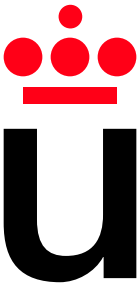


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

**Grado en Ingeniería de la Ciberseguridad**

**Curso 2023-2024**

**Trabajo Fin de Grado**

# CIBERSEGURIDAD Y CIENCIA DE DATOS: DETECCIÓN DE ATAQUES DE DENEGACIÓN DE SERVICIO

**Autor: Rodrigo Regaliza Alonso**
**Tutor: Isaac Martín de Diego**
**Co-tutora: Marina Cuesta Santa Teresa**

**3 de julio de 2024**

# Agradecimientos

Este trabajo está dedicado a mis padres, Javier y Mari Paz, quienes siempre han estado a mi lado desde que nací y me han impulsado para ser la persona que soy ahora. Este trabajo es por vosotros, por vuestro esfuerzo, por vuestro apoyo y por vuestro amor.

Además, quiero agradecer a Ángela por haber estado a mi lado y apoyado durante este tiempo. A Daniel, Fernando, Óscar, Rubén, José Luis, Jorge y Javier por haberme acompañado durante esta gran etapa.

Finalmente, también quiero agradecer a Marina y a Isaac por haber ayudado y guiado en todo lo posible para lograr completar este trabajo de fin de grado.

# Resumen

Los ataques de Denegación de Servicio son una amenaza cada vez más frecuente en los incidentes de ciberseguridad. La constante evolución de los sistemas y las telecomunicaciones entre estos representan grandes avances y a su vez una gran complejidad que hace que alcanzar la seguridad total sea un objetivo imposible. Frente a estas amenazas, existen cada vez más soluciones que pueden aplicarse como contramedidas.

Por ese motivo, este trabajo tiene como objetivo principal el desarrollo de un modelo de aprendizaje automático para la detección de ataques de Denegación de Servicio utilizando el protocolo HTTP (*HyperText Transfer Protocol*) en tiempo real. En concreto, para la detección de ataques generados por las herramientas *Golden Eye* y *Slowloris*.

Para alcanzar el objetivo del proyecto, se sigue una metodología de Ciencia de Datos con la que se analizan los registros de tráfico recopilados en un servidor web con el propósito de identificar patrones de tráfico malicioso. Este proceso incluye la preparación y limpieza de los datos, un análisis exploratorio de los datos para identificar patrones y anomalías, y la construcción de modelos utilizando árboles de decisión y *Random Forest*. Tras haber entrenado los modelos, se ha procedido a evaluar los resultados, los cuales han sido satisfactorios con un rendimiento alto.

Finalmente, una vez completado el desarrollo del modelo, se ha puesto en práctica con una simulación de un caso real en el que nuevas muestras de tráfico llegaban de forma secuencial al modelo. Sin embargo, se ha observado que el resultado de la simulación ha sido inferior al obtenido durante el desarrollo, sugiriendo la necesidad de ajustes adicionales en el modelo para mejorar su precisión en un entorno real.

**Palabras clave**: Ciberseguridad · Ciencia de Datos · Aprendizaje Automático · Denegación de Servicio · Clasificación

# Índice de contenidos

# Índice de tablas

# Índice de figuras

# 1 Introducción

En este capítulo, se realiza una introducción, explicando el contexto, los objetivos del trabajo, la planificación seguida para el desarrollo del mismo y la estructura implementada.

## 1.1. Contexto y alcance

Hoy en día, la sociedad vive experimentando una gran evolución constante de la tecnología y de la información. Este gran desarrollo ha convertido a la ciberseguridad en un aspecto necesario, ya que, cada vez más, estos sistemas son más complejos y más difíciles de proteger. Es por ello, que aunque la idea de lograr una seguridad total sea un objetivo inalcanzable, persiste el desafío de seguir innovando en el campo de ciberseguridad [2].

Debido a la evolución de la tecnología comentada anteriormente, aparecen nuevas amenazas con las cuales los cibercriminales pretenden cumplir diversos objetivos, ya sean económicos, reputacionales, ideológicos, entre otros. Algunas de las amenazas más frecuentes en la actualidad son los ataques de Denegación de Servicio, los ataques por *Ransomware*, la explotación de vulnerabilidades web, las campañas de phishing, etc. Estos incidentes de seguridad representan un problema a tener en cuenta desde el punto de vista de las empresas y organizaciones, ya que la materialización de estas amenazas, supondría pérdidas económicas y reputacionales, en función de la criticidad de los activos afectados y del tiempo de inactividad hasta la recuperación de los servicios mínimos.

Para contrarrestar este tipo de ataques, existen distintas soluciones actual-

mente que sirven como medidas de prevención y de corrección para las organizaciones. Algunas de estas contramedidas o recomendaciones son el uso de cortafuegos, WAF (*Web Application Firewall*), balanceadores de carga o la instalación de un parque de servidores para aumentar los recursos, entre otros [3]. Asimismo, la Ciencia de Datos se presenta también como una solución efectiva basada en el aprendizaje de datos históricos para predecir nuevos futuros ataques y detectarlos a tiempo. La Ciencia de Datos es la disciplina encargada de analizar los datos y extraer conocimiento de estos para dar respuesta a los distintos problemas de la sociedad [4]. En la sección 2.2 del documento, se detallará con mayor profundidad qué es la Ciencia de Datos y cómo se podría aplicar para resolver estos tipos de problemas.

Este trabajo se centra en el desarrollo de un modelo de aprendizaje automático siguiendo la metodología de la Ciencia de Datos con el propósito de detectar ataques de Denegación de Servicio que utilicen el protocolo HTTP (*HyperText Transfer Protocol*) mediante el análisis de los registros del tráfico entrante en un servidor web. Para lograr esto, se seguirá el ciclo estándar de un proyecto de Ciencia de Datos (véase la 2.2b en la sección 2.2) mediante el cual se identificarán los problemas, se definirán los objetivos del modelo, se realiza una preparación de los datos, a continuación se acometerá la construcción del modelo y se finalizará con su visualización y evaluación en una simulación de un caso real para observar su puesta en producción.

## 1.2. Identificación del problema

Tal y como se ha mencionado en la sección anterior, existen diversas amenazas utilizadas por los atacantes para lograr sus objetivos. Dentro de estas amenazas, una de las más utilizadas son los ataques de Denegación de Servicio (DoS, *Denial of Service* por sus siglas en inglés). Estos ataques son realizados con el objetivo de inhabilitar el acceso a un servidor o red de servidores mediante el envío de un gran volumen de tráfico o por medio de paquetes de tráfico maliciosos, causando así la interrupción y la inaccesibilidad de estos servicios a los usuarios legítimos [5].

Existen diferentes tipos de ataques de Denegación de servicio en función del protocolo de red utilizado para colapsar una red o servidor. Uno de los más comunes son los ataques mediante el protocolo HTTP en los cuales uno o varios clientes envían solicitudes para acceder a los recursos de un servidor web mediante paquetes alterados con el objetivo de mantener en el tiempo las sesiones abiertas y así inhabilitar la creación de nuevas sesiones de usuarios legítimos nuevos debido a la sobrecarga del servidor web. Para lograr este tipo de ataque, los ciberdelincuentes emplean distintas herramientas, entre las cuales destacan *Golden Eye* y *Slowloris*.

En este caso, mediante el uso del conocimiento y de los patrones extraídos de los datos por medio de la Ciencia de Datos, es posible monitorizar, detectar y mitigar de forma proactiva en tiempo real la actividad anómalas de las redes, evitando así la aparición de estos ataques de Denegación de Servicio.

## 1.3. Definición de objetivos

El objetivo principal de este proyecto es la creación de un modelo de aprendizaje automático que logre la detección de ataques de Denegación de Servicio utilizando el protocolo HTTP, específicamente mediante el uso las herramientas *Golden Eye* o *Slowloris* en tiempo real siguiendo una metodología de Ciencia de Datos. Los objetivos clave del proyecto se pueden listar de la siguiente forma:

- **Recolección de los datos:** Es fundamental recopilar los datos necesarios a partir de los cuales desarrollar un modelo adecuado para poder detectar con la mayor precisión los ataques de Denegación de Servicio.
- **Conocimiento de los datos:** Mediante el análisis de los datos es importante conocer el significado de las variables y su naturaleza, es decir, cómo estas son generadas, qué valor aportan y cómo se relacionan con otras variables del conjunto de datos. Este objetivo es fundamental para comprender mejor los datos con los que se va a trabajar.
- **Desarrollo de un modelo de predicción:** Es importante crear un modelo de clasificación que sea capaz de predecir de manera satisfactoria el tipo de tráfico que se recibe.
- **Detección en tiempo real:** El modelo debe ser capaz de analizar el tráfico entrante y saliente de un servidor en tiempo real, utilizando para ello los registros generados.
- **Rapidez en el entrenamiento y predicción:** Se busca que el modelo tenga una alta velocidad tanto en el proceso de entrenamiento como en la predicción de los datos, ya que debe ser aplicable en situaciones de tiempo real.
- **Eficiencia en el uso de recursos:** Este modelo debe consumir la menor cantidad de recursos posibles. Esto garantiza que su funcionamiento no interfiera con las operaciones normales del sistema en el que se implementa, manteniendo la estabilidad y la eficiencia del mismo.
- **Precisión:** El modelo debe ser capaz de diferenciar con la mayor exactitud posible entre el tráfico legítimo y el malicioso. Esta capacidad es importante para minimizar los falsos positivos y los falsos negativos, mejorando así la confiabilidad y la utilidad del modelo.

## 1.4. Estructura del documento

A continuación, se presenta la estructura que seguirá el presente trabajo de fin de grado:

1. **Introducción**: Es el presente capítulo. En él se presenta el problema a abordar en este trabajo, una introducción a su contexto y la definición de los objetivos.

2. **Estado del arte**: Se exponen los marcos conceptuales básicos empleados en el trabajo y se explican los conceptos básicos de la metodología empleada.

3. **Datos y su preprocesado**: En este capítulo se describe el *software* utilizado durante el desarrollo del modelo de aprendizaje automático, así como se describe la recolección y el preprocesamiento de los datos.

4. **Análisis Exploratorio de los Datos**: Se llevan a cabo distintos tipos de análisis de datos para extraer sus características y patrones como parte de un análisis exploratorio de los datos.

5. **Modelado**: Se lleva a cabo el entrenamiento de distintos modelos de aprendizaje automático y su validación para la detección de ataques de Denegación de Servicio.

6. **Nuevos Datos**: En este capítulo se simula un escenario real con nuevos datos no vistos durante la etapa de entrenamiento y validación con el objetivo de evaluar el modelo.

7. **Lecciones aprendidas**: Se reflexiona sobre las decisiones tomadas durante el desarrollo del trabajo y las lecciones aprendidas. Esta sección proporciona una perspectiva crítica que puede ser útil para establecer las líneas de trabajos futuros.

8. **Conclusiones y trabajo a futuro**: Se repasan todas las acciones llevadas a cabo durante el trabajo, sintetizando los principales problemas y hallazgos, y las implicaciones de los resultados obtenidos. Además, se establecen distintas líneas de trabajo para desarrollar en el futuro.

9. **Bibliografía**: Se recogen las citas bibliográficas referenciadas a lo largo del trabajo. Se listan las fuentes consultadas, incluyendo libros, artículos académicos y otros recursos relevantes.

10. **Apéndice**: Se expone el código con el que se ha realizado el trabajo ofreciendo la posibilidad de replicar y verificar los resultados obtenidos.

## 1.5. Línea de tiempo

A continuación, se presenta el proceso de desarrollo del presente trabajo a partir de una línea de tiempo indicando la duración de cada una de las fases y tareas realizadas en el mismo.

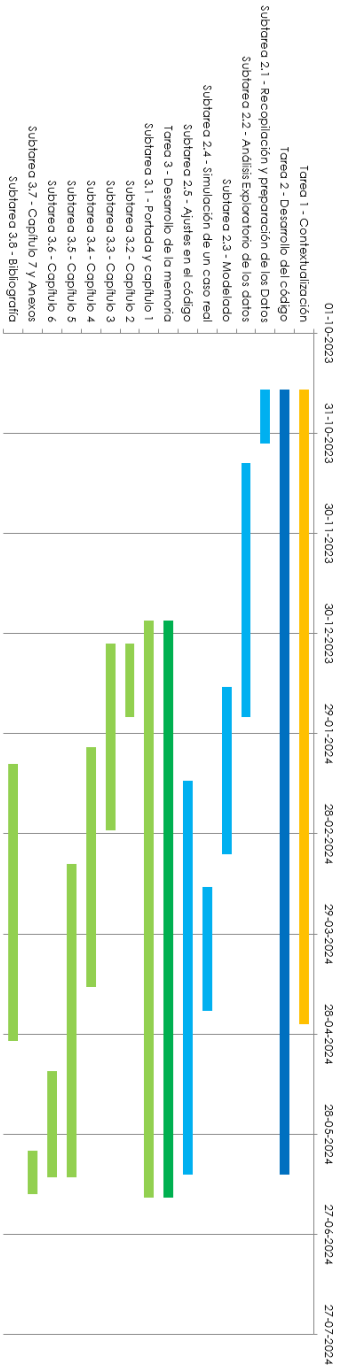



Figura 1.1: Línea de tiempo

# 2

# Estado del arte

En este capítulo se abordan los ataques de Denegación de Servicio, destacando su importancia en el contexto de la ciberseguridad y de dos herramientas comunes para realizarlos, *Golden Eye* y *Slowloris*. Adicionalmente, se introducirá la Ciencia de Datos, detallando su ciclo de vida para la identificación y mitigación de estos ataques, debido a su capacidad para analizar conjuntos de datos y descubrir patrones significativos.

Del mismo modo, se presentan las metodologías, las herramientas y el software utilizados en este trabajo. La selección de estos aspectos es importante para el análisis eficaz de los datos, permitiendo una detección precisa de los ataques y generando resultados relevantes para la ciberseguridad.

