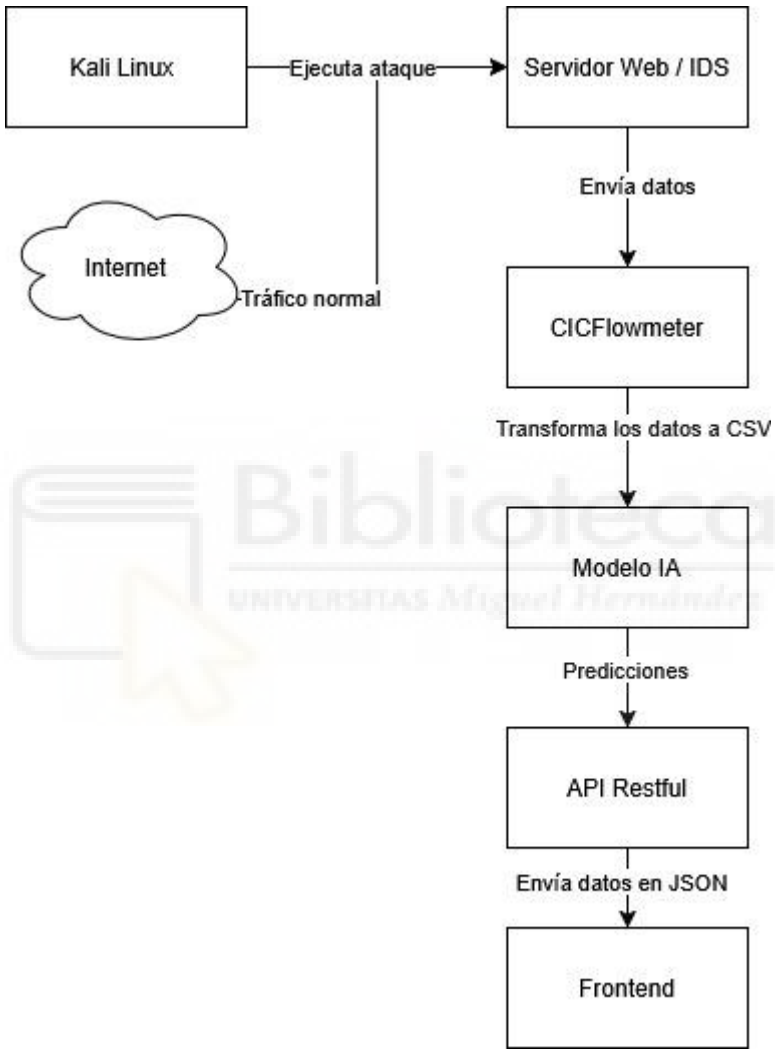



*Figura 4.10*. Diagrama que muestra el funcionamiento de los test de ataques sobre un servidor web Apache2 y el IDS ejecutándose de fondo.

# 5. RESULTADOS DE CLASIFICACIÓN DE LOS MODELOS DE INTELIGENCIA ARTIFICIAL

En este tema se abordan las diferentes pruebas realizadas para poder determinar qué modelo de inteligencia artificial se utilizará en el despliegue del programa, el objetivo es buscar los modelos que presenten el mejor rendimiento en el contexto de intentar detectar anomalías en el tráfico red.

Los modelos utilizan las características comunes que están presentes en las conexiones monitorizadas, a modo de simular los datos que se obtendrán en una situación de monitorización real con el uso del IDS. Se busca la mayor similitud a una situación real para que no se produzca el fenómeno *datashifting.* Esto sucede cuando los datos con los que se entrena el modelo varían mucho de los datos en un contexto real.



## 5.1 SELECCIÓN DE ATRIBUTOS

Para el entrenamiento de cada modelo se ha escogido una serie de atributos dependiendo de la herramienta de análisis utilizada. Para los modelos de UNSW-NB15 se ha utilizado Zeek y para CIC-IDS2017 se ha empleado CICFlowmeter.

### 5.1.1 SELECCIÓN DE ATRIBUTOS DE UNSW-NB15

Para este conjunto de datos se han escogido únicamente las características de las conexiones que se hayan podido extraer directamente sin ningún cálculo adicional. Posteriormente, se ha añadido un cálculo extra para ofrecer más información al modelo, y así aumentar su capacidad de predicción.

**Características base:**

- Duración conexión (*dur*).
- Número de bytes origen y destino (*sbytes*, *dbytes*).

- Número de paquetes en origen y destino (*spkts*, *dpkts*).
- Tasa de bits (*rate*).
- Protocolo empleado y servicio (proto, *service*).
- Estado de la conexión (*status*).

**Características adicionales calculadas:**

Se calculan a partir de las características base para añadir mayor detalle a los datos y que el modelo pueda aumentar el número de predicciones realizadas correctamente. Este es el número extra de característica añadidas:

- Número total de paquetes y número total de bytes (*tot_bytes*, *tot_pkts*): suma total del número de paquetes y bytes.
    - *tbytes*: *sbytes*+*dbytes*
    - *tpkts*: *spkts*+*dpkts*
- Ratio de paquetes y ratio de bytes (*byte_ratio*, *pkt_ratio*): proporción entre bytes de origen y destino y proporción entre paquetes de origen y destino.
    - *byte_ratio*: *sbytes*/(*dbytes*+1)
    - *pkt_ratio*: *spkts*/(*dpkts*+1)

Se suma +1 para evitar divisiones entre 0 y que cause error.

### 5.1.2 SELECCIÓN DE ATRIBUTOS DE CIC-IDS2017

Para poder determinar el tipo de atributos que se podría emplear a la hora de entrenar los modelos de inteligencia artificial, utilizando CIC-IDS2017, primero se ha tenido que realizar una monitorización de prueba sobre los datos obtenidos mediante CICFlowmeter, determinando, con ello, que atributos eran válidos para entrenar el modelo, es decir, que atributos no eran 0 o valores nulos “-”.

Una vez determinado el conjunto de atributos válido para entrenar, se ha empleado un algoritmo para seleccionar un número de atributos determinados, *kbest*, *pca* o *xgboost* para extraer los atributos con mayor peso y menor redundancia. El resultado obtenido es un listado ordenado con los atributos con mayor peso (*weight*).

## 5.2 MUESTRA Y BALANCEO DE DATOS

Para obtener los datos de los *datasets* se han tenido que concatenar los datos de todos los archivos CSV, almacenando estos datos de forma lógica, transformándolos en un DataFrame. Se ha utilizado SMOTE y RandomUndensampler para balancear la muestra de datos perteneciente al conjunto de pruebas para asegurarnos de poder ver un rendimiento correcto del modelo sin que hay un sesgo hacia clases mayoritarias. (Chawla et al, 2002; Lemaître et al., 2017)

Una vez obtenidos los datos totales se aplica la función *train_test_split*, encargada de separar los datos en entrenamiento y test, quedando el siguiente número de datos para cada acción:

| | **Datos UNSW_NB** | **Datos CIC-IDS2017** |
|---|---|---|
| **Muestra entrenamiento** | 1,778,030 | 1,718,500 |
| **Muestra test** | 762,013 | 736,500 |
| **Total** | 2,540,044 | 2,455,000 |

*Tabla 5.1*. Muestra de datos partida para entrenamiento y test.

*Nota*. Obtenida a partir de la muestra del *dataset* original.

**Balanceo de datos UNSW-5**

| **Clase** | Antes (Registros) | Antes (%) | Después (Registros) | Después (%) |
|---|---|---|---|---|
| normal | 1,553,135 | 87.4% | 40,000 | 16.8% |
| generic | 150,837 | 8.5% | 40,000 | 16.8% |
| exploits | 31,167 | 1.8% | 40,000 | 16.8% |
| fuzzers | 16,972 | 1.0% | 40,000 | 16.8% |
| dos | 11,447 | 0.6% | 34,341 | 14.5% |
| reconnaissance | 9,791 | 0.6% | 29,373 | 12.4% |
| analysis | 1,874 | 0.1% | 5,622 | 2.4% |
| backdoor | 1,630 | 0.1% | 4,890 | 2.1% |
| shellcode | 1,058 | 0.1% | 3,174 | 1.3% |
| Total | 1,777,911 | 100% | 237,400 | 100% |

*Tabla 5.2*. Muestra de datos de entrenamiento del conjunto de datos UNSW-NB15.

*Nota*. Obtención propia tras aplicar el balanceo de datos.

**Balanceo de datos CIC-IDS2017**

| **Ataque** | Antes (Registros) | Antes (%) | Después (Registros) | Después (%) |
|---|---|---|---|---|
| **BENIGN** | 1,591,168 | 80.3% | 40,000 | 16.5% |
| Dos Hulk | 161,751 | 8.2% | 40,000 | 16.5% |
| PortScan | 111,251 | 5.6% | 40,000 | 16.5% |
| DDoS | 89,619 | 4.5% | 40,000 | 16.5% |
| Dos GoldenEye | 7,205 | 0.4% | 21,615 | 8.9% |
| FTP-Patator | 5,557 | 0.3% | 16,671 | 6.9% |
| SSH-Patator | 4,128 | 0.2% | 12,384 | 5.1% |
| Dos slowloris | 4,057 | 0.2% | 12,171 | 5.0% |
| DoS slowhttptest | 3,849 | 0.2% | 11,547 | 4.8% |
| Bot | 1,376 | 0.1% | 4,128 | 1.7% |
| Web Attack Brute Force | 1,055 | 0.1% | 3,165 | 1.3% |
| Web Attack XSS | 456 | 0.01% | 1,368 | 0.6 % |

*Tabla 5.3*. Muestra de datos de entrenamiento del conjunto de datos CIC-IDS2017

*Nota*. Obtención propia tras aplicar el balanceo de datos.

## 5.3 ENTRENAMIENTO Y RESULTADOS DE CLASIFICACIÓN

En esta fase se muestran los resultados que se han obtenido respecto a las métricas de rendimiento mencionadas, con el fin de poder seleccionar el mejor modelo para el sistema de detección de intrusiones. A continuación, se presentan los resultados obtenidos para cada modelo de inteligencia artificial.

**Resultado tiempos de entrenamiento y test UNSW-NB15**

Se han calculado los tiempos de entrenamiento y test para tener una idea de qué modelos tardan más al entrenarse y si hay alguno que al realizar predicciones tenga un tiempo muy alto para descartarlo.

En la siguiente figura se aprecia que KNN representa un tiempo nulo en la fase de entrenamiento mientras que tarda considerablemente más en la fase de pruebas. El segundo algoritmo que más tiempo tarda es Random Forest, seguido de LinearSVC.

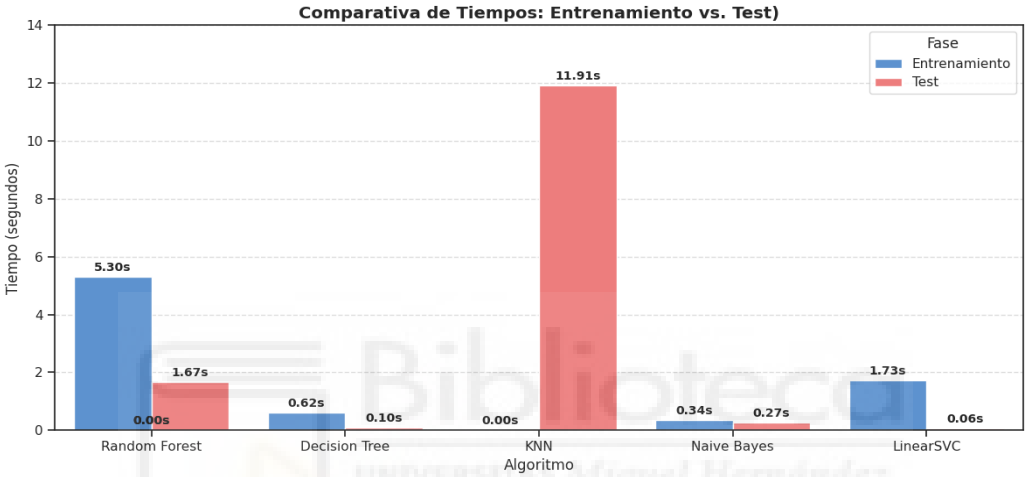



*Figura 5.1*. Gráfica con tiempos de entrenamiento y test de los modelos UNSW-NB15.

### Resultado tiempos de entrenamiento y test CIC-IDS2017

Ahora es el turno de los modelos de CIC-IDS2017, donde al tener un número de registros un poco más alto, el tiempo de entrenamiento sube ligeramente, sobre todo en el modelo de K-Nearest Neighbors donde la fase de test llega a durar hasta 87.33 segs.

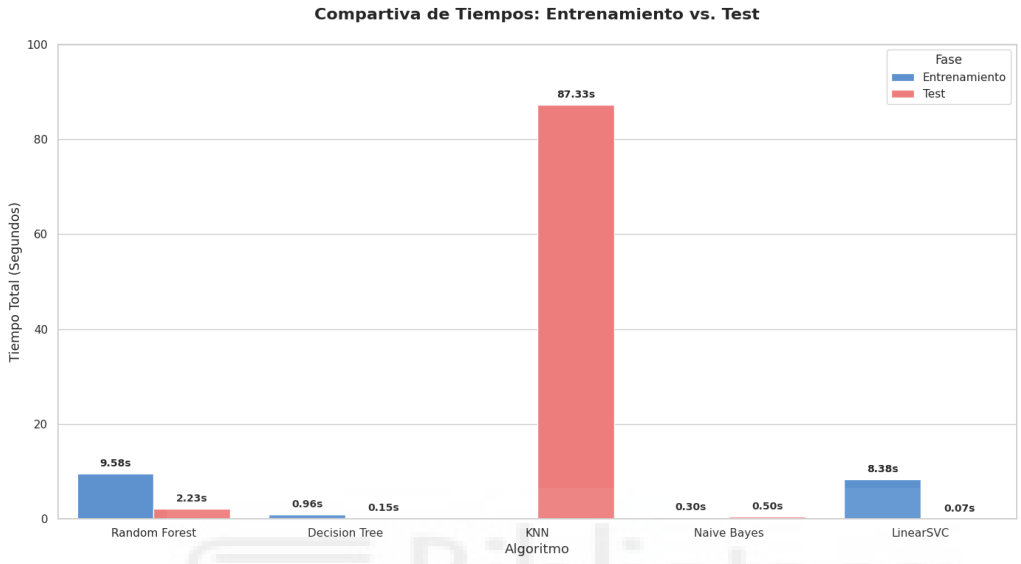



Figura 5.2. Gráfica con tiempos de entrenamiento y test de los modelos CIC-IDS2017

**Resultado general de todos los modelos:**

Se muestra el resultado tras evaluar todos los modelos listados para ambos conjuntos de datos:

| | CIC-IDS2017 | | | | UNSW-NB15 | | | |
|---|---|---|---|---|---|---|---|---|
| **Modelo** | **Accuracy** | **Precision** | **Recall** | **F1-score** | **Accuracy** | **Precision** | **Recall** | **F1-score** |
| Random Forest | 0.9929 | 0.9967 | 0.9929 | 0.9945 | 0.9717 | 0.9836 | 0.9710 | 0.9759 |
| Decision Tree | 0.9264 | 0.9787 | 0.9264 | 0.9489 | 0.9691 | 0.9823 | 0.9691 | 0.9743 |
| KNN | 0.9837 | 0.9837 | 0.9837 | 0.9835 | 0.9689 | 0.9833 | 0.9689 | 0.9746 |
| Naive Bayes | 0.5891 | 0.954 | 0.5891 | 0.7172 | 0.9376 | 0.9677 | 0.9376 | 0.9517 |
| Linear SVM | 0.865 | 0.8931 | 0.865 | 0.872 | 0.9634 | 0.9784 | 0.9634 | 0.9691 |

Tabla 5.4. Resultados globales obtenidos para cada modelo entrenado por ambos conjuntos de datos.

*Nota*. Obtención propia, tras obtener el resultado de clasificación para todos los modelos.

En el siguiente gráfico vertical de barras se visualiza mejor la diferencia entre cada modelo:

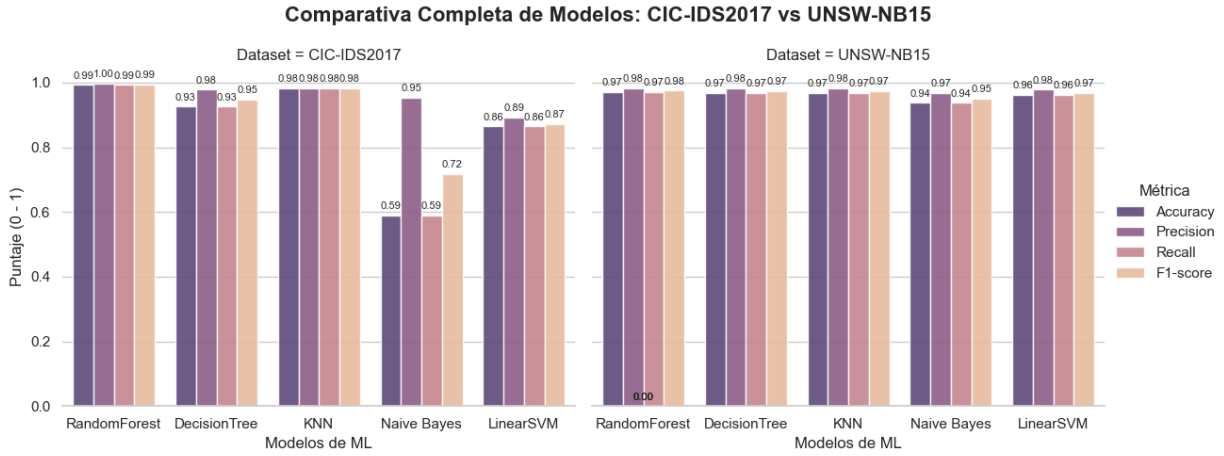


*Figura 5.3*. Gráfica de barras para representar los datos de la *tabla 5.4.*

Tras entrenar todos los modelos y realizar la primera tanda de prueba, se puede apreciar que tanto Random Forest, Decision Tree y K-Nearest Neighbors tienen un porcentaje de precisión mucho más alto que Naive Bayes y LinearSVC. Por lo tanto, Naive Bayes y LinearSVC no van a formar parte del análisis multiclase. Sin embargo, K-Nearest Neighbors tiene un tiempo de predicción muy alto, aunque las predicciones que realicen los usuarios sean de menor tamaño, sigue siendo mucho, por lo que también se descarta.