## 2.1. Ataques de Denegación de Servicio

Un ataque de Denegación de Servicio es una acción maliciosa que tiene como objetivo comprometer la disponibilidad de forma parcial o total de un sistema o sistemas, impidiendo el acceso a usuarios legítimos. Por lo general, este propósito se logra mediante el agotamiento de los recursos del sistema por medio de una alta volumetría de tráfico o fuerza bruta [5] [6]. Estos ataques pueden estar dirigidos a diferentes sistemas, entre los cuales destacan los servidores web. Estos pueden ser comprometidos mediante este tipo de ataques a partir de distintos protocolos como por ejemplo HTTP, TCP, FTP, entre otros. Uno de los más frecuentes es el uso del protocolo HTTP para agotar los recursos de un servidor web.

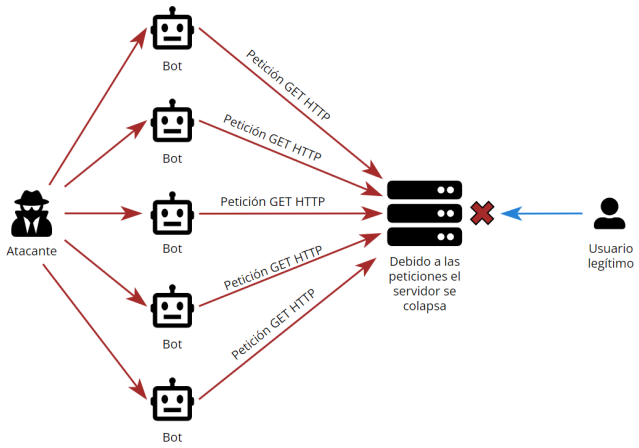


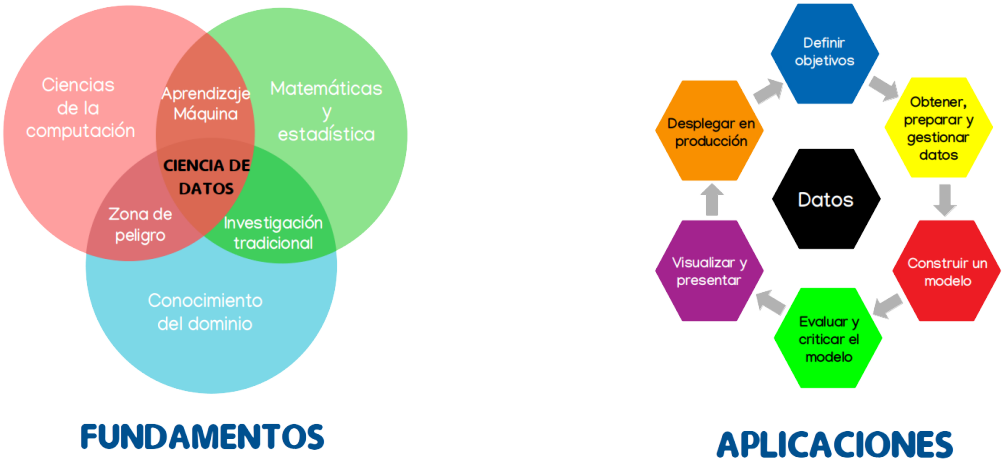

Figura 2.1: Ataque de Denegación de Servicio por el protocolo HTTP.

Un ataque DoS mediante el protocolo HTTP consiste en el envío de múltiples solicitudes GET a un servidor web desde distintos sistemas controlados por un atacante con el objetivo de mantener estas sesiones abiertas y denegar el servicio a los usuarios legítimos. En la figura 2.1 se puede observar una representación de este tipo de ataque.

Existen herramientas específicas diseñadas para ejecutar o simular ataques de Denegación de Servicio por el protocolo HTTP para fines de prueba de seguridad o para llevar a cabo ataques reales. Entre estas herramientas se encuentran *Slowloris* y *GoldenEye*.

*Slowloris*, es una herramienta de ataque de DoS que funciona mediante la apertura de múltiples conexiones al sistema objetivo y manteniéndolas abiertas el mayor tiempo posible. Esto lo hace mediante el envío de cabeceras HTTP parciales, que dejan abiertas las conexiones, y periódicamente enviando cabeceras adicionales para mantenerlas. Esto agota los recursos del servidor al mantener ocupados los hilos que gestionan las conexiones, impidiendo que el servidor atienda usuarios legítimos. Esta herramienta es principalmente efectiva contra servidores web, ya que utilizan el protocolo HTTP en su operativa habitual [7].

Por otra parte, *GoldenEye* es otra herramienta de ataque, inspirada en el ataque HTTP DoS, pero con capacidades ampliadas. Utiliza un enfoque de fuerza bruta tanto en la capa de aplicación como en la capa de conexión del modelo OSI, generando una gran cantidad de solicitudes HTTP para agotar los recursos del servidor objetivo. Del mismo modo, esta herramienta es utilizada para realizar pruebas de estrés en los servidores web, identificando vulnerabilidades en la gestión de conexiones y la capacidad de respuesta bajo cargas extremas [8].




(a) Diagrama de Venn de la Ciencia de datos.

(b) Ciclo de vida de un proyecto de Ciencia de Datos.

Figura 2.2: Diagrama de Venn y ciclo de un proyecto de Ciencia de Datos (figuras extraídas de [1]).

## 2.2. Ciencia de Datos

Para resolver el problema de la detección de los ataques de Denegación de Servicio, la Ciencia de Datos puede ser una solución factible. La Ciencia de Datos es la combinación de tecnología, matemáticas y estadística, y el conocimiento del dominio (figura 2.2a) para convertir los datos en información valiosa y útil para la toma de decisiones en una amplia variedad de disciplinas. En un mundo cada vez más digital, los datos se generan rápidamente en muchas disciplinas. La Ciencia de Datos permite a las empresas y organización transformar estos datos en información, que a su vez se transforma en conocimiento, para mejorar su rendimiento y tomar decisiones [4].

El ciclo de vida de un proyecto de Ciencia de Datos generalmente incluye los pasos de definición de objetivos, recopilación de los datos, limpieza de los datos, análisis, construcción de modelos, presentación de los resultados e integración del modelo [1] (véase la figura 2.2b).

La estructura de este trabajo está basada en el ciclo de vida de la Ciencia de Datos descrito anteriormente, empezando por la definición de los objetivos y por la recopilación de los datos necesarios para dar respuesta a las preguntas planteadas. A continuación, se realiza un preprocesado de los datos mediante un estudio inicial para que estén preparados para el modelo. Una vez preprocesados, se analizan en profundidad las distribuciones y las relaciones de las variables, para extraer la máxima información de los datos y lograr un mayor entendimiento.

Seguidamente, se realiza el ajuste del modelo de aprendizaje automático, haciendo las validaciones necesarias, para que, finalmente, se efectúe una simulación del despliegue en producción con el objetivo de evaluar el modelo con nuevos datos.

## 2.3. Metodología empleada

En la siguiente sección se explicarán las técnicas matemáticas y estadísticas utilizadas durante el presente proyecto de Ciencia de Datos para abordar el problema planteado de la detección de ataques de Denegación de Servicio.

### 2.3.1. Análisis Exploratorio de los Datos

Siguiendo con el ciclo de vida de la Ciencia de Datos, una vez definidos los objetivos, el proyecto continúa con la recopilación de datos. Este paso tiene como finalidad identificar y reunir los datos necesarios para disponer de suficiente información para el análisis. Esta recopilación se puede realizar de distintas maneras dependiendo del ámbito en el que se trabaje. Tras obtener estos datos, se procede con la limpieza de estos. Esta fase es realizada con el objetivo de aumentar la precisión de los datos y la utilidad de estos mediante la eliminación de errores, valores faltantes, duplicados y el filtrado de las muestras que puedan ser útiles a priori.

El siguiente paso consiste en el análisis de los datos. En concreto, en este proyecto se realizará un extenso Análisis Exploratorio de Datos (EDA, por sus siglas en inglés). Este análisis es un enfoque que busca comprender y descubrir patrones, anomalías y relaciones clave en un conjunto de datos que no se puedan apreciar a simple vista. Este proceso comienza con la exploración de las distribuciones de las variables individuales con el objetivo de entender la naturaleza y la forma de los datos [9] [10]. En el ámbito de los proyectos de Ciencia de Datos, la etapa de preparación de los datos representa, habitualmente, el 80 % del tiempo invertido. Dentro de esta fase, el EDA es un componente fundamental.

Una parte del EDA consiste en la identificación de patrones y anomalías. Se buscan valores atípicos que difieran significativamente del resto de datos y se examina cualquier indicio de sesgo o irregularidad. Estos hallazgos son importantes para aumentar la calidad y fiabilidad del conjunto de datos. Esta búsqueda se realiza mediante la observación de las distintas variables a través de gráficos como *boxplot* o diagramas de barras en función del tipo de variable que se quiera analizar.

El análisis multivariante en el que se estudia cómo interactúan las diferentes variables es otro componente importante del análisis exploratorio. Mediante

contraste de hipótesis, se exploran las relaciones entre las variables, lo que es importante para entender la dinámica del conjunto de datos y para la modelización futura.

Adicionalmente, una técnica importante en la selección de variables dentro del proceso de EDA es el análisis del Factor de Infracción de la Varianza (VIF, por sus siglas en inglés). Este método será empleado para identificar y cuantificar el grado de multicolinealidad entre las variables explicativas y, posteriormente, seleccionar aquellas variables que presenten un mayor grado de multicolinealidad para su eliminación [11].

Gracias a estas técnicas, se logrará una compresión de los datos que permitirá una adaptación de los datos a las especificaciones del modelo con el propósito de obtener un sistema capaz de predecir los ataques de Denegación de Servicio de forma eficiente y eficaz.

### 2.3.2. Árbol de decisión

El siguiente paso del análisis de los datos es la construcción de un modelo por medio de la aplicación de algoritmos o técnicas estadísticas. En la Ciencia de Datos existen múltiples modelos que se ajustan a distintas necesidades en base de los datos proporcionados y del objetivo. Por una parte, es necesario diferenciar si se va a realizar un aprendizaje supervisado o no supervisado. Esto depende si los datos recopilados tienen las categorías objetivo ya establecidas o no, respectivamente. A partir de este punto, se abre un abanico de posibilidades entre los cuales se encuentra los modelos de clasificación de datos.

Dentro de los modelos de clasificación, los árboles de decisión son unos de los algoritmos más frecuentes y sencillos de aplicar. Se utilizan por su facilidad de interpretación, versatilidad y rendimiento. Un árbol de decisión es de una estructura similar a la de un árbol formado por nodos, ramas y hojas. Cada nodo representa un atributo, cada rama una regla de decisión y cada hoja representa un resultado.

El funcionamiento de un árbol de decisión parte del nodo raíz donde se toma la decisión inicial basada en la característica más decisiva de los datos. La elección de la característica más decisiva se suele realizar mediante la comparación de los coeficientes de Gini de cada atributo. Esta decisión crea ramas que conducen a otros nodos, donde se toman más decisiones, y así sucesivamente, hasta que se alcanza el nodo hoja que representa el resultado de la predicción [12].

Los árboles de decisión ofrecen varias ventajas. Destacan por su interpretación intuitiva, ya que son fáciles de entender e interpretar porque las decisiones tomadas se parecen a las que un humano tomaría. Además, pueden manejar datos tanto categóricos como numéricos. También son útiles para identificar las carac-

terísticas más importantes, ya que pueden proporcionar una visión intuitiva de los datos.

Sin embargo, los árboles de decisión también tienen algunas desventajas. Una de las más notables es el sobreajuste. Es posible crear árboles muy complejos que no generalicen bien los datos, lo que puede conducir en un rendimiento inferior en datos no vistos. Este problema puede ser mitigado mediante la poda del árbol o la configuración de límites en la profundidad máxima. Los árboles también pueden ser sensibles a pequeñas variaciones en los datos, lo que da como resultado árboles muy diferentes y posiblemente incoherentes.

### 2.3.3. Bosque aleatorio

Un bosque aleatorio o *random forest* es otro modelo de aprendizaje automático, frecuentemente utilizado, que opera construyendo múltiples árboles de decisión durante el entrenamiento. Esencialmente, es un conjunto de árboles de decisión, donde cada árbol se construye a partir de una muestra aleatoria del conjunto de datos de entrenamiento. En la creación de estos árboles, se selecciona una subdivisión aleatoria de características en cada paso de división, lo que contribuye a la diversidad de los modelos y a la robustez del algoritmo [1].

El funcionamiento de un bosque aleatorio implica que cada árbol en el bosque, hace una predicción y el resultado final se obtiene ya sea por mayoría de votos, en problemas de clasificación, o promediando las predicciones, en problemas de regresión. Esta metodología de agregación reduce el riesgo de sobreajuste haciendo que este modelo sea más robusto y preciso.

Entre las ventajas del *random forest*, destaca su capacidad para manejar grandes conjuntos de datos con una dimensionalidad alta (muchas características). Es eficaz para clasificación y regresión y no requiere de una gran cantidad de preprocesamiento de los datos. Además, es menos propenso al sobreajuste que los árboles de decisión individuales y proporciona indicadores de la importancia de las características, lo cual es útil para la selección de características.

Sin embargo, un bosque aleatorio también tiene desventajas. Puede ser computacionalmente intensivo, lo que lo hace menos ideal para aplicaciones en tiempo real. La interpretabilidad de este método es menor en comparación con un árbol de decisión individual, ya que la decisión final es el resultado de múltiples árboles. Además, en casos de datos muy desequilibrados, el modelo puede estar sesgado hacia la clase dominante. Finalmente, aunque es menos propenso al sobreajuste, esto puede seguir siendo un problema si el número de árboles es demasiado grande.

# 3

# Datos y su preprocesado

Una vez explicadas las metodologías que se van a emplear en este trabajo, es necesario comenzar la etapa de recolección de los datos y de su preprocesamiento para adecuarlos al modelo.

Este capítulo aborda distintos aspectos necesarios para la creación de un modelo de aprendizaje automático. Estos aspectos consisten en la manera de obtención de los datos con los que se va a entrenar y validar el modelo, no sin antes, realizando un preprocesado en el cual se filtrarán y limpiarán estos datos para conseguir un conjunto de datos con solo las muestras y las características acordes.