### 5.3.1. RESULTADOS DE CLASIFICACIÓN CON EL CONJUNTO DE DATOS UNSW-NB15

Para decidir qué modelo utilizar en el sistema de detección de intrusiones, he hecho una comparativa para ver qué modelo tiene mejor rendimiento por cada clase, Random Forest o Decision Tree para cada conjunto de datos.

#### Modelo Random Forest UNSW-NB15

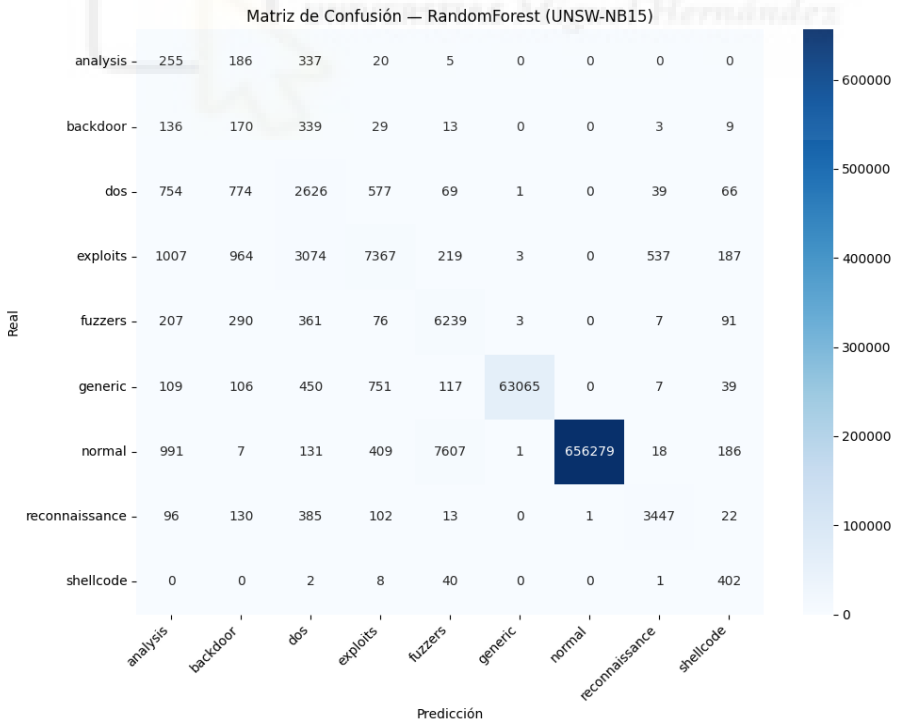



*Figura 5.4*. Matriz de confusión Random Forest UNSW-NB15.

| Clase | Precisión (Precision) | Sensibilidad (Recall) | F1-Score | Muestra |
|---|---|---|---|---|
| Analysis | 0.0717 | 0.3176 | 0.1170 | 803 |
| Backdoor | 0.0647 | 0.2432 | 0.1022 | 699 |
| DoS | 0.3408 | 0.5353 | 0.4165 | 4,906 |
| Exploits | 0.7888 | 0.5515 | 0.6492 | 13,358 |
| Fuzzers | 0.4356 | 0.8577 | 0.5778 | 7,274 |
| Generic | 0.9999 | 0.9756 | 0.9876 | 64,644 |
| Normal | 1.0000 | 0.9860 | 0.9929 | 665,629 |
| Reconnaissance | 0.8492 | 0.8215 | 0.8351 | 4,196 |
| Shellcode | 0.4012 | 0.8874 | 0.5526 | 453 |

*Tabla 5.5.* Resultados clasificación de cada clase para el modelo Random Forest de UNSW-NB15.

*Nota*. Obtención propia, tras obtener el resultado de todos los modelos.

### Modelo Decision Tree UNSW-NB15

De forma similar a Random Forest, se muestran todos los resultados obtenidos del reporte de clasificación:

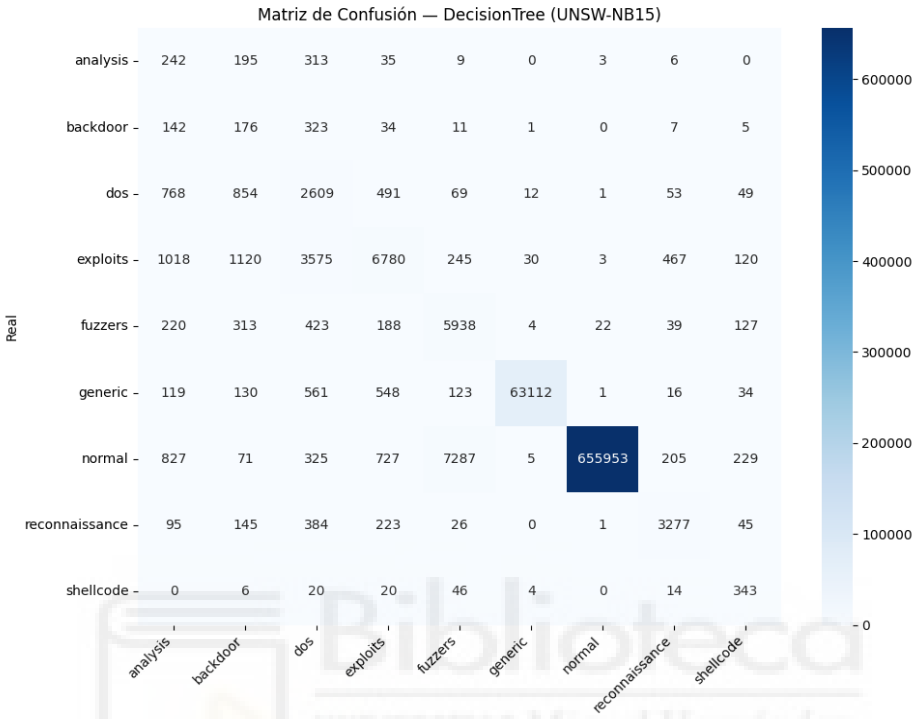



*Figura 5.5*. Matriz de confusión Decision Tree UNSW-NB15.

| Clase | Precisión (Precision) | Sensibilidad (Recall) | F1-Score | Muestra |
|---|---|---|---|---|
| **Analysis** | 0.0705 | 0.3014 | 0.1143 | 803 |
| **Backdoor** | 0.0585 | 0.2518 | 0.0949 | 699 |
| **DoS** | 0.3058 | 0.5318 | 0.3883 | 4,906 |
| **Exploits** | 0.7495 | 0.5076 | 0.6052 | 13,358 |
| **Fuzzers** | 0.4317 | 0.8163 | 0.5648 | 7,274 |
| **Generic** | 0.9991 | 0.9763 | 0.9876 | 64,644 |
| **Normal** | 1.0000 | 0.9855 | 0.9927 | 665,629 |
| **Reconnaissance** | 0.8024 | 0.7810 | 0.7915 | 4,196 |
| **Shellcode** | 0.3603 | 0.7572 | 0.4883 | 453 |

*Tabla 5.6*. Métricas obtenidas de los resultados de clasificación de cada clase.

*Nota*. Elaboración propia a partir de los resultados del modelo.

**Modelo elegido para UNSW-NB15**

Después de haber analizado el resto de los modelos se puede determinar que Random Forest ha presentado un rendimiento mejor respecto a Decision Tree. Básicamente, es una versión ligeramente mejor, a pesar de que presenta un desbalanceo en cuanto porcentaje de clasificaciones exitosas, sin embargo, esto también es resultado de la baja muestra de datos y de la desproporción entre clases, donde los modelos no pueden "aprender" a realizar clasificaciones correctas con una muestra de datos insuficiente.

### 5.3.2 RESULTADOS DE CLASIFICACIÓN CON EL CONJUNTO DE DATOS CIC-IDS2017

Para el conjunto de datos CIC-IDS2017, se aplica el mismo proceso.

**Resultados clasificación, Random Forest CIC-IDS2017**


*Figura 5.6*. Matriz de confusión Random Forest CIC-IDS2017.

| Ataque | Precisión | Sensibilidad (Recall) | F1-Score | Muestra |
|---|---|---|---|---|
| **BENIGN** | 0.9999 | 0.9921 | 0.9960 | 681,929 |
| **Bot** | 0.1584 | 0.9932 | 0.2733 | 590 |
| **DDoS** | 0.9992 | 0.9995 | 0.9994 | 38,408 |
| **DoS GoldenEye** | 0.9376 | 0.9981 | 0.9669 | 3,088 |
| **DoS Hulk** | 0.9873 | 0.9992 | 0.9932 | 69,322 |
| **DoS Slowhttptest** | 0.9306 | 0.9915 | 0.9601 | 1,650 |
| **DoS slowloris** | 0.9769 | 0.9960 | 0.9863 | 1,739 |
| **FTP-Patator** | 0.9983 | 0.9996 | 0.9990 | 2,381 |
| **PortScan** | 0.9940 | 0.9994 | 0.9967 | 47,679 |
| **SSH-Patator** | 0.9465 | 0.9994 | 0.9722 | 1,769 |
| **Web Attack - Brute Force** | 0.3620 | 0.6327 | 0.4605 | 452 |
| **Web Attack - XSS** | 0.2594 | 0.6684 | 0.3738 | 196 |

*Tabla 5.7*. Valores de medición Random Forest CIC-IDS2017.

*Nota*. Elaboración propia a partir de los resultados del modelo.

**Resultados clasificación, Decision Tree CIC-IDS2017**


*Figura 5.7*. Matriz de confusión DecisionTree CIC-IDS2017.

*Nota*. Elaboración propia a partir de los resultados del modelo.

| Clase | Precisión | Sensibilidad (Recall) | F1-Score | Muestra |
|---|---|---|---|---|
| **BENIGN** | 0.9996 | 0.9816 | 0.9905 | 681,929 |
| **Bot** | 0.1409 | 0.9797 | 0.2464 | 590 |
| **DDoS** | 0.9869 | 0.9983 | 0.9926 | 38,408 |
| **DoS GoldenEye** | 0.7729 | 0.9951 | 0.8700 | 3,088 |
| **DoS Hulk** | 0.9624 | 0.9953 | 0.9786 | 69,322 |
| **DoS Slowhttptest** | 0.7011 | 0.9921 | 0.8216 | 1,650 |
| **DoS slowloris** | 0.7304 | 0.9954 | 0.8425 | 1,739 |
| **FTP-Patator** | 0.9904 | 0.9996 | 0.9950 | 2,381 |
| **PortScan** | 0.9902 | 0.9992 | 0.9947 | 47,679 |
| **SSH-Patator** | 0.8150 | 0.9983 | 0.8974 | 1,769 |
| **Web Attack - Brute Force** | 0.1509 | 0.6527 | 0.2451 | 452 |
| **Web Attack - XSS** | 0.0674 | 0.5408 | 0.1198 | 196 |

*Tabla 5.8*. Métricas por clase Decision Tree CIC-IDS2017.

*Nota*. Elaboración propia a partir de los resultados del modelo.

**Modelo elegido CIC-IDS2017**

Tras haber analizado los dos conjuntos de datos, he podido apreciar que Random Forest presenta un mayor rendimiento en cada clase, siendo este elegido para realizar las funciones de predicción para el sistema de detección de intrusiones.

**A modo de conclusión, Random Forest realizará las funciones:**

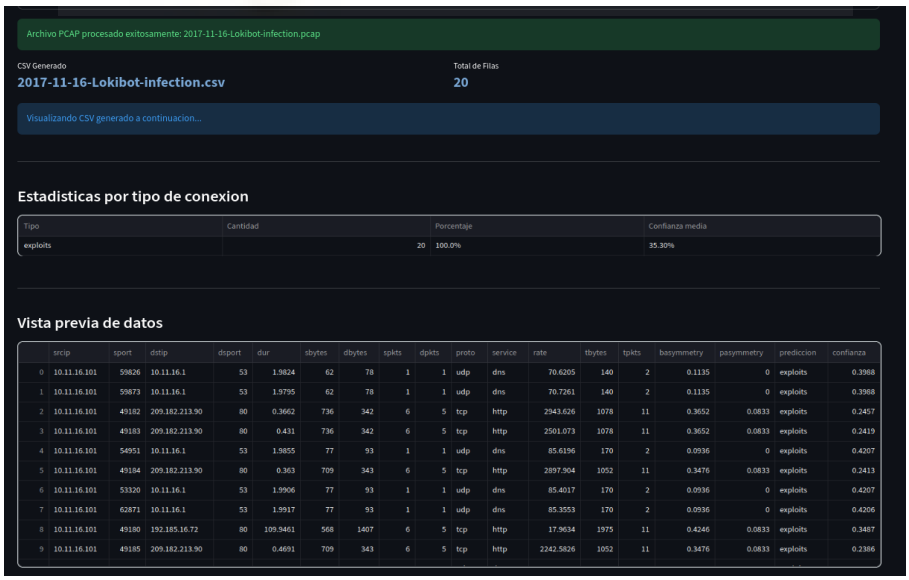
- Predicción en tiempo real, modelo CIC-IDS2017
- Predicción de un archivo PCAP para ambos conjuntos de datos.

# 6. EVALUACIÓN DEL SISTEMA INTEGRADO EN UN ENTORNO DE ATAQUES REAL

Como ya se ha seleccionado a Random Forest como modelo de inteligencia artificial y se ha explicado todos los apartados del sistema de detección de intrusiones, solo falta ponerlo en marcha y efectuar pruebas de sus dos funcionalidades: analizar archivos PCAP y analizar tráfico en tiempo real.

## 6.1 TESTING CON MODELO UNSW-NB15 SOBRE ARCHIVOS PCAP

Al introducir unos datos de entrada en formato PCAP, se obtendrá un archivo CSV con los mismos que se le asignaron al modelo UNSW-NB15 en la fase de entrenamiento. También se muestra en una tabla las estadísticas respecto al tipo de conexión que hay en el archivo. En la siguiente figura se muestra una captura de pantalla del archivo PCAP correspondiente con la infección de *exploits* sobre un conjunto de conexiones. (Sanders, 2017)




| | srcip | sport | dstip | dsport | dur | sbytes | dbytes | spkts | dpkts | proto | service | rate | tbytes | tpkts | basymmetry | pasymmetry | prediccion | confianza |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 10.11.16.101 | 59826 | 10.11.16.1 | 53 | 1.9824 | 62 | 78 | 1 | 1 | udp | dns | 70.6205 | 140 | 2 | 0.1135 | 0 | exploits | 0.3988 |
| 1 | 10.11.16.101 | 59873 | 10.11.16.1 | 53 | 1.9795 | 62 | 78 | 1 | 1 | udp | dns | 70.7261 | 140 | 2 | 0.1135 | 0 | exploits | 0.3988 |
| 2 | 10.11.16.101 | 49182 | 209.182.213.90 | 80 | 0.3662 | 736 | 342 | 6 | 5 | tcp | http | 2943.626 | 1078 | 11 | 0.3652 | 0.0833 | exploits | 0.2457 |
| 3 | 10.11.16.101 | 49183 | 209.182.213.90 | 80 | 0.431 | 736 | 342 | 6 | 5 | tcp | http | 2501.073 | 1078 | 11 | 0.3652 | 0.0833 | exploits | 0.2419 |
| 4 | 10.11.16.101 | 54951 | 10.11.16.1 | 53 | 1.9855 | 77 | 93 | 1 | 1 | udp | dns | 85.6196 | 170 | 2 | 0.0936 | 0 | exploits | 0.4207 |
| 5 | 10.11.16.101 | 49184 | 209.182.213.90 | 80 | 0.363 | 709 | 343 | 6 | 5 | tcp | http | 2897.904 | 1052 | 11 | 0.3476 | 0.0833 | exploits | 0.2413 |
| 6 | 10.11.16.101 | 53320 | 10.11.16.1 | 53 | 1.9906 | 77 | 93 | 1 | 1 | udp | dns | 85.4017 | 170 | 2 | 0.0936 | 0 | exploits | 0.4207 |
| 7 | 10.11.16.101 | 62871 | 10.11.16.1 | 53 | 1.9917 | 77 | 93 | 1 | 1 | udp | dns | 85.3553 | 170 | 2 | 0.0936 | 0 | exploits | 0.4206 |
| 8 | 10.11.16.101 | 49180 | 192.185.16.72 | 80 | 109.9461 | 568 | 1407 | 6 | 5 | tcp | http | 17.9634 | 1975 | 11 | 0.4246 | 0.0833 | exploits | 0.3487 |
| 9 | 10.11.16.101 | 49185 | 209.182.213.90 | 80 | 0.4691 | 709 | 343 | 6 | 5 | tcp | http | 2242.5826 | 1052 | 11 | 0.3476 | 0.0833 | exploits | 0.2386 |

*Figura 6.1*. Análisis PCAP para lokibot-infection.

El archivo PCAP es procesado con éxito y la mayoría de los ataques se corresponden con *exploits*, que eran el tipo de ataque. Cabe destacar que la clase *exploits* era de las mejores predichas por el modelo UNSW-NB15, por lo que es normal que haya acertado en este caso.

## 6.2 PRUEBA DE ATAQUES EN TIEMPO REAL MEDIANTE KALI LINUX

Como he explicado en el apartado 4, se ha configurado un entorno de pruebas para poder evaluar la funcionalidad del sistema de detección de intrusiones en un contexto real. Cada ataque quedará registrado en una sesión diferente, luego he hecho un listado con todas las conexiones registradas en una sesión en conjunto con la del ataque. Se listarán casos de acierto del modelo y casos de error. Se han probado los ataques con constan el conjunto de datos CIC-IDS2017.

En la siguiente tabla he descrito toda la información respecto a las pruebas a realizar, se ha utilizado un servidor Apache2 como host víctima y una máquina virtual Kali Linux como atacante. Desde esta máquina virtual se ejecutarán los comandos y sistema de detección de intrusiones los detectará.

| Elemento | Detalle |
|---|---|
| **Máquina atacante** | Kali Linux (misma red local), IP 192.168.100.219 |
| **Máquina víctima (IDS)** | Ubuntu, IP 192.168.100.221, interfaz wlp3s0 |
| **Servidor web** | servidor web víctima Python en puerto 80 |
| **API** | FastAPI en 127.0.0.1:8000 |
| **Modelo ML** | CIC-IDS2017 (Random Forest) |
| **Captura** | CICFlowMeter en tiempo real sobre wlp3s0 o la interfaz de red correspondiente |

*Tabla 6.1*. Descripción del procedimiento para realizar los test de captura.

### 6.2.1 TRÁFICO NORMAL

Para monitorizar el tráfico normal, simplemente se pulsa el botón para iniciar captura de tráfico, mientras se realizan algunas peticiones a servidores web comunes a páginas como lo son *google*, *gmail*, etc. De esta forma se capta todo el tráfico que transcurre por la interfaz seleccionada, que en este caso es wlp3s0 la que ha sido asignada. Si no hay ninguna predicción que se corresponda a un ataque, se muestra la notificación en verde indicando la ausencia de estas.