Asimismo, el código utilizado para la elaboración de este capítulo puede encontrarse en las secciones A.1, A.2 y A.3 del apéndice A.

## 3.1. *Software* empleado

Existen diferentes entornos para la creación y utilización de modelos de Ciencia de Datos. Sin embargo, para el desarrollo del sistema de análisis de tráfico del presente proyecto se ha elegido Jupyter Notebook [13] como entorno de desarrollo por varios motivos. En primer lugar, Jupyter Notebook proporciona un entorno donde se permite combinar código, texto explicativo y visualizaciones en un solo documento. Esta característica es especialmente útil para el análisis de registros porque es importante visualizar los resultados, describirlos y explicar los métodos utilizados.

Además, este documento tiene la capacidad de ejecutar código de forma interactiva y en bloques. Esto permite realizar pruebas y experimentos de forma rápida y sencilla, ya que se pueden ejecutar y depurar porciones de código de forma independiente.

Finalmente, otra ventaja de Jupyter Notebook es que es compatible con varios lenguajes de programación, incluyendo Python, R y Julia, lo que brinda flexibilidad a la hora de elegir el lenguaje más apropiado para el análisis de registros. Para acometer el desarrollo de un sistema para la detección de ataques de Denegación de Servicio, se trabajará con el lenguaje Python [14].

El uso de Python en el ámbito del aprendizaje supervisado y específicamente para analizar y clasificar ataques de Denegación de Servicio en registros de tráfico tiene importantes ventajas. Una de las ventajas más importantes de Python es su simplicidad y legibilidad, lo que facilita el mantenimiento y la revisión del código. Otra ventaja es que Python es un lenguaje de propósito general con una gran comunidad de usuarios, lo que significa que se pueden encontrar fácilmente soluciones a problemas comunes y hay una gran cantidad de documentación disponible.

## 3.2. Fuente de los datos

Para realizar este trabajo, se ha contado con un conjunto de datos volumétrico y representativo. Este conjunto de datos reflejar las características y patrones de tráfico que se observan en situaciones reales, permitiendo así entrenar y validar un modelo de aprendizaje automático con precisión. En este contexto, se utilizará un conjunto de datos que contiene los logs de los paquetes de tráfico dirigidos a los servidores de la Universidad de New Brunswick (Canadá) entre febrero y marzo de 2018. Este conjunto de datos ha sido descargado desde el sitio web Kaggle [15] [16].

En particular, este conjunto de datos se divide en varios archivos, dividido por fechas. Asimismo, cada archivo recoge los registros de distintos tipos de ataques, por lo que se utilizarán solo las muestras recogidas en el fichero llamado “02-15-2018.csv”, ya que es el único donde se recogen los ataques realizados por las herramientas *Golden Eye* y *Slowloris*.

La información contenida en este conjunto de datos permite el análisis de patrones de tráfico y facilita la identificación de los ataques de Denegación de Servicio por medio del protocolo HTTP mediante el uso de *Golde Eye* y *Slowloris*. Este conjunto de datos proporciona información sobre la fecha y hora de los eventos, las direcciones IP tanto de origen como de destino, los protocolos de red utilizados, los puertos asociados, entre otros datos relevantes. Debido a esto, este conjunto permite la exploración y compresión de las características de este tipo

de ataques, lo que permite en el desarrollo favorable de un modelo de aprendizaje automático.

### 3.2.1. Datos

Este trabajo emplea un conjunto de datos disponible en el archivo "02-15-2018.csv", que se encuentra en una ruta personalizada del dispositivo donde se realiza este. Es importante mencionar que la ruta es específica para el sistema en el que se está realizando el proyecto. Si se necesita replicar en otro sistema, la ruta del archivo puede requerir modificaciones para reflejar la estructura del sistema de archivos de ese sistema en particular.

Este conjunto de datos proporciona los registros necesarios para el análisis de los ataques de Denegación de Servicio dirigidos al servidor web de la Universidad de New Brunswick.

Para la recopilación de los datos en el entorno de desarrollo, ha sido necesario cargar el archivo CSV (*Comma-Separated Value*) que los contenía en un *DataFrame* de la librería *Pandas*. Este tipo de variable es una estructura bidimensional, de tamaño modificable y heterogénea, con filas y columnas etiquetadas. Esta estructura es bastante útil para el análisis de datos debido a su facilidad y versatilidad.

Tras la carga inicial de los datos, se ha realizado un análisis preliminar del conjunto de datos para obtener información sobre su estructura. Previamente a esto y como parte de este proyecto, se ha decidido dividir el conjunto de datos en dos subconjuntos, siguiendo una proporción de 90/10. Esto significa que el 90 % de los datos se utilizarán durante todo el proceso de desarrollo y entrenamiento del modelo, mientras que el 10 % restante se reservará para simular un caso real, a posteriori, en el que se reciben nuevos datos de forma secuencial.

Al aplicar esta división, el subconjunto principal que se usará en la mayoría de etapas del ciclo de vida de Ciencia de Datos consta de 943.717 filas y 80 columnas. Esto indica la presencia de un volumen significativo de muestras, cada una con 80 variables distintas, proporcionando una base sólida para trabajar.

Además, con el objetivo de obtener un conjunto desordenado de datos y una visión general de estos, se ha realizado una aleatorización de los datos. Esta aleatorización es realizada debido a que el conjunto de datos obtenido se encuentra ordenado por etiqueta, lo que conlleva que todas las muestras legítimas y maliciosas estén divididas. Este procedimiento es útil para entender mejor la naturaleza de los datos y sus variables. A continuación, en la figura 3.1 se puede observar una selección de cinco registros al azar.

Mediante la observación de la figura 3.2, se ha llevado a cabo un análisis de las etiquetas existentes en el conjunto de datos. En este caso, se han identifica-

```
datos = datos.sample(frac=1, random_state = 12)
datos.head(5)
```

| | Dst Port | Protocol | Timestamp | Flow Duration | Tot Fwd Pkts | Tot Bwd Pkts | TotLen Fwd Pkts | TotLen Bwd Pkts | Fwd Pkt Len Max | Fwd Pkt Len Min | ... | Fwd Seg Size Min | Active Mean | Active Std | Active Max | Active Min | Idle Mean | Idle Std | Idle Max | Idle Min | Label |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **236130** | 53 | 17 | 15/02/2018 12:08:49 | 12441 | 1 | 1 | 40 | 101 | 40 | 40 | ... | 8 | 0.0 | 0.0 | 0 | 0 | 0.0 | 0.0 | 0 | 0 | Benign |
| **716473** | 80 | 6 | 15/02/2018 10:41:00 | 191 | 2 | 0 | 0 | 0 | 0 | 0 | ... | 20 | 0.0 | 0.0 | 0 | 0 | 0.0 | 0.0 | 0 | 0 | Benign |
| **203850** | 80 | 6 | 15/02/2018 10:20:13 | 54743301 | 2 | 0 | 0 | 0 | 0 | 0 | ... | 20 | 0.0 | 0.0 | 0 | 0 | 0.0 | 0.0 | 0 | 0 | Benign |
| **88225** | 445 | 6 | 15/02/2018 04:11:09 | 126229 | 3 | 1 | 0 | 0 | 0 | 0 | ... | 20 | 0.0 | 0.0 | 0 | 0 | 0.0 | 0.0 | 0 | 0 | Benign |
| **1020415** | 53 | 17 | 15/02/2018 11:46:16 | 280 | 1 | 1 | 35 | 99 | 35 | 35 | ... | 8 | 0.0 | 0.0 | 0 | 0 | 0.0 | 0.0 | 0 | 0 | Benign |

5 rows × 80 columns

Figura 3.1: Extracto del conjunto de datos.

do las etiquetas "Benign", "DoS attacks-GoldenEye" y "DoS attacks-Slowloris". La presencia de estas etiquetas indica que el conjunto de datos no solo incluye tráfico normal ("Benign"), sino también registros de los dos tipos de ataques de Denegación de Servicio que se pretenden abordar en el presente trabajo.

La existencia de registros de estos ataques, junto con la gran cantidad de datos disponibles, sugiere que el conjunto de datos tiene un gran potencial para el entrenamiento de un modelo de aprendizaje automático supervisado que pueda detectar eficazmente los ataques de Denegación de Servicio.

Adicionalmente, se ha observado que de los 943.717 registros disponibles, 896.469 son muestras legítimas, 37.357 corresponden a muestras de tráfico creado mediante la herramienta *Golden Eye* y 9.891 datos corresponde a paquetes de tráfico originado por la herramienta *Slowloris*.

Posteriormente, se calcula el porcentaje de cada tipo de tráfico con respecto al total de registros. Los resultados muestran que aproximadamente el 95 % es tráfico legítimo, el 4 % corresponde a "DoS attacks-GoldenEye" y el 1 % a "DoS attacks-Slowloris". Como se puede observar, el problema que se trata en este trabajo consiste en un problema de clasificación notablemente desbalanceado, donde las etiquetas de interés, "DoS attacks-GoldenEye" y "DoS attacks-Slowloris", ocurren en un porcentaje muy bajo de los casos.

Asimismo, se ha generado un gráfico de barras para visualizar el número de registros de cada categoría (véase figura 3.2). En este gráfico, cada tipo de tráfico se representa con un color distintivo (verde para la categoría "Benign", rojo para "DoS attacks-GoldenEye" y naranja para "DoS attacks-Slowloris"). Este gráfico permite una comprensión clara y rápida de la distribución de los registros.

Por lo tanto, uno de los objetivos que tendrá este modelo de aprendizaje automático será la capacidad de predecir el tipo de tráfico entrante al servidor mediante la diferenciación entre tráfico legítimo y tráfico malicioso.

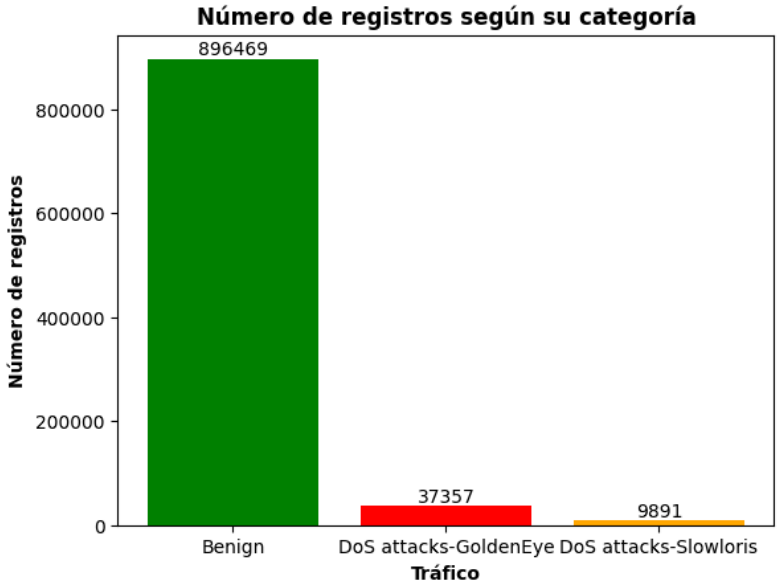



Figura 3.2: Gráfico de barras del número de muestras por etiquetas de tráfico.

### 3.2.2. Descripción de las variables

Para poder comprender el conjunto de datos recopilado, es necesario conocer cada una de las variables que componen cada muestra. A continuación, se detalla una tabla con cada una de las características del conjunto de datos, así como una breve descripción y el tipo de variable que es [16].

- **Dst Port**: Puerto de destino. Variable categórica.
- **Protocol**: Protocolo. Variable categórica.
- **Timestamp**: Marca de tiempo. Variable numérica.
- **Flow Duration**: Duración del flujo en microsegundos. Variable numérica.
- **Tot Fwd Pkts**: Número total de paquetes entrantes. Variable numérica.
- **Tot Bwd Pkts**: Número total de paquetes salientes. Variable numérica.
- **TotLen Fwd Pkts**: Tamaño total del paquete entrante. Variable numérica.
- **TotLen Bwd Pkts**: Tamaño total del paquete saliente. Variable numérica.
- **Fwd Pkt Len Max**: Longitud máxima del paquete entrante. Variable numérica.
- **Fwd Pkt Len Min**: Longitud mínima del paquete entrante. Variable numérica.

- **Fwd Pkt Len Mean**: Longitud media del paquete entrante. Variable numérica.
- **Fwd Pkt Len Std**: Desviación estándar de la longitud del paquete entrante. Variable numérica.
- **Fwd Bwd Len Max**: Longitud máxima del paquete saliente. Variable numérica.
- **Fwd Bwd Len Min**: Longitud mínima del paquete saliente. Variable numérica.
- **Fwd Bwd Len Mean**: Longitud media del paquete saliente. Variable numérica.
- **Fwd Bwd Len Std**: Desviación estándar de la longitud del paquete saliente. Variable numérica.
- **Flow Byts/s**: Número de bytes de flujo por segundo. Variable numérica.
- **Flow Pkts/s**: Número de paquetes de flujo por segundo. Variable numérica.
- **Flow IAT Mean**: Tiempo medio entre dos paquetes enviados en el flujo. Variable numérica.
- **Flow IAT Std**: Desviación estándar del tiempo entre dos paquetes enviados en el flujo. Variable numérica.
- **Flow IAT Max**: Tiempo máximo entre dos paquetes enviados en el flujo. Variable numérica.
- **Flow IAT Min**: Tiempo mínimo entre dos paquetes enviados en el flujo. Variable numérica.
- **Fwd IAT Tot**: Tiempo total entre dos paquetes entrantes enviados. Variable numérica.
- **Fwd IAT Mean**: Tiempo medio entre dos paquetes entrantes enviados. Variable numérica.
- **Fwd IAT Std**: Desviación estándar en el tiempo entre dos paquetes entrantes enviados. Variable numérica.
- **Fwd IAT Max**: Tiempo máximo entre dos paquetes entrantes enviados. Variable numérica.
- **Fwd IAT Min**: Tiempo mínimo entre dos paquetes entrantes enviados. Variable numérica.