*Figura 6.2*. Resultado de una captura de tráfico sin amenazas.

### 6.2.2 ESCANEO DE PUERTOS MEDIANTE NMAP

El primer ataque que he probado se corresponde con el escaneo de puertos *portscan*, este consiste en enviar un gran número de peticiones al a víctima con el fin de descubrir que servicios están ejecutándose, deduciendo esto en base al puerto que reciba respuesta. Si la máquina atacante recibe respuesta eso significa que el servicio asociado a este puerto está abierto.

Debido a que el conjunto de datos no es capaz de detectar el escaneo de puertos mediante el conjunto de datos original, se ha programado una configuración adicional llamada escaneo de puertos, donde si se recibe más de 5 peticiones provenientes de la misma dirección IP el sistema de detección de intrusiones clasifica esa conexión como escaneo de puertos.

He de destacar el uso de esta implementación debido a que los sistemas de detección de intrusiones basados en inteligencia artificial, (que son muy pocos) no son capaces de discernir entre una conexión normal y una de escaneo de puertos. La razón principal se debe a que al realizar un escaneo se realizan muchas conexiones y el modelo de inteligencia artificial realiza predicciones de forma singular y no como un conjunto, por lo que esta predicción reside en una regla heurística que he programado y no reside en una predicción íntegra del modelo de inteligencia artificial.

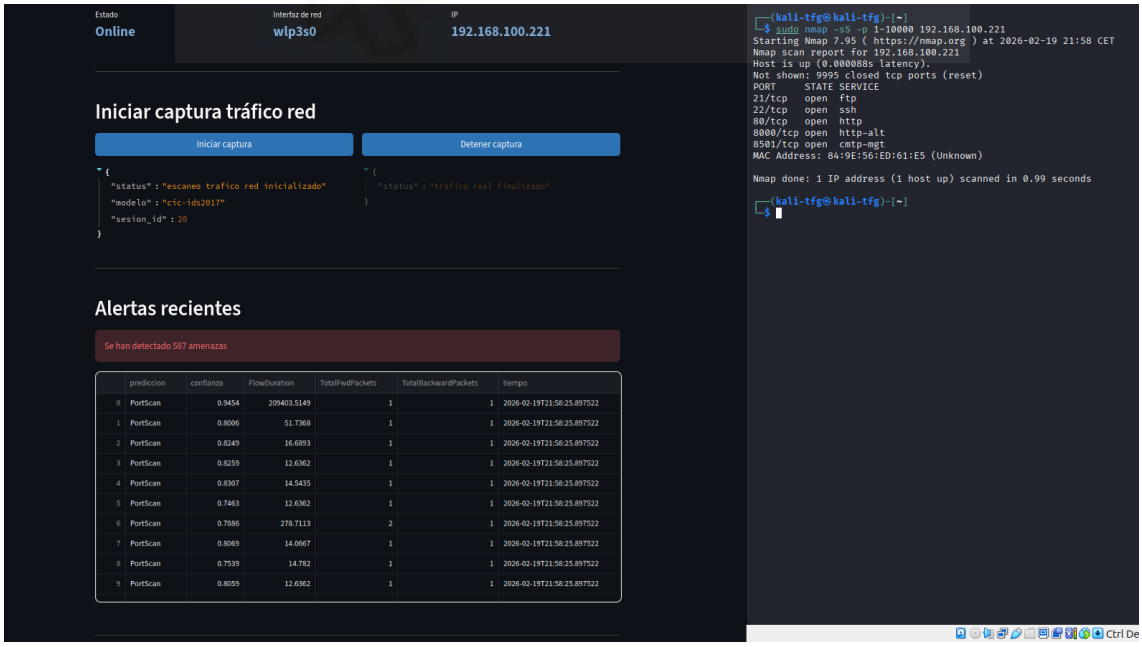



*Figura 6.3*. Captura de ataque PortScan.

Como se puede ver en la figura del resultado final, el análisis es satisfactorio ya que se puede predecir el ataque sin problemas con un alto nivel de confianza.

### 6.2.3 ATAQUE DDOS MEDIANTE HPING3

Para ejecutar una simulación de ataque basado en deshabilitación de servicios, he utilizado la herramienta hping3 que pertenece al conjunto de herramientas de Kali Linux, y se considera como un ataque DDoS genérico.

En la figura se puede apreciar el resultado, el cual no es puramente positivo, debido a que detecta el ataque como un deshabilitación de servicios, sin embargo, este es clasificado como escaneos de puertos (normal porque se realizan muchas peticiones a muchos puertos distintos) y DoS Slowloris, que es otro tipo de ataque de deshabilitación de servicios presente en el conjunto de datos.

A pesar de no detectar el ataque como des habilitación de servicios “puro”, creo que el resultado sigue siendo positivo porque el usuario que esté ejecutando el sistema de detección de intrusiones será consciente de que hay un número muy alto de amenazas correspondiente a ataques DoS Slowloris y esta amenaza no quedará desatendida.

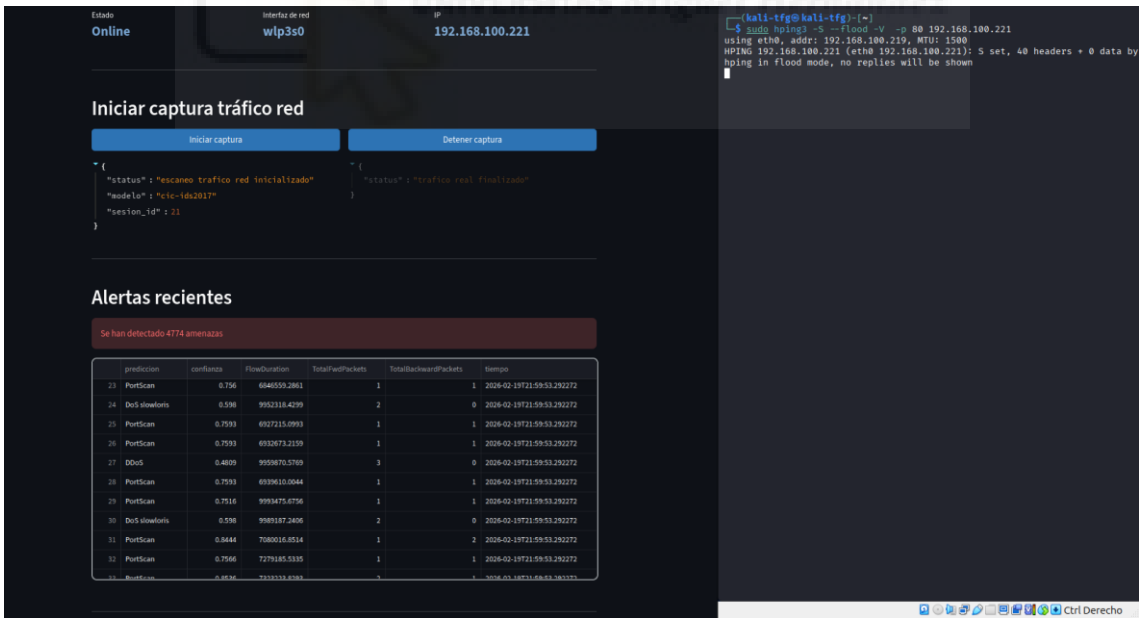



*Figura 6.4*. Captura del ataque *h3ping*.

### 6.2.4 ATAQUE FUERZA BRUTA MEDIANTE SSH PATATHOR

Para este tercer ataque he decidido probar cómo sería el funcionamiento de un ataque de fuerza bruta, para esto tenemos SSH Patator y FTP Patator, ambos se encargan de hacer bypass o intentar forzar el inicio de sesión en un servidor víctima mediante el intento masivo de inicio de sesión, las claves para intentar este inicio de sesión provienen de un documento TXT, el documento que viene de prueba se llama *rockyou.txt* y ha sido el empelado en esta simulación. El resultado es positivo y se muestra en la siguiente figura:

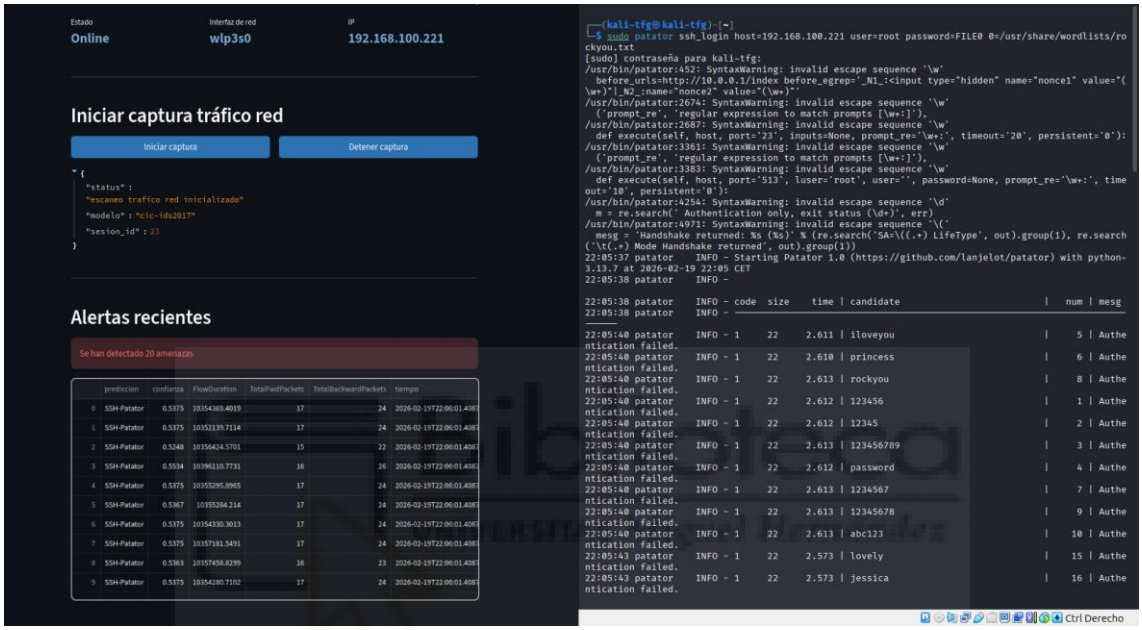



*Figura 6.5*. Captura del ataque SSH Patator.

En la parte de la derecha se aprecia los intentos e inicio de sesión con todas las contraseñas y en la derecha se puede visualizar que el ataque es detectado por el sistema de detección de intrusiones.

### 6.2.5 ATAQUE DOS ESPECÍFICO SLOWHTTPTEST

Para finalizar, he realizado una prueba para determinar si el sistema de detección de intrusiones es capaz de detectar el tipo concreto de ataque DDoS que se está perpetrando. Para ello, he realizado una simulación del ataque Slowhttpest que consta dentro del conjunto de datos CIC-IDS2017. En la figura se puede ver que el ataque es detectado con éxito, junto con algunas conexiones que se han clasificado también como escaneo de puertos debido a que el ataque envía peticiones HTTP a múltiples puertos de una dirección IP.

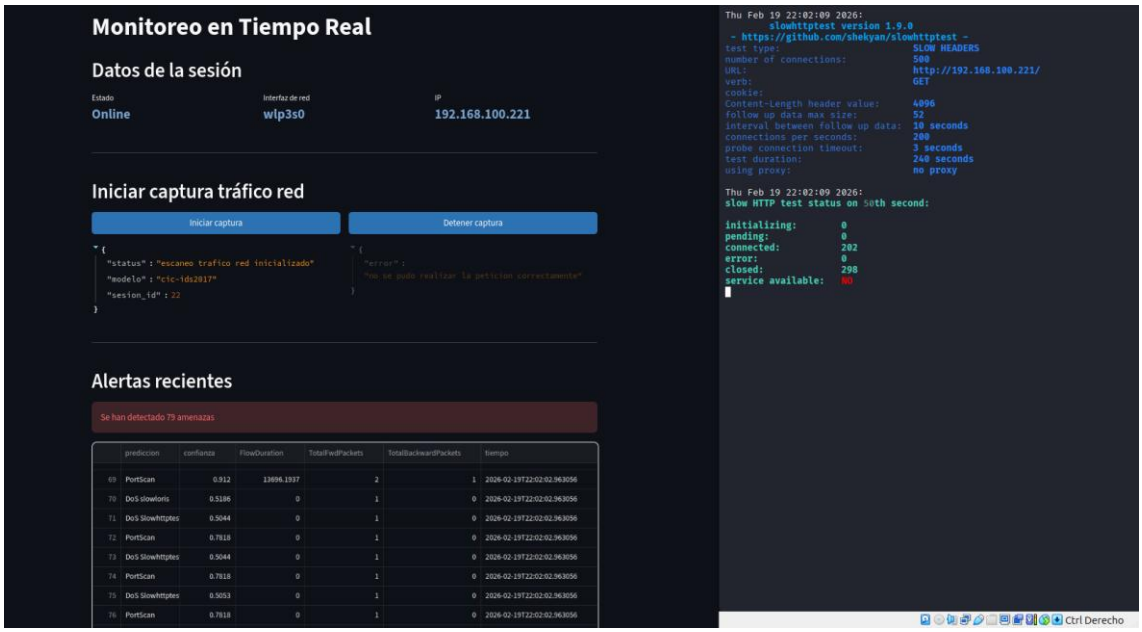



Figura 6.6. Captura del ataque DoS slowloris.

# 7. CONCLUSIONES

Respecto a los objetivos propuestos en el capítulo uno, puedo decir que se han satisfecho la mayoría, ya que he conseguido desarrollar la aplicación web con las funcionalidades descritas. Los objetivos que se han satisfecho son los siguientes:

- Monitorización de red y análisis de conexiones: mediante CICFlowmeter y Zeek se pueden desglosar las conexiones obtenidas del tráfico de red procedentes de paquetes.
- Análisis y detección de patrones en conexiones: a lo largo de este trabajo se ha ido perfeccionando el conocimiento que he tenido sobre los parámetros que constituyen cada conexión y su relevancia en el contexto de ciberseguridad.
- Entrenar un modelo para realizar predicciones: aunque aún no es perfecto el aprendizaje automático en el aprendizaje supervisado en el campo de ciberseguridad, puede generar información útil y relevante para investigadores o gente interesada en tener más conocimientos sobre la naturaleza, conexiones, ataques de red y protocolos de red.
- API y aplicación cliente servidor: la aplicación que he desarrollado, permite realizar dos funcionalidades básicas pero efectivas mediante una interfaz de usuario simple e intuitiva.

**Puntos positivos razonados tras realizar el trabajo**

- **El uso de algoritmos de inteligencia artificial puede desempeñar un papel importante en la ciberseguridad**: puede ayudar al análisis de conexiones. Personas que quieran tener un mayor control sobre el tráfico que circula por una red privada, pueden usar modelos de inteligencia artificial entrenados para determinados tipos de ataques.
- **Los modelos especializados pueden llegar a tener mucha precisión**: si se entrena un modelo sobre un conjunto de datos bien elaborado y etiquetado, se puede llegar a entrenar un modelo que realice predicciones con un alto porcentaje de acierto, si se mantiene el formato de los datos vistos en entrenamiento.

- **Facilidad de uso y de ejecución:** si se optimiza bien un programa IDS puede ejecutarse de fondo y generar alertas que mantenga al usuario al tanto.

**Puntos negativos razonados tras realizar el trabajo**

- **El papel del aprendizaje supervisado es limitado:** si bien es útil, sigue teniendo muchas limitaciones. Debido a tener que depender de un conjunto de datos grandes obtenidos por otras personas o instituciones, tener grandes conjuntos de datos de tráfico de red cualitativo es complicado, limitando, con ello, la efectividad de los modelos de inteligencia artificial.
- **Dependencia de datos estáticos e ineficacia frente nuevos ataques**: al tener que utilizar datos de entrenamiento para aprender, es complicado generar nuevas reglas para modelos, ya que no es fácil obtener datos de entrenamiento nuevos y siempre habrá nuevos ataques surgiendo que no se podrán detectar.
- **Falta de ejemplos de casos reales**: los sistemas de intrusiones basados en anomalías escasean en el "mundo real", la mayoría están basados en reglas como Snort o Suricata. Esto puede darse debido a la falta de investigación de inteligencia artificial en este ámbito, o bien falta de inversión por parte de empresas que quieran desarrollar proyectos de este tipo.

# 8. BIBLIOGRAFÍA


Apache Software Foundation. (2026). *Apache Spark: Motor de procesamiento de datos unificado*. https://spark.apache.org/

Bayes, T., & Price, R. (1763). Un ensayo para resolver un problema en la doctrina de las probabilidades. *Philosophical Transactions of the Royal Society of London*, *53*, 370-418.

Bishop, C. M. (2006). *Reconocimiento de patrones y aprendizaje automático*. Springer.

Breiman, L. (2001). Bosques aleatorios. *Machine Learning*, *45*(1), 5-32.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Árboles de clasificación y regresión*. CRC Press.

Canadian Institute for Cybersecurity. (s.f.). *Conjunto de datos CIC-IDS2017*. https://www.unb.ca/cic/datasets/ids-2017.html

Caswell, T. A., Droettboom, M., Lee, A., Hunter, J., Firing, E., Stansby, D., Andrade, E. S., Hoffmann, T., Klymak, J., Varoquaux, N., Nielsen, J. H., Root, B., May, R., Elson, P., Seppänen, J. K., Dale, D., Lee, J.-J., McDougall, D., Straw, A., ... Silvester, S. (2021). *Matplotlib: Biblioteca de visualización para Python* (Versión 3.5.0). https://matplotlib.org/

Cortes, C., & Vapnik, V. (1995). Redes de vectores de soporte. *Machine Learning*, *20*(3), 273-297.

Código del proyecto en GitHub. (2026). https://github.com/juanan-00/tfg-ids

Cover, T., & Hart, P. (1967). Clasificación de patrones mediante vecinos más cercanos. *IEEE Transactions on Information Theory*, *13*(1), 21-27.

Fielding, R. T. (2000). *Estilos arquitectónicos y el diseño de arquitecturas de software basadas en red* [Tesis doctoral, University of California]. Repositorio institucional UCI. https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Géron, A. (2019). *Aprendizaje automático práctico con Scikit-Learn, Keras y TensorFlow* (2.ª ed.). O'Reilly Media.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Aprendizaje profundo*. MIT Press.