- **Bwd IAT Tot**: Tiempo total entre dos paquetes salientes enviados. Variable numérica.
- **Bwd IAT Mean**: Tiempo medio entre dos paquetes salientes enviados. Variable numérica.
- **Bwd IAT Std**: Desviación estándar en el tiempo entre dos paquetes salientes enviados. Variable numérica.
- **Bwd IAT Max**: Tiempo máximo entre dos paquetes salientes enviados. Variable numérica.
- **Bwd IAT Min**: Tiempo mínimo entre dos paquetes salientes enviados. Variable numérica.
- **Fwd PSH Flag**: Número de veces que se activó la flag PSH en paquetes que viajaban en la dirección de entrada. Variable numérica.
- **Bwd PSH Flag**: Número de veces que se activó la flag PSH en paquetes que viajaban en la dirección de salida. Variable numérica.
- **Fwd URG Flag**: Número de veces que se activó la flag URG en paquetes que viajaban en la dirección de entrada. Variable numérica.
- **Bwd URG Flag**: Número de veces que se activó la flag URG en paquetes que viajaban en la dirección de salida. Variable numérica.
- **Fwd Header Len**: Total de bytes usados por las cabeceras en la dirección de entrada. Variable numérica.
- **Bwd Header Len**: Total de bytes usados por las cabeceras en la dirección de salida. Variable numérica.
- **Fwd Pkts/s**: Número de paquetes entrantes enviados por segundo. Variable numérica.
- **Bwd Pkts/s**: Número de paquetes salientes enviados por segundo. Variable numérica.
- **Pkt Len Min**: Longitud mínima de un paquete. Variable numérica.
- **Pkt Len Max**: Longitud máxima de un paquete. Variable numérica.
- **Pkt Len Mean**: Longitud media de un paquete. Variable numérica.
- **Pkt Len Std**: Desviación estándar de la longitud de un paquete. Variable numérica.
- **Pkt Len Var**: Varianza de la longitud de un paquete. Variable numérica.

- **FIN Flag Cnt**: Número de paquetes con flag FIN. Variable numérica.
- **SYN Flag Cnt**: Número de paquetes con flag SYN. Variable numérica.
- **RST Flag Cnt**: Número de paquetes con flag RST. Variable numérica.
- **PSH Flag Cnt**: Número de paquetes con flag PSH. Variable numérica.
- **ACK Flag Cnt**: Número de paquetes con flag ACK. Variable numérica.
- **URG Flag Cnt**: Número de paquetes con flag URG. Variable numérica.
- **CWE Flag Cnt**: Número de paquetes con flag CWE. Variable numérica.
- **ECE Flag Cnt**: Número de paquetes con flag ECE. Variable numérica.
- **Down/Up Ratio**: Ratio de descargas y subidas. Variable numérica.
- **Pkt Size Avg**: Tamaño del paquete promedio. Variable numérica.
- **Fwd Seg Size Avg**: Tamaño promedio observado en la dirección entrante. Variable numérica.
- **Bwd Seg Size Avg**: Tamaño promedio observado en la dirección saliente. Variable numérica.
- **Fwd Byts/b Avg**: Número promedio de bytes a granel en la dirección entrante. Variable numérica.
- **Fwd Pkts/b Avg**: Número promedio de paquetes a granel en la dirección entrante. Variable numérica.
- **Fwd Blk Rate Avg**: Promedio de la tasa a granel en la dirección entrante. Variable numérica.
- **Bwd Byts/b Avg**: Número promedio de bytes a granel en la dirección saliente. Variable numérica.
- **Bwd Pkts/b Avg**: Número promedio de paquetes a granel en la dirección saliente. Variable numérica.
- **Bwd Blk Rate Avg**: Promedio de la tasa a granel en la dirección saliente. Variable numérica.
- **Subflow Fwd Pkts**: Número promedio de paquetes en un subflujo en dirección entrante. Variable numérica.
- **Subflow Fwd Byts**: Número promedio de bytes en un subflujo en dirección entrante. Variable numérica.

- **Subflow Bwd Pkts**: Número promedio de paquetes en un subflujo en dirección saliente. Variable numérica.
- **Subflow Bwd Byts**: Número promedio de bytes en un subflujo en dirección saliente. Variable numérica.
- **Init Fwd Win Byts**: Número total de bytes entrantes enviados en la ventana inicial. Variable numérica.
- **Init Bwd Win Byts**: Número total de bytes salientes enviados en la ventana inicial. Variable numérica.
- **Fwd Act Data Pkts**: Recuento de paquetes entrantes con al menos 1 byte de carga útil de datos TCP. Variable numérica.
- **Fwd Seg Size Min**: Tamaño mínimo del segmento entrante observado. Variable numérica.
- **Active Mean**: Tiempo medio que el flujo estuvo activo. Variable numérica.
- **Active Std**: Desviación estándar del tiempo que el flujo estuvo activo. Variable numérica.
- **Active Max**: Tiempo máximo que el flujo estuvo activo. Variable numérica.
- **Active Min**: Tiempo mínimo que el flujo estuvo activo. Variable numérica.
- **Idle Mean**: Tiempo medio de inactividad en el flujo. Variable numérica.
- **Idle Std**: Desviación estándar del tiempo de inactividad en el flujo. Variable numérica.
- **Idle Max**: Tiempo máximo de inactividad en el flujo. Variable numérica.
- **Idle Min**: Tiempo mínimo de inactividad en el flujo. Variable numérica.
- **Label**: Etiqueta. Variable categórica. Benign, DoS attacks-GoldenEye y DoS attacks-Slowloris.

## 3.3. Filtrado de datos

Tras la carga del conjunto de datos en el entorno de desarrollo, es necesario ajustar los datos a las necesidades y requisitos del problema, así como del modelo de aprendizaje automático para evitar sesgos y operaciones innecesarias. Entre estas modificaciones, es necesario que el conjunto de datos sea filtrado para descartar aquellas muestras y características que no contengan ninguna relevancia para abordar el presente problema.

El filtrado de los datos consiste en seleccionar un subconjunto específico de datos de un conjunto más amplio, basándose en criterios predefinidos *a priori*. Al eliminar datos irrelevantes o no pertenecientes para el estudio especificado, se simplifica el análisis y se mejora la precisión de los resultados obtenidos.

Por otro lado, este proceso ayuda a reducir el volumen de datos a analizar, lo cual es importante cuando se trabaja con grandes volúmenes de datos, como en este caso. Además, no solo permite ahorrar tiempo y recursos en el procesamiento, sino que también puede ser importante para descubrir patrones.

### 3.3.1. Eliminación de variables irrelevantes

Durante la fase de preprocesado de los datos, es posible identificar variables que no proporcionan información significativa para el desarrollo del modelo. Es por esto, que las columnas correspondientes a estas variables deben omitirse para garantizar que la interpretación de los datos sea más clara.

De forma anticipada, se ha podido identificar que la variable *Timestamp* no contiene información relevante para el desarrollo de un sistema de detección de ataques de Denegación de Servicio. Esta característica del conjunto de datos muestra la fecha y la hora en la que el paquete de tráfico fue registrado en el servidor. Por este motivo, no contiene información relevante para un problema de clasificación

Debido a esto, se ha decidido eliminar dicha variable de todas las muestras del conjunto de datos.

### 3.3.2. Eliminación de muestras irrelevantes

Dentro del proceso de filtrado de datos, una tarea específica e importante es la eliminación de muestras irrelevantes. Este proceso implica identificar y descartar aquellas partes del conjunto de datos que no contribuyen al análisis en cuestión.

Una prueba de ello puede ser la decisión de eliminar todos aquellos paquetes de las muestras que no se hayan realizado por el puerto asociado al protocolo HTTP, es decir, todas aquellas muestras que no tengan el valor 80 en la variable *Dst Port*. Esta medida es tomada debido a que el objetivo del trabajo es el de analizar el tráfico de red, específicamente del puerto 80, correspondiente al protocolo HTTP, debido a que los ataques a tratar solo se realizan en ese puerto de destino.

Al eliminar los datos de otros puertos, se reduce el ruido en el conjunto de datos, ya que son irrelevantes para el objetivo del análisis.

Tras el filtrado de los datos realizados en esta sección, el conjunto de datos

restante cuenta con 215.075 filas y 79 columnas. Esto supone una reducción del conjunto de datos en un 77,2 %.

## 3.4. Limpieza de datos

La limpieza de datos se centra en identificar y corregir errores e inconsistencias en estos. Esto puede incluir la gestión de valores faltantes o nulos, la corrección de datos duplicados y atípicos y la transformación de los datos para asegurar la coherencia, entre otros.

Esta etapa es bastante importante debido a que los datos en su estado original frecuentemente contienen errores o inconsistencias que pueden llevar a conclusiones erróneas o a un mal desempeño de los modelos de análisis y predicción.

### 3.4.1. Eliminación de muestras duplicadas

La eliminación de muestras duplicadas consiste en identificar y remover registros en un conjunto de datos que tienen exactamente la misma información en todas las columnas. Estos duplicados pueden surgir por errores durante la recolección, almacenamiento o procesamiento de los datos.

Mantener registros duplicados puede llevar a resultados engañosos en el análisis, ya que distorsionan la representatividad y la distribución de los datos. Por ejemplo, si un registro duplicado representa un caso anómalo, su repetición excesiva puede sesgar el análisis hacia ese caso en particular. Es esencial, por lo tanto, identificar y eliminar estos duplicados para asegurar la calidad y precisión del análisis siguiente. Al hacerlo, se obtiene un conjunto de datos más limpio y fiable, permitiendo una interpretación más exacta y válida de la información.

En el caso del conjunto de datos escogido para el modelo de aprendizaje automático, se han observado 40.310 muestras, las cuales se procederá a su eliminación para evitar sesgos.

### 3.4.2. Eliminación de variables constantes

Eliminar variables que tienen el mismo valor en todas las muestras de un conjunto de datos es una práctica recomendada en el análisis de datos. Estas variables, también conocidas como variables constantes, no agregan información de variación o diferencia al conjunto de datos.

Su presencia puede provocar ruido innecesario, aumentar la complejidad de los cálculos y complicar la interpretación de los resultados. Además, como no pro-

porcionan variación, no tienen poder discriminatorio o predictivo en los modelos estadísticos y de aprendizaje automático. Por lo tanto, eliminar estas variables mejorará la eficiencia del análisis y la calidad de la información extraída del conjunto de datos.

Tras comprobar que variables solo tienen un único valor en el conjunto de datos, se han identificado trece características constantes. Entre estas, se ha observado la variable *Dst Port* debido a que se han escogido solo aquellas muestras con el valor 80 en esa variable en la sección anterior de eliminación de muestras irrelevantes. El resto de variables constantes son: *Protocol, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Pkt Len Min, CWE Flag Count, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg* y *Bwd Blk Rate Avg*.

### 3.4.3. Eliminación de valores faltantes

Los valores nulos pueden surgir por diversas causas, tales como fallos en la recopilación de datos, omisiones durante la entrada de datos, o porque la información no se encuentra disponible.

```
datos = datos.replace(["Infinity", "infinity"], np.inf)
datos = datos.replace([np.inf, -np.inf], np.nan)

valores_faltantes = datos.isnull().sum()
valores_faltantes = valores_faltantes[valores_faltantes > 0]
print (valores_faltantes)
```

```
Series([], dtype: int64)
```

Figura 3.3: Valores faltantes.

Tal y como se muestra en la figura 3.3, no se han identificado valores faltantes en el conjunto de datos, por lo que no se eliminará ninguna muestra según este criterio.

### 3.4.4. Eliminación de valores erróneos

La eliminación de valores erróneos es un proceso crucial en la preparación de datos, ya que estos valores pueden distorsionar y afectar negativamente a los resultados del análisis. Estos valores son un caso particular de valores atípicos que surgen por equivocación por diversas razones como la medición, la entrada de los datos o la variabilidad natural. Los valores erróneos son valores no plausibles y, por tanto, han de ser eliminados.

Los valores erróneos pueden distorsionar y afectar negativamente a los resultados del análisis. Estos surgen como consecuencia de errores provocados por diversas razones como la medición, la entrada de los datos al sistema de recopilación o la variabilidad natural del dato. Los valores erróneos son no plausibles, ya que pueden distorsionar y afectar negativamente a los resultados del análisis y, por lo tanto, han de ser eliminados del conjunto.

En particular, la identificación de valores negativos dentro de las variables del conjunto es un aspecto importante debido a que este tipo de valores no puede ocurrir dentro de este conjunto de datos.

Tras haber identificado los valores negativos, se ha observado que la variable *Init Bwd Win Byts* presenta este tipo de error. Por lo tanto, se procederá al borrado de todas las muestras que contengan estos valores atípicos porque pueden causar que el resultado del modelo no se ajuste a la realidad.

Tras el proceso de eliminación de distintos tipos de muestras y de variables, el conjunto de datos actual consta de 128.247 filas y 66 columnas.

## 3.5. Transformación de la variable objetivo a binario

Transformar una variable destino con múltiples clases, como es el caso, en una variable binaria puede ser beneficioso. En el problema presentado, las clases *"DoS attackas-GoldenEye"* y *"DoS attacks-Slowloris"* no muestran diferencias entre sí que sean relevantes para el objetivo del análisis de datos, que es la detección de ataques de Denegación de Servicio. En este caso, sería más provechoso juntar ambas clases y crear la clase "Malicioso".

La binarización de la variable destino introduce simplicidad en el análisis. Al reducir el problema a una clasificación binaria, se facilita la interpretación y comunicación de los resultados. Además, esto mejora el rendimiento del modelo. Los modelos binarios tienden a ser más eficientes en cuanto a recursos y tiempo de entrenamiento en comparación con los modelos multiclase.

Del mismo modo, algunos métodos de aprendizaje automático están primordialmente diseñados para la clasificación binaria y pueden no ser tan eficaces para problemas multiclase o podrían necesitar ajustes adicionales.

Desde un punto de vista de la evaluación, la clasificación binaria simplifica el cálculo y la interpretación de métricas como la precisión, el *recall*, el *F1-score* y el *AUC-ROC*.