Hipp, R. D. (2026). *SQLite: Motor de base de datos SQL embebido*. https://www.sqlite.org/

Instituto Nacional de Ciberseguridad. (2017). *Guía de diseño y configuración de IPS, IDS y SIEM en sistemas de control industrial*. https://www.incibe.es/sites/default/files/contenidos/guias/doc/certsi_diseno_configuracion_ips_ids_siem_en_sci.pdf

Lashkari, A. H. (2026). *CICFlowMeter: Herramienta de extracción de características de flujo de red*. Canadian Institute for Cybersecurity. https://www.unb.ca/cic/research/applications.html

Mitchell, T. M. (1997). *Aprendizaje automático*. McGraw-Hill.

Moustafa, N., & Slay, J. (2015). UNSW-NB15: Un conjunto de datos completo para sistemas de detección de intrusiones en redes. En *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1-6). IEEE. https://doi.org/10.1109/MilCIS.2015.7348942

Pandas Development Team. (2026). *Pandas: Biblioteca de análisis y manipulación de datos para Python*. https://pandas.pydata.org/

Python Software Foundation. (2026). *Python: Lenguaje de programación*. https://www.python.org/

Rish, I. (2001). Un estudio empírico del clasificador Naive Bayes. En *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence* (Vol. 3, n.º 22, pp. 41-46).

Russell, S., & Norvig, P. (2020). *Inteligencia artificial: Un enfoque moderno* (4.ª ed.). Pearson.

Sanders, B. (2017). 2017-11-16 - Traffic analysis exercise - LokiBot infection. Malware-Traffic-Analysis.net. https://www.malware-traffic-analysis.net/2017/11/16/index.html

Scikit-learn Developers. (2026). *Scikit-learn: Aprendizaje automático en Python*. https://scikit-learn.org/

Shannon, C. E. (1948). Una teoría matemática de la comunicación. *Bell System Technical Journal*, *27*(3), 379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Hacia la generación de un nuevo conjunto de datos de detección de intrusiones y caracterización del tráfico de intrusión. En *Proceedings of the 4th International Conference on Information Systems Security and Privacy* (pp. 108-116). SCITEPRESS. https://doi.org/10.5220/0006639801080116

Streamlit Inc. (2026). *Streamlit: Framework para aplicaciones de datos en Python*. https://streamlit.io/

Sutton, R. S., & Barto, A. G. (2018). *Aprendizaje por refuerzo: Una introducción* (2.ª ed.). MIT Press.

Van Rossum, G., & Drake, F. L. (2009). *Manual de referencia de Python 3*. CreateSpace.

Waskom, M. (2021). Seaborn: Visualización de datos estadísticos en Python. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Zeek Community. (2026). *Zeek: Monitor de seguridad de red*. https://zeek.org/

Zhu, X. (2005). *Revisión bibliográfica sobre aprendizaje semi-supervisado* (Informe Técnico n.º 1530). Department of Computer Sciences, University of Wisconsin-Madison.

Figuras:

*Figura 2.1*. Popova, L. (2023). *Unraveling AI Complexity - A Comparative View of AI, Machine Learning, Deep Learning, and Generative AI.png* [Diagrama]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Unraveling_AI_Complexity_-_A_Comparative_View_of_AI,_Machine_Learning,_Deep_Learning,_and_Generative_AI.png

*Figura 2.2*. Antimundo. (2016). *Red neuronal artificial.svg* [Diagrama]. Wikimedia Commons. commons.wikimedia.org https://upload.wikimedia.org/wikipedia/commons/9/94/Red_neuronal_artificial.svg

*Figura 2.3*. TseKiChun. (2017, 13 de mayo). *Random forest explain* [Diagrama]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Random_forest_explain.png

*Figura 2.4*. Evolutions algoritmos. (2019, 25 de septiembre). *ÁRBOL DE DECISIONES SIMPLE* [Diagrama]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:%C3%81RBOL_DE_DECISIONES_SIMPLE.png

*Figura 2.5*. Jeremybeauchamp. (2020, 24 de junio). *Decision Tree vs. Random Forest* [Diagrama]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Decision_Tree_vs._Random_Forest.png

*Figura 2.6*. Larhmam. (2018, 14 de mayo). *SVM margin* [Diagrama]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:SVM_margin.png

*Figura 2.7*. Ajanki, A. (2016, 26 de agosto). *KnnClassification* [Diagrama]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:KnnClassification.svg

*Figura 3.1*. Tiangolo. (2018). *FastAPI logo.svg* [Logotipo]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:FastAPI_logo.svg

*Figura 3.2*. Streamlit. (2022). *Streamlit logo primary colormark darktext* [Logotipo]. Wikimedia Commons. https://commons.wikimedia.org/wiki/File:Streamlit-logo-primary-colormark-darktext.png

# 9. ANEXO

## APÉNDICE A. CARACTERÍSTICAS SOBRES LOS DATASETS

A.1 Tabla con las características de la sección Flow (UNSW-NB15)

| | Número | Nombre | Tipo de dato | Descripción |
|---|---|---|---|---|
| Flow | 1 | *srcip* | N | Dirección IP Origen |
| | 2 | *sport* | I | Origen número de puerto |
| | 3 | *dstip* | N | Dirección IP destino |
| | 4 | *dsport* | I | Número de puerto de destino |
| | 5 | *proto* | N | Protocolo en capa de transporte |

*Tabla A.1*. Características de la tabla Flow.

A.2 Tabla con las características de la sección Basic (UNSW-NB15)

| | Número | Nombre | Tipo de dato | Descripción |
|---|---|---|---|---|
| Basic | 6 | *state* | N | Estado y protocolo independiente. Ej: http, smtp, etc. |
| | 7 | *dur* | F | Duración total de la conexión. |
| | 8 | *sbytes* | I | Total de bytes enviados de origen a destino. |
| | 9 | *dbytes* | I | Total de bytes enviados de destino a origen. |
| | 10 | *sttl* | I | Tiempo de vida de origen a destino. (TTL). |
| | 11 | *dttl* | I | Tiempo de vida de destino a origen. (TTL). |
| | 12 | *sloss* | I | Paquetes de origen retransmitidos o perdidos. |
| | 13 | *dloss* | I | Paquetes de destino retransmitidos o perdidos. |
| | 14 | *service* | N | Protocolo en capa de aplicación: http, ftp, ssh, dns…, otro (-). |
| | 15 | *sload* | F | Tasa de bits de origen. |
| | 16 | *dload* | F | Tasa de bits de destino. |
| | 17 | *spkts* | I | Número de paquetes de origen. |
| | 18 | *dpkts* | I | Número de paquetes de destino. |

*Tabla A.2*. Características de la tabla Basic.

A.3 Tabla con las características de la sección Content (UNSW-NB15)

| | Número | Nombre | Tipo de dato | Descripción |
|---|---|---|---|---|
| Content | 19 | *Swin* | I | Tamaño de ventana TCP de origen. |
| | 20 | *Dwin* | I | Tamaño de ventana TCP de destino. |
| | 21 | *Stcpb* | I | Número de secuencia TCP de origen. |
| | 22 | *Dtcpb* | I | Número de secuencia TCP de destino. |
| | 23 | *Smeansz* | I | Media del tamaño de paquete de flujo transmitido por el origen. |
| | 24 | *Dmeansz* | I | Media del tamaño de paquete de flujo transmitido por el destino. |
| | 25 | *Trans_Depth* | I | Nivel de capa de profundidad de intercambio de datos al realizar el protocolo http. |
| | 26 | *Res_bdy_len* | I | Tamaño del contenido de los datos transferidos de servidor utilizando el servicio http. |

*Tabla A.3*. Características de la tabla Content.

A.4 Tabla con las características de la sección Time (UNSW-NB15)

| | # | Nombre | Tipo de dato | Descripción |
|---|---|---|---|---|
| Time | 27 | *Sjit* | F | Tiempo ocurrido de jitter en la conexión origen (ms) |
| | 28 | *Djit* | F | Tiempo ocurrido de jitter en la conexión destino (ms) |
| | 29 | *Stime* | T | Registro tiempo de inicio (Start time) |
| | 30 | *Ltime* | T | Registro tiempo fin conexión (Last time) |
| | 31 | *Sinpkt* | F | Tiempo origen de Inter-packet time, tiempo que pasa entre la llegada (o envío) de un paquete de datos y el siguiente (ms) |
| | 32 | *Dinpkt* | F | Destination inter-packet arrival time (ms) |
| | 33 | *Tcprtt* | F | La suma de 'synack' y 'ackdat' de TCP |
| | 34 | *Synack* | F | El tiempo ocurrido entre el envío del paquete SYN y el recibo del paquete SYN-ACK en una conexión TCP. El RTT inicial (Initial Round-Trip Time) |
| | 35 | *Ackdat* | F | El tiempo ocurrido entre la recepción del paquete SYN-ACK y el envío del paquete ACK en una conexión TCP. |

*Tabla A.4*. Características de la tabla Content.

A.5 Tabla con las características de la sección Labeled (UNSW-NB15)

| | Número | *Nombre* | Tipo de dato | Descripción |
|---|---|---|---|---|
| Labelled | 48 | *Attack_ cat* | N | Tipo de ataque que ha sido perpetrado (DDos, Fuzzer, Ransomware, Fuzzer) |
| | 49 | *Label* | B | 0. Determina una conexión normal, 1 Determina un ataque. |

*Tabla A.5*. Características de la tabla Labeled.