Debido a esto, se ha decidido transforma la variable *Label* para que solo contenga dos categorías: "Legítimo" y "Malicioso". Cabe puntualizar que se realiza

una traducción de la clase “Benign” por “Legítimo” para una mayor compresión.

Tras la binarización, se observa que el conjunto de datos contiene 96.362 muestras legítimas y 31885 muestras maliciosas.

Se ha identificado que la proporción de la muestra es de 0,75 para las muestras legítimas y 0,25 para las muestras maliciosas. Tras la binarización, se puede observar que persiste el problema de clasificación desbalanceada, ya que la clase de interés, “Malicioso”, tiene una proporción muy baja respecto a la clase “Legítimo”.

# 4

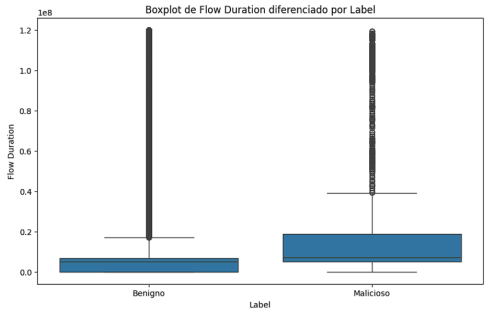
# Análisis Exploratorio de los Datos

En esta sección, se llevará a cabo un Análisis Exploratorio de los Datos o EDA, por sus siglas en inglés. El EDA es un proceso por el cual se examinan y se transforman los datos con el fin de garantizar su calidad, extraer sus características principales, identificar patrones, entre otras funciones.

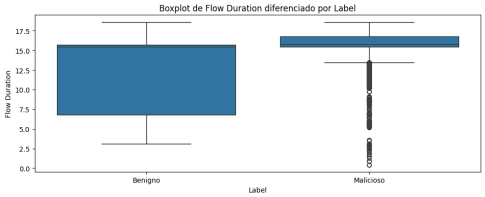
Este análisis es un paso importante en cualquier proyecto de Ciencia de Datos antes de proceder al modelado, ya que permite encontrar las variables más relevantes del conjunto de datos y permite estudiar su relación con las demás características. Además, en esta fase se realizan las transformaciones y los ajustes necesarios para que los datos estén en el formato adecuado y sean óptimos para su uso en los modelos del análisis [10].

Para comenzar este proceso, se realizarán distintas etapas con el objetivo de aumentar la calidad de las muestras y encontrar las variables más importantes para el modelo del proyecto. Estas etapas incluyen la transformación de algunas variables, el balanceo de carga, el contraste de hipótesis, la detección de variables hiperpredictoras y el estudio de la multicolinealidad entre variables.

Adicionalmente, el código de Python correspondiente a la elaboración de este capítulo está incluido en la sección A.4 del apéndice.




(a) Distribución de la variable *Flow Duration*




(b) Distribución de Flow Duration en función de *Label* tras la transformación.

Figura 4.1: Comparación tras la transformación logarítmica

## 4.1. Análisis de las distribuciones univariantes

El primer paso para comenzar el análisis exploratorio de los datos es comprende la naturaleza de los datos mediante el análisis de cada una de las variables y sus distribuciones. Este análisis permite identificar valores atípicos en las características del conjunto de datos que pueden ser resultado de comportamientos anómalos en la red y que podrían afectar a la precisión del modelo de clasificación. Con el objetivo de facilitar la visualización de estos valores atípicos, se han empleado diagramas de barras y *boxplots* para las variables con una distribución discreta y continua, respectivamente.

Se ha llevado a cabo la visualización de la distribución de todas las variables. Sin embargo, debido a la existencia de 66 características, se ha decidido focalizar la memoria en una variable específica llamada *Flow Duration* a modo de ilustración que represente todas las demás variables. Se puede observar que la distribución de esta variable presenta una distribución de cola pesada o *heavy tail distribution* (véase la figura 4.1a). Esta característica indica una concentración de valores atípicos en un extremo del espectro.

Frente a este problema, se ha planteado como solución la aplicación de una transformación logarítmica de los datos para intentar que su distribución se asemeje más a la de una distribución normal. Esta técnica matemática es frecuentemente utilizada para manejar distribuciones de cola pesada, ya que modifica la escala de los valores, reduciendo el impacto de los extremos. La transformación logarítmica normaliza los valores atípicos altos, resultando en una distribución más uniforme y menos sesgada. En particular, se transformará logarítmicamente todas las variables continúas del conjunto de datos, sumándolas 0,5 a su valor original para evitar la aparición del valor 0 en alguna operación [1].

Como solución a este problema, se ha planteado la transformación logarítmica de todas aquellas variables continuas que presenten este inconveniente con el ob-

```
Fwd Pkt Len Min: 6 valores distintos
Bwd Pkt Len Min: 5 valores distintos
Fwd PSH Flags: 2 valores distintos
FIN Flag Cnt: 2 valores distintos
SYN Flag Cnt: 2 valores distintos
RST Flag Cnt: 2 valores distintos
PSH Flag Cnt: 2 valores distintos
ACK Flag Cnt: 2 valores distintos
URG Flag Cnt: 2 valores distintos
ECE Flag Cnt: 2 valores distintos
Fwd Seg Size Min: 2 valores distintos
```

Figura 4.2: Señalización de variables discretas.

jetivo de intentar que su distribución se asemeje a la de una distribución normal. Esta técnica matemática es frecuentemente utilizada para manejar distribuciones de cola pesada, ya que modifica la escala de los valores, reduciendo el impacto en los extremos. Al aplicar la transformación, los datos se convierten a una escala en la que los valores atípicos más altos se normalizan, lo que resulta en una distribución más uniforme y menos sesgada. En particular, la transformación consiste en calcular el logaritmo del valor de la variable más 0,5. Esta suma se realiza con el propósito de evitar que el sistema realice el logaritmo de un valor de alguna variable que sea igual a 0.

Antes de realizar la transformación mencionada anteriormente, se ha realizado una diferenciación en el conjunto de datos para separar las variables discretas de las continuas. Esto es debido a que no se va a realizar la transformación logarítmica en las variables que tengan una distribución discreta (véase la figura 4.2).

La fórmula de la transformación logarítmica que se va a utilizar en este trabajo es la siguiente:

$$y = \log(x + 0{,}5) \tag{4.1}$$

Tras la transformación, se ha podido observar, en la figura 4.1b, que la mayor parte de las variables continuas mantenían la distribución de cola pesada, siendo insuficiente esta transformación para normalizarlas. Sin embargo, las variables *Fwd IAT Tot, Fwd IAT Mean, Fwd IAT Std* y *Fwd IAT Max*; han sido satisfactoriamente modificadas (véase figura 4.3 como ejemplo para *Flow IAT Tot*).

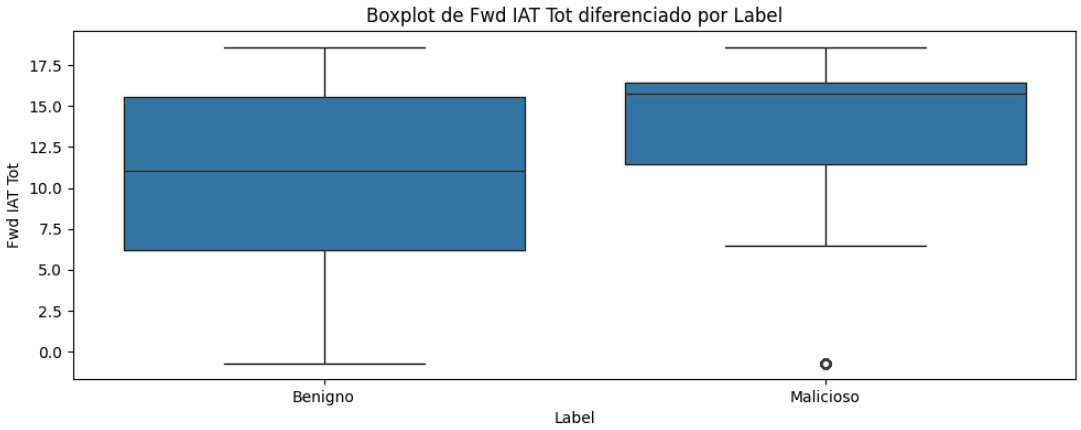



Figura 4.3: Distribución de la variable Fwd IAT Tot tras la transformación.

## 4.2. Discretización de variables

Tras la aplicación de la una transformación logarítmica a las variables que presentaban problemas de distribución de cola pesada, se ha observado que muchas de estas aún conservan dicha problemática. En respuesta a esto, se ha optado por la discretización de estas variables a partir de sus valores originales, es decir, sin haber realizado la transformación.

Este proceso de discretización se ha llevado a cabo mediante el entrenamiento de un árbol de decisión con una profundidad limitada a 2 niveles para cada variable. Este enfoque se centra en la variable destino, con el propósito de establecer cuatro intervalos distintos. Estos intervalos permitirán transformar las variables continuas en discretas, facilitando así su análisis e interpretación en el contexto del modelo [1] [17] [18]. Pese a que este proceso se ha realizado en todas las variables continuas que presentan dicha problemática, en la presente memoria se explicará este proceso en dos de esas variables. Estas variables son *Subflow Bwd Pkts* e *Init Fwd Win Byts*

Por una parte, para la variable *Subflow Bwd Pkts*, el árbol de decisión entrenado ha determinado cuatro categorías distintas en función de sus valores y la variable destino. Para empezar, el nodo de raíz se ramifica en dos nodos, diferenciando aquellos valores que son menores o iguales de 3,5 y los que son mayores. A continuación, estos nodos a su vez se dividen en otros dos nodos hoja cada uno, dividiendo los datos dependiendo si son menores o iguales a 1,5 y menores o iguales a 4,5 (véase la figura 4.4).

Como resultado, se ha obtenido una primera categoría para aquellas muestras con un valor igual a 1, una segunda categoría donde se incluyen los valores 2 y 3, una tercera para aquellos con valor 4 y una cuarta categoría para aquellas muestras con un valor entre 5 y 19.181 (véase la figura 4.5).

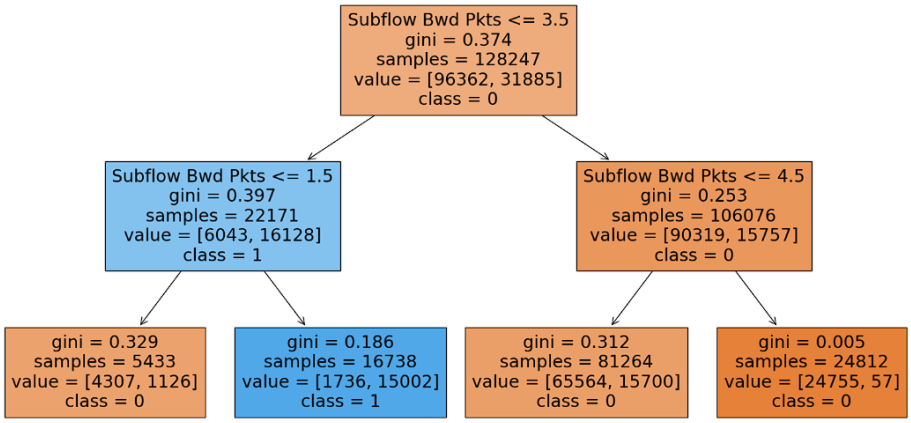



Figura 4.4: Árbol de decisión de la variable *Subflow Bwd Pkts"*.

| | Subflow Bwd Pkts | Subflow Bwd Pkts |
|---|---|---|
| **Subflow Bwd Pkts tree** | | |
| **1** | 1 | 1 |
| **2** | 2 | 3 |
| **3** | 4 | 4 |
| **4** | 5 | 19181 |

Figura 4.5: Categorías de la variable *Subflow Bwd Pkts"*.

A continuación, en la figura 4.6, se representa la distribución de la variable mediante un diagrama de barras donde se puede observar la cantidad de muestras legítimas y maliciosas por cada categoría, siendo el valor 0 tráfico legítimo y el valor 1 tráfico malicioso.

Por otra parte, en el caso de la variable *Init Fwd Win Byts* el árbol de decisión empieza por un primer nodo raíz donde se dividen los datos menores o iguales a 22.181,5 y, posteriormente, de los dos nodos resultantes de la anterior división se dividen en otros dos nodos hoja separando los datos entre los datos con valores menores o iguales a 227,5 y los datos menores o iguales a 28.041 (véase la figura 4.7).

Tras haber realizado el árbol de decisión, la variable ahora tan solo tiene cuatro posibles categorías. La primera categoría abarca todos los valores del 0 al 226, la segunda del 229 al 17.480, la tercera categoría es para aquellas muestras con valor 26.883 y, por último, la cuarta categoría para aquellos datos con valor entre 29.200 y 65.535 (véase la figura 4.8). La distribución de esta variable discretizada se puede ver en la figura 4.9.

Como se puede observar, la cuarta categoría tiene un número muy bajo de

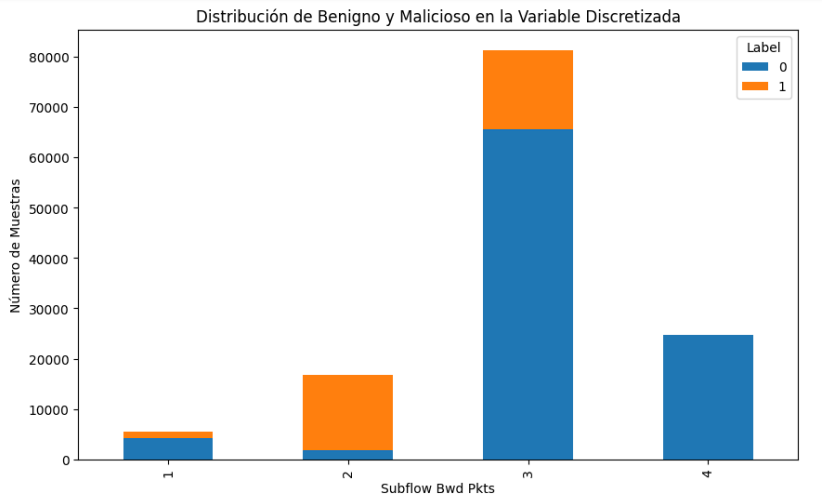



Figura 4.6: Distribución de la variable *Subflow Bwd Pkts"*.

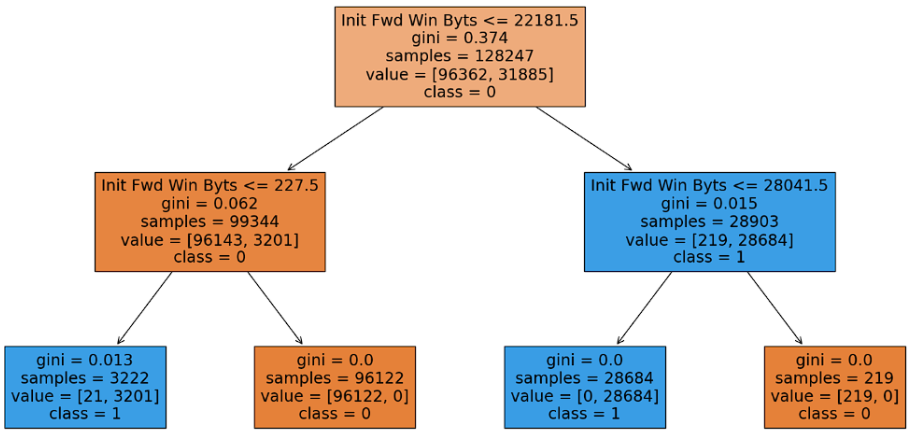



Figura 4.7: Árbol de decisión de la variable *Init Fwd Win Byts*.

muestras en comparación al resto. Por lo tanto, con el objetivo de optimizar el modelado, se realizará una unión de la tercera categoría y la cuarta, teniendo en cuenta que la distribución de la categoría resultante no supondrá una perdida de precisión considerable (véase la figura 4.10).

Tal y como ha ocurrido anteriormente durante el proceso de discretización, se ha observado la presencia de categorías con un número muy pequeño de muestras o que resultan poco relevantes para el análisis. En estos casos, se ha optado por fusionar los intervalos de estas categorías con el de la categoría más cercana.

Además, en algunas variables, se ha detectado que la mayoría de las muestras se concentraban en una única categoría. Este desequilibrio implica que dichas variables ofrecen poca variabilidad o información diferencial asemejándose a características constantes y, por lo tanto, no siendo de utilidad para el análisis. Por

| Init Fwd Win Byts tree | Init Fwd Win Byts | Init Fwd Win Byts |
|---|---|---|
| **1** | 0 | 226 |
| **2** | 229 | 17480 |
| **3** | 26883 | 26883 |
| **4** | 29200 | 65535 |

Figura 4.8: Categorías de la variable *Init Fwd Win Byts*.

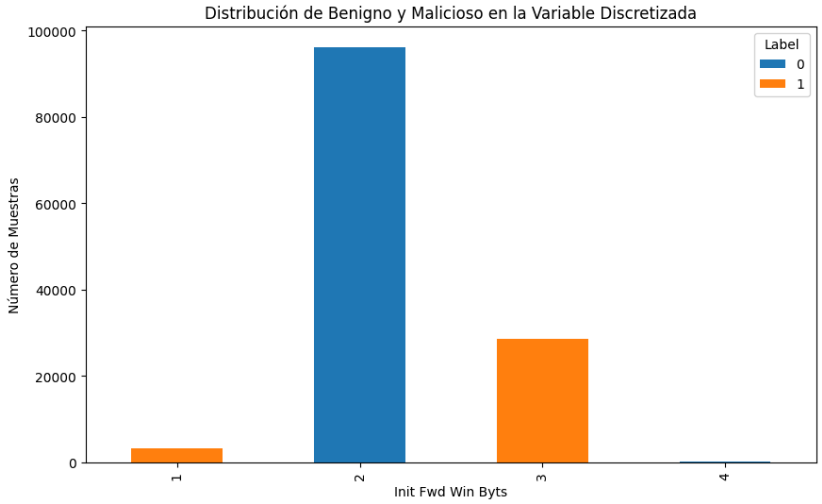



Figura 4.9: Distribución de la variable *Init Fwd Win Byts*.

este motivo, se ha decidido eliminar estas variables del conjunto de datos. Dichas variables son las siguientes: *Fwd Pkt Len Min, Bwd Pkt Len Min, Fwd PSH Flags, FIN Flag Cnt, SYN Flag Cnt, PSH Flag Cnt, ACK Flag Cnt* y *URG Flag Cnt*.

Tras la eliminación de estas variables, el conjunto de datos tiene 128.247 filas y 58 columnas.

## 4.3. Balanceo de carga

El balanceo de carga se refiere a la técnica utilizada para igualar la distribución de las clases de la variable destino en el conjunto de datos. Un conjunto de datos desequilibrado se manifiesta cuando una de la clase tiene muchas más muestras que otra. Este desequilibrio puede causar problemas en el entrenamiento de modelos, ya que tienden a sesgarse hacia la clase más representada, lo que afecta al rendimiento de predicción, siendo además de interés, normalmente, la clase minoritaria.

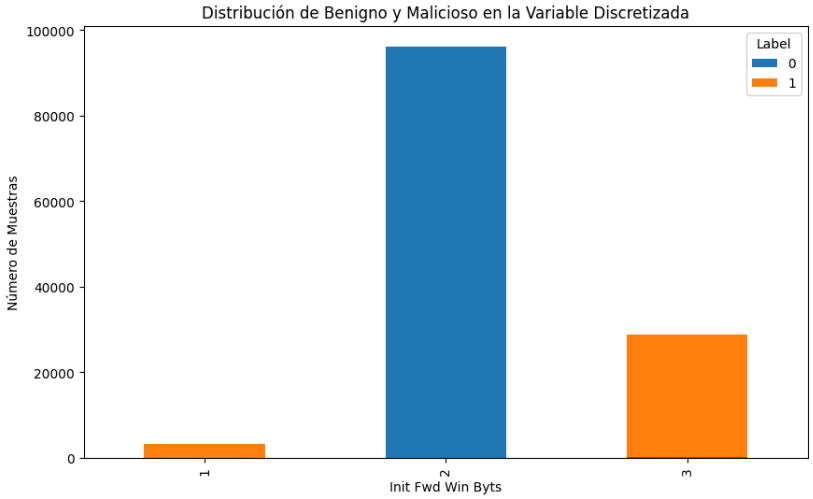



Figura 4.10: Distribución de la variable *Init Fwd Win Byts* fusionando categorías.

Existen diferentes maneras de abordar el desequilibrio de clases, entre las que destacan el submuestreo y el sobremuestreo. El submuestreo implica reducir las muestras de la clase más representada para igualar la distribución con la clase menos representada. Por su parte, el sobremuestreo busca aumentar las muestras de la clase menos representada.

El uso del submuestreo puede ser una opción adecuada por diferentes motivos. En primer lugar, al reducir el número de muestras, el proceso de análisis y entrenamiento del modelo puede ser más rápido y requerir menos recursos computacionales. Además, al emplear solo datos reales y no generar datos sintéticos, se mantiene la autenticidad y calidad de la información.

Por este motivo, se implementará un submuestreo de 3.000 muestras, de forma que 1.500 sean muestras legítimas y las otras 1.500 muestras sean maliciosas. Esta decisión se basa en la necesidad de manejar de manera eficiente el volumen de datos, al tiempo que se busca conservar la representatividad y diversidad de la información contenida en el conjunto de datos original. Con 3.000 muestras, se puede lograr un balance entre la calidad de los datos y la eficiencia computacional, facilitando así un análisis más ágil y un entrenamiento del modelo menos costosos en términos de tiempo y recursos.

## 4.4. Contraste de hipótesis

Uno de los objetivos del desarrollo de este modelo de análisis de datos es la comprensión de las variables y cómo estas se relacionan. Por este motivo, es necesario estudiar el nivel de relación entre las variables descriptivas de cada una de las muestras del conjunto y la variable destino u objetivo. Para ello, se ha

decidido emplear un contraste de hipótesis.

El contraste de hipótesis es un proceso en el que se definen dos hipótesis opuestas que determinan el análisis estadístico. En estadística estas hipótesis reciben el nombre de hipótesis nula e hipótesis alternativa. La hipótesis nula es aquella que se mantiene mientras los datos de la muestra no reflejen lo contrario. Se representa como $H_0$. En este caso, la hipótesis nula en todos los casos será la suposición de que no existe una relación significativa entre la variable independiente y la variable dependiente o destino. Por otra parte, la hipótesis alternativa es la negación de la hipótesis nula y generalmente representa la afirmación que se pretende probar. Se representa como $H_1$. En este caso, será la suposición de que existe una relación significativa entre la variable independiente y la variable destino.

Tras la definición de ambas hipótesis, se ha establecido un nivel de significancia adecuado. El nivel de significancia es la probabilidad de rechazar la hipótesis nula cuando esta es cierta. Comúnmente se utiliza 0,05 como valor, pero este puede ajustarse según el contexto de estudio. En el caso de este EDA, se ha establecido el valor mencionado anteriormente, lo que significa que debe existir una probabilidad de 0,05 de rechazar la hipótesis nula cuando esta sea verdadera [1].

Por otro lado, para el análisis de las variables, se ha seleccionado la prueba t-Student para aquellas variables con una distribución continua y la prueba Chi-cuadrado para las variables discretas.

La prueba t-Student se utiliza cuando los datos siguen una distribución normal, como la conseguida tras la transformación logarítmica, y se tiene como objetivo la comparación de las medias entre dos grupos. En este análisis, la prueba t-Student ha sido empleada para determinar si existen diferencias estadísticamente significativas en las medias de las variables continuas en relación con la variable destino.

Por otro lado, se ha decidido utilizar la prueba Chi-cuadradro para el estudio de las variables discretas originales de conjunto y las variables continuas que han sido discretizadas en etapas anteriores. Al igual que con la prueba t-Student el propósito de emplear esta prueba es determinar si existen diferencias estadísticamente significativas entre las viables discretas y la variable objetivo.

Sin embargo, se ha observado en algunas de estas variables discretas ciertas categorías que no tienen el mínimo número de muestras requeridas para realizar una prueba Chi-cuadrado efectiva. En tales casos, se ha planteado la implementación del test de Fisher como alternativa, ya que es particularmente útil en situaciones donde el tamaño de la muestra es pequeño, proporcionando una alternativa fiable para analizar la relación entre las variables discretas y la variable destino.

Sin embargo, debido a que la implementación del test de Fisher en Python está diseñada para variables descriptivas de solo dos categorías, se ha decidido

implementar la corrección de Yates en las pruebas de Chi-cuadrado. Esta corrección ajusta los resultados para tener en cuenta el tamaño pequeño de la muestra y proporcionar una estimación más precisa [19].

En ambos casos, ya sea por la prueba de t-Student como por la prueba Chi-cuadrado, el p-valor obtenido será el que determine el grado de dependencia entre las variables. Un p-valor menor a 0,05 generalmente indica una relación significativa. Si se encuentra un p-valor por debajo de este umbral, se puede concluir que existe suficiente evidencia para rechazar la hipótesis nula, sugiriendo una relación significativa entre las variables descriptivas y la variable destino.

Tras haber realizado estas pruebas con el conjunto de datos preprocesado con anterioridad, se ha observado que, en los test realizados, todas las variables tienen un p-valor asociado menor a 0,05. Por lo tanto, en todas las variables se tiene evidencia estadística para rechazar la hipótesis nula. Así, se puede concluir que todas las variables son dependientes de la variable objetivo, *Label*.

Una vez concluido el contraste de hipótesis, el conjunto de datos podría estar preparado para la generación de un modelo capaz de entrenar sobre estos y posteriormente predecir si una muestra se trata de tráfico legítimo o se trata de un ataque de Denegación de Servicio.

## 4.5. Detección de variables hiperpredictoras

Tal y como se ha podido observar en los diagramas de barras de las variables, existen algunas características que discriminan la variable objetivo por completo y, por tanto, podrían predecir el resultado de la variable objetivo por sí solas. Un ejemplo de esto, podría ser la variable *Fwd Seg Size Min* (véase la figura 4.11). De esta manera, no sería necesario entrenar con las demás variables debido a que mediante esta característica no habría ningún error.

Las variables hiperpredictoras pueden presentar un problema a la hora de realizar un modelo de análisis de datos. Estas características pueden sesgar un modelo al incluir sobreajuste, donde el modelo aprende patrones específicos de los datos de entrenamiento, que puede que no se generalicen a nuevos datos, debido a su correlación anormalmente alta con la variable objetivo. Esta situación puede derivar de una relación no causal o altamente específica entre la variable hiperpredictora y la variable destino, lo que conduce a conclusiones erróneas sobre cómo las variables interactúan en la realidad. A simple vista, las variables que podrían ser hiperpredictoras son: *Fwd Seg Size Min, Init Fwd Win Byts, Init Bwd Win Bytes* y *Flow IAT Min*.

Como se puede observar en la figura 4.11, la variable *Fwd Seg Size Min* es capaz de predecir, de manera correcta, el tipo de tráfico de una muestra en fun-

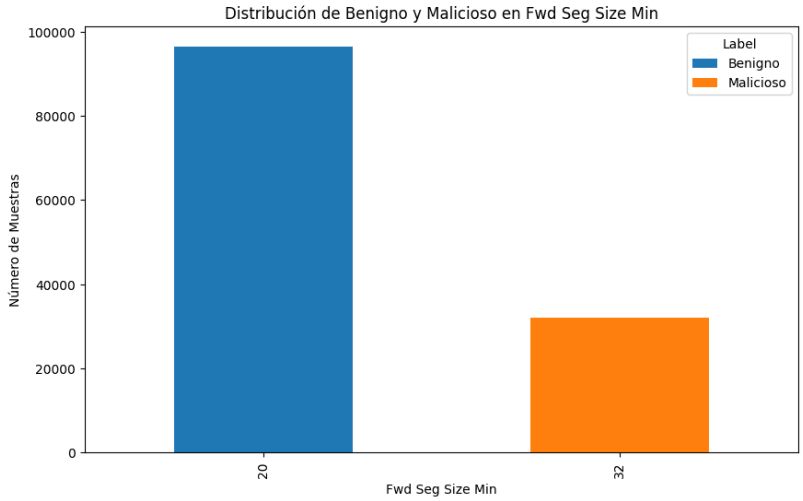



Figura 4.11: Diagrama de barras de la variable *Fwd Seg Size Min*

ción del valor que tenga. Sin embargo, antes de realizar alguna acción con estas variables, es necesario hacer un estudio de la definición de cada una de estas características y analizar si pueden estar, *a priori*, relacionadas con el tipo de tráfico que se realiza hacia el servidor, es decir, con la variable objetivo.