A.6 Tabla con los datos de cada archivo CSV del *dataset* CIC-IDS2017

| Día | Nombre del Archivo | Ataques Encontrados |
|---|---|---|
| Lunes | Monday-WorkingHours.pcap_ISCX.csv | Benign (Normal human activities) |
| Martes | Tuesday-WorkingHours.pcap_ISCX.csv | Benign, FTP-Patator, SSH-Patator |
| Miércoles | Wednesday-workingHours.pcap_ISCX.csv | Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed |
| Jueves | Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv | Benign, Web Attack – Brute Force, Web Attack – Sql Injection, Web Attack – XSS |
| Jueves | Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv | Benign, Infiltration |
| Viernes | Friday-WorkingHours-Morning.pcap_ISCX.csv | Benign, Bot |
| Viernes | Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv | Benign, PortScan |
| Viernes | Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv | Benign, DDoS |

*Tabla A.6*. Archivos CSV de CIC-IDS2017.

A.7 Cantidad de datos según archivos CSV de los *datasets* UNSW-NB15 y CIC-IDS2017

| **Archivos CIC-IDS2017** | **Muestras CIC-IDS2017** | **Archivos UNSW-NB15** | **Muestras** |
|---|---|---|---|
| Monday-WorkingHours.pcap_ISCX.csv | 450,000 | UNSW-NB15_1.csv | 700,000 |
| Tuesday-WorkingHours.pcap_ISCX.csv | 440,000 | UNSW-NB15_2.csv | 700,000 |
| Wednesday-workingHours.pcap_ISCX.csv | 705,000 | UNSW-NB15_3.csv | 700,000 |
| Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv | 180,000 | UNSW-NB15_4.csv | 440,044 |
| Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv | 170,000 | | |
| Friday-WorkingHours-Morning.pcap_ISCX.csv | 140,000 | | |
| Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv | 170,000 | | |
| Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv | 200,000 | | |
| Total CIC-IDS2017 | 2,455,000 | Total UNSW-NB15 | 2,540,044 |

*Tabla A.7*. Muestra de datos por archivos CSV de CIC-IDS2017 y UNSW-NB15.

## APÉNDICE B. RESULTADOS RESTANTES DEL REPORTE DE CLASIFICACIÓN.

B1. Resultados de clasificación por clase para K-Nearest Neighbors (CIC-IDS2017)

| Clase | Precisión | Sensibilidad (Recall) | F1-Score | Muestra |
|---|---|---|---|---|
| BENIGN | 0.998 | 0.9747 | 0.9871 | 681,929 |
| Bot | 0.1240 | 0.9864 | 0.2203 | 590 |
| DDoS | 0.9876 | 0.9963 | 0.9919 | 38,408 |
| DoS GoldenEye | 0.8382 | 0.9984 | 0.9113 | 3,088 |
| DoS Hulk | 0.9766 | 0.9976 | 0.9870 | 69,322 |
| DoS Slowhttptest | 0.8567 | 0.9927 | 0.9197 | 1,650 |
| DoS Slowloris | 0.8196 | 0.9931 | 0.8981 | 1,739 |
| FTP Patator | 0.9746 | 0.9975 | 0.9859 | 2,381 |
| PortScan | 0.8370 | 0.9985 | 0.9107 | 47,679 |
| SSH Patator | 0.8792 | 1.0000 | 0.9357 | 1,769 |
| Web Attack – Brute Force | 0.4527 | 0.7301 | 0.5588 | 452 |
| Web Attack XSS | 0.2485 | 0.4286 | 0.3146 | 196 |

*Tabla B.1*. Resultado de clasificación del modelo K-Nearest-Neighbors CIC-IDS2017.

B2. Resultados de clasificación por clase para Naive Bayes (CIC-IDS2017)

| **Clase** | **Precisión** | **Sensibilidad (Recall)** | **F1-Score** | **Muestra** |
|---|---|---|---|---|
| BENIGN | 0.9993 | 0.6897 | 0.8161 | 681,929 |
| Bot | 0.0034 | 0.9966 | 0.0068 | 590 |
| DDoS | 0.9044 | 0.5463 | 0.6812 | 38,408 |
| DoS GoldenEye | 0.0709 | 0.9825 | 0.1323 | 3,088 |
| DoS Hulk | 0.9284 | 0.6998 | 0.7981 | 69,322 |
| DoS Slowhttptest | 0.3035 | 0.6109 | 0.4056 | 1,650 |
| DoS Slowloris | 0.2259 | 0.9270 | 0.3633 | 1,739 |
| FTP Patator | 0.9538 | 0.9979 | 0.9754 | 2,381 |
| PortScan | 0.9925 | 0.9689 | 0.9805 | 47,679 |
| SSH Patator | 0.3222 | 0.9955 | 0.4869 | 1,769 |
| Web Attack – Brute Force | 0.0190 | 0.9004 | 0.0371 | 452 |
| Web Attack XSS | 0.0013 | 0.0051 | 0.0021 | 196 |

*Tabla B.2*. Resultado de clasificación del modelo Naive Bayes CIC-IDS2017.

B3. Resultados de clasificación por clase para Linear SVM (CIC-IDS2017)

| **Clase** | **Precisión** | **Sensibilidad (Recall)** | **F1-Score** | **Muestra** |
|---|---|---|---|---|
| BENIGN | 0.9993 | 0.6897 | 0.8161 | 681,929 |
| Bot | 0.0034 | 0.9966 | 0.0068 | 590 |
| DDoS | 0.9044 | 0.5463 | 0.6812 | 38,408 |
| DoS GoldenEye | 0.0709 | 0.9825 | 0.1323 | 3,088 |
| DoS Hulk | 0.9284 | 0.6998 | 0.7981 | 69,322 |
| DoS Slowhttptest | 0.3035 | 0.6109 | 0.4056 | 1,650 |
| DoS Slowloris | 0.2259 | 0.9270 | 0.3633 | 1,739 |
| FTP Patator | 0.9538 | 0.9979 | 0.9754 | 2,381 |
| PortScan | 0.9925 | 0.9689 | 0.9805 | 47,679 |
| SSH Patator | 0.3222 | 0.9955 | 0.4869 | 1,769 |
| Web Attack – Brute Force | 0.0190 | 0.9004 | 0.0371 | 452 |
| Web Attack XSS | 0.0013 | 0.0051 | 0.0021 | 196 |

*Tabla B.3*. Resultado de clasificación del modelo Linear SVM CIC-IDS2017.

B4. Resultados de clasificación por clase para K-Nearest Neighbors (UNSW-NB)

| **Clase** | **Precisión** | **Sensibilidad (Recall)** | **F1-Score** | **Muestra** |
|---|---|---|---|---|
| Analysis | 0.0603 | 0.7385 | 0.1116 | 803 |
| Backdoor | 0.0698 | 0.1245 | 0.0894 | 699 |
| DoS | 0.3285 | 0.2313 | 0.2715 | 4,906 |
| Exploits | 0.7893 | 0.5292 | 0.6336 | 13,358 |
| Fuzzers | 0.4357 | 0.8448 | 0.5749 | 7,274 |
| Generic | 0.9990 | 0.9786 | 0.9887 | 64,644 |
| Normal | 1.0000 | 0.9861 | 0.9930 | 665,629 |
| Reconnaissance | 0.8112 | 0.7815 | 0.7961 | 4,196 |
| Shellcode | 0.4785 | 0.6137 | 0.5377 | 453 |

*Tabla B.4*. Resultado de clasificación del modelo K-Nearest-Neighbors UNSW-NB15.

B5. Resultados de clasificación por clase para Naive Bayes (UNSW-NB)

| **Clase** | **Precisión** | **Sensibilidad (Recall)** | **F1-Score** | **Muestra** |
|---|---|---|---|---|
| Analysis | 0.0135 | 0.0672 | 0.0255 | 803 |
| Backdoor | 0.0499 | 0.5064 | 0.0908 | 699 |
| DoS | 0.3396 | 0.0171 | 0.0254 | 4,906 |
| Exploits | 0.3396 | 0.4263 | 0.3780 | 13,358 |
| Fuzzers | 0.3310 | 0.2255 | 0.2682 | 7,274 |
| Generic | 0.9887 | 0.9808 | 0.9808 | 64,644 |
| Normal | 0.9999 | 0.9823 | 0.9823 | 665,629 |
| Reconnaissance | 0.1397 | 0.1797 | 0.1572 | 4,196 |
| Shellcode | 0.0285 | 0.9978 | 0.0554 | 453 |

*Tabla B.5.* Resultado de clasificación del modelo Naive Bayes UNSW-NB15.

B6. Resultados de clasificación por clase para Linear SVM (UNSW-NB)

| **Clase** | **Precisión** | **Sensibilidad (Recall)** | **F1-Score** | **Muestra** |
|---|---|---|---|---|
| Analysis | 0.0380 | 0.1681 | 0.0620 | 803 |
| Backdoor | 0.0986 | 0.1474 | 0.1181 | 699 |
| DoS | 0.3228 | 0.6040 | 0.4108 | 4,906 |
| Exploits | 0.7091 | 0.4255 | 0.5319 | 13,358 |
| Fuzzers | 0.3606 | 0.7130 | 0.4789 | 7,274 |
| Generic | 0.9954 | 0.9755 | 0.9854 | 64,644 |
| Normal | 0.9999 | 0.9823 | 0.9910 | 665,629 |
| Reconnaissance | 0.4186 | 0.7057 | 0.5255 | 4,196 |
| Shellcode | 0.1232 | 0.4084 | 0.1898 | 543 |

*Tabla B.6*. Resultado de clasificación del modelo Linear SVM UNSW-NB15.