En primer lugar, la variable *Fwd Seg Size Min* se refiere al tamaño mínimo de segmento observado en el tráfico que se dirige desde el origen al destino. En ataques utilizando *Slowloris* o *Golden Eye*, donde el atacante intenta abrir múltiples conexiones a un servidor con la intención de mantenerlas abiertas el mayor tiempo posible con cabeceras HTTP, un tamaño mínimo del segmento más pequeño que del normal podría indicar que se está intentando utilizar pequeños segmentos para ocupar recursos del servidor de manera prolongada. Por lo tanto, esta característica podría predecir si un tráfico de red se realiza con fines maliciosos o no.

En segundo lugar, las variables *Init Fwd Win Byts* e *Init Bwd Win Byts* son características que reflejan el tamaño de la ventana de congestión inicial en la conexión TCP, tanto en el sentido de ida como en el de vuelta, respectivamente. La ventana de congestión en TCP es un mecanismo diseñado para regular la cantidad de datos que pueden ser enviados a la red sin confirmación. En el contexto de un ataque DoS mediante las dos herramientas nombradas anteriormente, un ciberatacante podría manipular el tamaño de la ventana para realizar un ataque de inundación o para mantener abiertas las conexiones de manera ineficiente. Un tamaño anormalmente grande o pequeño podría ser utilizado para manipular cómo se gestionan los recursos del servidor y cómo se controla la congestión, lo que podría ser un indicador de actividad maliciosa.

Por último, *Flow IAT Min* se refiere al tiempo mínimo entre la llegada de paquetes consecutivos en un flujo de datos. En el caso de ataques empleando

*Slowloris* o *Golden Eye*, el tiempo de llegada entre los segmentos de datos podría manipularse para ser anormalmente largo, para mantener la conexión abierta con el mínimo tráfico posible, o corto, para enviar ráfagas de tráfico y sobrecargar el servidor. Un valor anómalo en este parámetro podría indicar tanto intentos de mantener las conexiones abiertas de manera artificial como intentos de inundar el servidor con tráfico.

Tras haber estudiado las distintas variables, se ha comprobado que ninguna de estas ha podido ser generada tras la creación de la variable objetivo. Sin embargo, debido a que son características hiperpredictoras este hecho podría llevar a un problema de sobreajuste en el modelo, ya que si se cambiaran en el futuro estos parámetros o no se registrasen para su análisis, el modelo quedaría obsoleto. Por este motivo, se procederá a la eliminación de estas características para evitar este problema.

## 4.6. Análisis de multicolinealidad y selección de variables

Debido a que el conjunto de datos presenta un número elevado de variables incluso tras haber realizado distintas técnicas durante el desarrollo del modelo, es necesario realizar un estudio de la multicolinealidad entre las variables para evitar la existencia de información redundante dentro del modelo. Esta redundancia puede resultar en variaciones a la hora de añadir o eliminar datos.

Mediante un análisis de multicolinealidad se detectan las variables redundantes con respecto al resto. Estas variables pueden ser eliminadas y se puede trabajar con el conjunto restante. Este tipo de análisis es común para el empleo de modelos de regresión logística, sin embargo, es posible su utilización en problemas de clasificación para la selección de variables. En el caso de este trabajo, se ha observado que el conjunto de datos con el que se está trabajando presenta un número elevado de variables, por lo que una selección de estas permite reducir la complejidad del problema y llegar a un modelo más sencillo.

Adicionalmente, la existencia de multicolinealidad complica la interpretación de cómo las variables independientes influyen en la variable objetivo, ya que es difícil discernir si el efecto de una variable es genuino o está influido por su correlación con otras.

Para abordar el estudio de la multicolinealidad, existen diversas soluciones. En este trabajo se ha decidio emplear el análisis del Factor de Inflación de la Varianza (VIF, por sus siglas en inglés) en las variables discretas. Este análisis consiste en la medición del aumento de la varianza de un coeficiente de regresión debido a la correlación con otras variables.

| | variables | VIF |
|---|---|---|
| 0 | Flow Duration | 740.611718 |
| 1 | Tot Fwd Pkts | inf |
| 2 | Tot Bwd Pkts | inf |
| 3 | TotLen Fwd Pkts | inf |
| 4 | TotLen Bwd Pkts | inf |
| 5 | Fwd Pkt Len Max | 667.538429 |
| 6 | Fwd Pkt Len Mean | inf |
| 7 | Fwd Pkt Len Std | 135.458900 |
| 8 | Bwd Pkt Len Max | 585.176206 |
| 9 | Bwd Pkt Len Mean | inf |
| 10 | Bwd Pkt Len Std | 475.077565 |
| 11 | Flow Byts/s | 137.768214 |
| 12 | Flow Pkts/s | 736.018275 |
| 13 | Flow IAT Mean | 532.334526 |
| 14 | Flow IAT Std | 992.696063 |
| 15 | Flow IAT Max | 981.569397 |

(a) Extracto del análisis del primer VIF

| | variables | VIF |
|---|---|---|
| 0 | Flow Duration | 739.047716 |
| 1 | Tot Fwd Pkts | inf |
| 2 | Tot Bwd Pkts | inf |
| 3 | TotLen Fwd Pkts | inf |
| 4 | TotLen Bwd Pkts | inf |
| 5 | Fwd Pkt Len Max | 667.481466 |

(b) Extracto del análisis del segundo VIF

| | variables | VIF |
|---|---|---|
| 0 | Tot Fwd Pkts | inf |
| 1 | Tot Bwd Pkts | 5.739309e+04 |
| 2 | TotLen Fwd Pkts | 6.388584e+04 |
| 3 | TotLen Bwd Pkts | 4.076616e+03 |
| 4 | Fwd Pkt Len Max | 6.467620e+02 |
| 5 | Fwd Pkt Len Mean | 2.537308e+04 |
| 6 | Fwd Pkt Len Std | 1.350428e+02 |
| 7 | Bwd Pkt Len Max | 5.845497e+02 |

(c) Extracto del análisis del tercer VIF

Figura 4.12: Análisis del VIF

Un VIF alto indica una fuerte multicolinealidad, sugiriendo que la variable correspondiente podría necesitar ser removida para reducir la redundancia. Es importante señalar que el proceso de selección de variables es iterativo para alcanzar un resultado satisfactorio. Esto implica que, tras la eliminación de una característica, es necesario reevaluar el análisis, dado que el VIF puede experimentar cambios. Este proceso de revisión y reajuste se repite hasta lograr la configuración óptima de variables [11].

Como se puede observar en la figura 4.12a, la variable *Flow IAT std* tiene el VIF más alto de todas las variables discretas. Debido a esto, se ha procedido con su eliminación para disminuir la redundancia. Tras haber sido eliminado, se ha vuelto a realizar un nuevo análisis del VIF.

Del mismo modo que la primera vez, se ha observado que la *Flow Duration* es la que tiene el VIF más alto (véase la figura 4.12b), por lo que se ha borrado del conjunto de datos.

Tras haber realizado el análisis por tercera vez, se ha observado que los valores VIF de las variables restantes son notablemente bajos, es decir, tienen un grado de multicolinealidad muy débil. Debido a esto, se ha decidido no realizar más iteraciones del VIF (figura 4.12c).

| subspace | variables | | %n accum. | | | %n | | | kDN |
|---|---|---|---|---|---|---|---|---|---|
| | | | total | class | | total | class | | |
| | | | | Benigno | Malicioso | | Benigno | Malicioso | |
| 1 | Fwd.IAT.Max | Fwd.Header.Len_2 | 98.12 | 97.28 | 98.96 | 98.12 | 97.28 | 98.96 | 0 |

Figura 4.13: Separabilidad entre el par *Fwd IAT Max* y *Fwd Heather Len_2*.

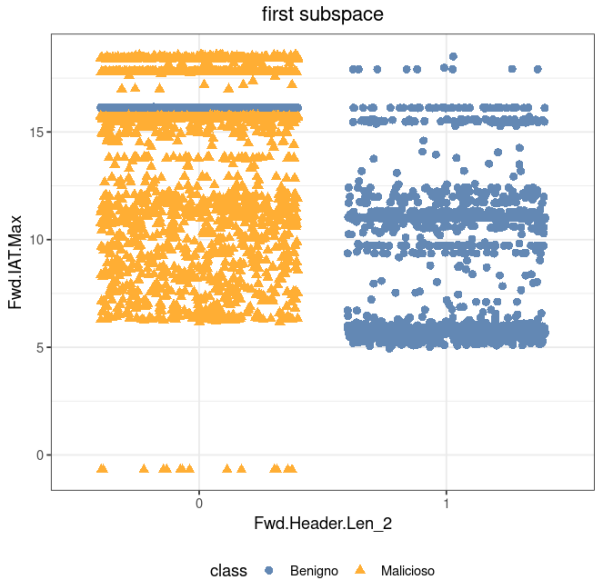



Figura 4.14: Visualización de la separabilidad entre el par *Fwd IAT Max* y *Fwd Heather Len_2*.

## 4.7. Visualización de la separabilidad de clases

Con el objetivo de profundizar más en la distribución de la variable clase en función de las variables explicativas, se ha realizado un análisis visual de la separabilidad de las clases.

Se ha implementado un análisis utilizando la herramienta CSViz (*Class Separability Visualization* [20], por sus siglas en inglés) para determinar qué pares de variables descriptivas maximizan la separabilidad entre las clases de la variable objetivo. Este método analiza la distancia y la superposición entre las distribuciones de clases en un espacio bidimensional definido por cualquier par de variables y devuelve los pares en los que la variable objetivo tiene patrones más separables. De este modo, facilita la identificación de las variables que, en combinación, proporcionan la mayor discriminación de grupos. Finalmente, se ha observado que el subespacio formado por las variables *Fwd IAT Max* y *Fwd Header Len_2* es el subconjunto de puntos que más separa la variable *Label*. Adicionalmente, este subconjunto comprende más del 98 % de los datos (véase la figura 4.13 y 4.14).

# 5 Modelado

Tras la recopilación y la preparación de los datos, el siguiente paso en el trabajo es la creación de un modelo de aprendizaje supervisado que sea capaz de entrenar con el conjunto de datos actual y predecir el resultado con la mayor exactitud posible cuando entren datos nunca antes vistos por el modelo.

En el presente trabajo, se ha optado por estudiar los resultados de dos modelos diferentes. Por una parte, se utilizará un árbol de decisión para entrenar y evaluar los resultados y, por otra parte, se empleará un *Random Forest*.

Antes de comenzar con el entrenamiento de los modelos, es importante la partición del conjunto de datos en dos subconjuntos distintos: uno de entrenamiento y otro de prueba. Para el conjunto de datos utilizado en el entrenamiento, se han escogido los 3.000 datos balanceados mencionados en el apartado 4.3 para que el modelo no generalice sus predicciones con muestras legítimas. Asimismo, con el objetivo de evaluar el modelo con una muestra que se ajuste más a la realidad, se ha decidido utilizar un conjunto de prueba de 2.000 datos con la proporción original de muestras legítimas y maliciosas.

En el desarrollo de modelos de árbol de decisión y *Random Forest* en Python, es importante entender y configurar los hiperparámetros para optimizar el rendimiento del modelo. Los hiperparámetros son configuraciones externas al modelo que influyen en su comportamiento y en su capacidad predictiva. Para los árboles de decisión y *Random Forest*, algunos hiperparámetros básicos incluyen la profundidad máxima del árbol (*max_depth*), el número mínimo de muestras necesarias para dividir un nodo (*min_samples_split*), y el número de árboles en el modelo (*n_estimators*) en el caso de *Random Forest*.

Para lograr la configuración óptima, se ha utilizado la técnica *Grid Search* junto con una estrategia de validación cruzada (*cross-validation*, en inglés). Por una parte, *Grid Search* es una técnica que implica la búsqueda a través de un conjunto específico de valores de hiperparámetros, evaluando el rendimiento del modelo según una métrica predeterminada, en este caso la exactitud o *accuracy*. Por otra parte, una estrategia de validación cruzada consiste en dividir el conjunto de datos en partes iguales, utilizando sucesivamente una parte como conjunto de prueba y el resto como conjunto de entrenamiento con el objetivo de mejorar la precisión.

Para consultar el código utilizado en este capítulo, diríjase al apéndice A.5 del apéndice.

## 5.1. Árbol de decisión

Tal y como se ha mencionado anteriormente, el primer modelo de aprendizaje automático que se ha explorado es el árbol de decisión, el cual su principal funcionalidad es identificar las características que realizan mejores divisiones en los datos

Al igual que lo mencionado en el capítulo 2, una de las principales ventajas que tiene este modelo en comparación con otros es su interpretabilidad, ya que es fácil entender cómo se llega a una decisión mediante la visualización del árbol y siguiendo el camino desde la raíz hasta una hoja [12].

Se ha procedido a la selección y configuración de los hiperparámetros con el objetivo de optimizar el rendimiento del árbol. Los hiperparámetros más comunes a la hora de crear un árbol de decisión son *criterion*, que determina la función para evaluar la calidad de una división, *max_depth* que establece la profundidad máxima del árbol, *min_samples_split* que define el mínimo número de muestras para dividir un nodo, y *min_samples_leaf* que especifica el número mínimo de muestras en una hoja. Estos hiperparámetros ayudan a controlar la complejidad del modelo, prevenir el sobreajuste y asegurar una adecuada generalización.

Tras utilizar *CV Grid Search*, se ha determinado que la configuración óptima de hiperparámetros para mejorar la precisión del modelo incluye el criterio de Gini para medir la calidad de las divisiones, una profundidad máxima de 9 niveles para el árbol, al menos una muestra para considerarse como nodo hoja y como mínimo dos muestras para dividir un nodo interno. Adicionalmente, se ha observado que la puntuación de exactitud media del árbol de decisión, con esta configuración, es de 0,999.

Cabe señalar que entrenar un árbol de decisión con una profundidad de 9 niveles puede llevar al sobreajuste del modelo, a pesar de obtener la mejor pun-

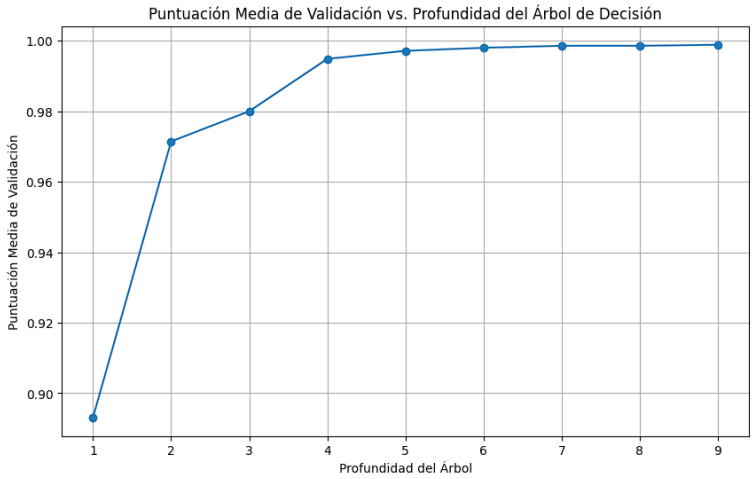



Figura 5.1: Comparación entre la profundidad del árbol de decisión y exactitud media

tuación de exactitud. Por lo tanto, se ha realizado un estudio de las exactitudes obtenidas con diferentes configuraciones en función de la profundidad del árbol. A partir del análisis llevado a cabo (véase la figura 5.1), se ha observado que la exactitud media mejora notablemente hasta llegar al nivel 4 de profundidad. A partir de ese nivel, la mejora solo es ligera. Debido a esto, en base con el principio de parsimonia, que sugiere que, en igualdad de condiciones, la solución más sencilla es la preferida, se ha decidido que la configuración utilizada para el árbol de decisión es con una profundidad de 4 niveles.

Una vez encontrada la configuración óptima de los hiperparámetros, se vuelve a entrenar un árbol de decisión sobre toda la muestra del entrenamiento de los datos (la figura 5.2 contiene el árbol de decisión ajustado). Adicionalmente, se ha establecido el umbral de predicción basado en los *F1 score*, indicando que el *F1 score* más alto se obtiene con un umbral entre 0,2 y 0,75 (véase la figura 5.3). Sin embargo, se elige el umbral de 0,7 porque es preferible que el modelo genere más falsos negativos que falsos positivos. Esto es debido a que los ataques de Denegación de Servicio pueden ser mitigados con la detección y mitigación parcial de los paquetes maliciosos recibidos.

Una vez entrenado, se ha realizado una predicción de la variable objetivo con el conjunto de prueba. Como resultado, el *F1 score* del árbol, sobre el conjunto de prueba, ha sido de un 0,981 y la exactitud 0,991. Si se compara este resultado con el del árbol de decisión utilizado en el *CV Grid Search*, se puede observar que son bastante similares y, por lo tanto, se evidencia que no existe ningún error de distribución entre ambos conjuntos de entrenamiento que hayan podido dar configuraciones de hiperparámetros erróneas.

Además, se han identificado que las variables más determinantes dentro del árbol de decisión han sido *Pkt Size Avg*, *Flow IAT Max* y *Subflow Bwd Byts*.

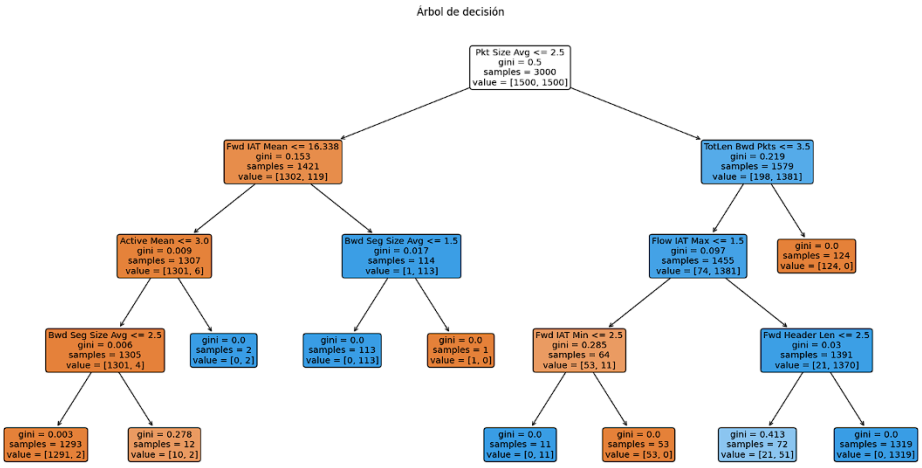



Figura 5.2: Árbol de decisión

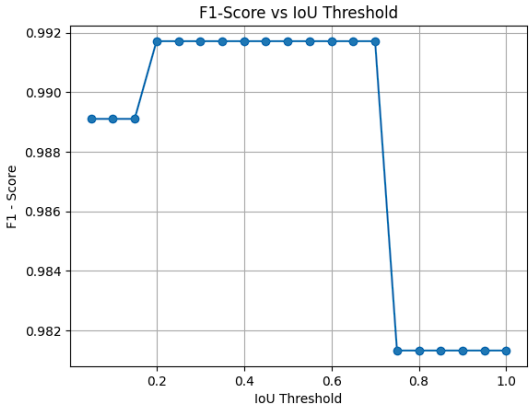



Figura 5.3: Comparación entre el F1 score del árbol de decisión y el umbral de predicción

## 5.2. Random Forest

Después de estudiar el modelo de árbol de decisión, se ha procedido con la investigación del modelo de *Random Forest*.

El modelo de *Random Forest* funciona construyendo múltiples árboles de decisión durante el entrenamiento y dando como resultado la clase más frecuente entre las predicciones de todos los árboles individuales. La principal ventaja de este modelo es su capacidad para reducir el riesgo de sobreajuste [1].

Del mismo modo que lo realizado en el modelo de árbol de decisión, se ha procedido con la búsqueda de la configuración más adecuada mediante *Grid Search* y la validación cruzada. Se ha determinado que los hiperparámetros óptimos para el modelo de *Random Forest* del trabajo incluye un criterio de *entropy* para medir la calidad de las divisiones mediante la ganancia de información, una profundidad

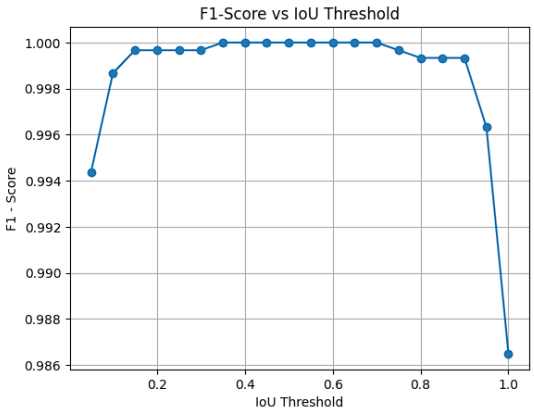



Figura 5.4: Comparación entre el F1 score del *Random Forest* y el umbral de predicción

| predictor | importancia |
|---|---|
| Pkt Len Mean | 0.093890 |
| Pkt Size Avg | 0.078667 |
| Bwd IAT Min | 0.066601 |
| Fwd Pkt Len Mean | 0.050661 |
| Fwd Header Len | 0.045407 |

Figura 5.5: Variables más importantes del Random Forest.

máxima configurada como *None*, permitiendo que los árboles crezcan tanto como sea necesario; *max_features* fijado en 5, limitando el número de características a considerar cuando se busca la mejor división; y el número de árboles en el bosque limitado a 50. La exactitud media del *Random Forest* con esta configuración dentro del *Grid Search* es de 0,999.

Tras haber encontrado los mejores hiperparámetros, se ha realizado el entrenamiento del modelo de *Random Forest* sobre toda la muestra de entrenamiento y, posteriormente, su evaluación con el conjunto de prueba. Cabe señalar, al igual que se realizó en el árbol de decisión, se han analizado los *F1 scores* dependiendo del umbral de predicción en este modelo (véase la figura 5.4). Como resultado, el F1 score de este modelo en la muestra de prueba, con la configuración señalada y aplicando un umbral de 0,65, es de 0,997. Por otro lado, se ha observado que la exactitud del *Random Forest* es de 0,999.

A partir de este modelo, se puede determinar las variables más importantes a la hora de predecir el resultado de la variable objetivo. En este caso, las características más importantes han sido *"Pkt Len Mean"*, *"Pkt Size Avg"* y *"Bwd IAT Min"*, tal y como se puede ver en la figura 5.5.

## 5.3. Selección del mejor modelo

La selección del mejor modelo entre el árbol de decisión y el bosque aleatorio ajustados en las secciones anteriores implica considerar varios factores, incluyendo los resultados generados y la complejidad del modelo. Tras la experimentación de estos con la muestra de datos del proyecto, se observa, en la tabla 5.1, que el árbol de decisión alcanza un *F1 score* de 0,992 en la muestra *train* y 0,981 en la muestra *test*, mientras que el *Random Forest* logra un *F1 score* ligeramente superior de 1 en *train* y 0,997 en la muestra *test*.

| | **Árbol de decisión** | **Bosque aleatorio** |
|---|---|---|
| Muestra *train* | 0,992 | 1 |
| Muestra *test* | 0,981 | 0,997 |

Tabla 5.1: Comparativa de *F1 score* entre modelos.

Aunque el *Random Forest* muestra un mayor resultado, la diferencia en el desempeño es marginal. Este incremento viene acompañado de un aumento significativo en la complejidad computacional y en el uso de recursos, dado que el *Random Forest* crea múltiples árboles de decisión y combina sus resultados, lo cual requiere más poder de procesamiento y memoria.

Dado que uno de los objetivos de este sistema de detección de ataques de Denegación de Servicio es la eficiencia en el uso de recursos, la velocidad y la interpretabilidad del modelo, la elección óptima es el árbol de decisión. A pesar de ofrecer un *F1 score* ligeramente inferior, su simplicidad estructural garantiza un menor consume de recursos y una mayor rapidez tanto en el entrenamiento como en la fase de predicción.

Por lo tanto, considerando la necesidad de un modelo que maximice la eficiencia y la minimice el consumo de recursos sin comprometer significativamente la precisión, se elige el árbol de decisión como el modelo más adecuado para abordar el problema.

# 6

# Nuevos Datos

Para finalizar este proceso de Ciencia de Datos, en este capítulo se abordará una simulación de puesta en producción real en la que se pone a prueba el modelo desarrollado en los anteriores capítulos con datos nuevos que no se han tratado previamente en ningún momento.

Por otro lado, puede consultar el código empleado durante el desarrollo de este capítulo en el apéndice A.6 del apéndice A

## 6.1. Obtención, preparación y predicción de los datos

El principal objetivo de esta simulación es evaluar el modelo de árbol de decisión entrenado con datos nuevos recogidos por el servidor donde se implementa esta herramienta. Para realizar esta prueba, se utilizarán los datos reservados mencionados en la sección de Recopilación del tercer capítulo.

Estos datos van a ser predichos por el modelo de manera secuencial para recrear lo más exacto posible el tráfico natural de la red.

El número total de muestras de este conjunto de datos es de 104.858 muestras. Sin embargo, debido a la limitación que tiene Jupyter Notebook en su capacidad de computación, se ha determinado que el número de muestras a utilizar será de 10.486, en las que se incluyen muestras tanto legítimas, como maliciosas. Esta

decisión se ha tomado debido a que el entorno de Jupyter Notebook no era capaz de procesar la predicción de tantas muestras de forma secuencial.

Para que el árbol de decisión sea capaz de predecir los nuevos datos de forma correcta, estos datos han de pasar por los mismos procesos de preprocesado que los datos que han servido para el desarrollo del modelo. Esto incluye la eliminación de todas las columnas desestimadas debido a su irrelevancia o por ser demasiado hiperpredictoras, además de la transformación logarítmica de algunas de las variables descriptivas, así como de la discretización de múltiples variables siguiendo los puntos de corte obtenidos en la muestra de entrenamiento. Cabe señalar que la discretización es realizada de forma manual aplicando los intervalos establecidos en el árbol entrenado para la discretización del conjunto de datos utilizado durante el desarrollo del modelo.

Una vez realizada la preparación de los datos. Se ha realizado la predicción individual de cada uno mediante el árbol de decisión entrenado anteriormente en la sección de Árbol de Decisión de la sección 5.1.

## 6.2. Resultados

Tras la predicción de todos los datos escogidos para la simulación, es necesario interpretar los resultados obtenidos. En primer lugar, se ha observado que el *F1 score* medio del árbol de decisión con este conjunto de datos ha sido de 0,667. Este resultado es notablemente inferior a la exactitud media de este modelo con el conjunto utilizado durante su desarrollo.

Asimismo, en este trabajo se determina que el servidor está bajo el ataque de una Denegación de Servicio cuando el modelo predice 5 o más paquetes de tráfico malicioso seguidos. Es necesario explicar que este protocolo para determinar un ataque de Denegación de Servicio ha sido creado con el objetivo de ejemplificar la aplicación del modelo. Una vez notificado el aviso, se abre un abanico de posibilidades en el que se podría bloquear el tráfico hacia el servidor de este tipo de paquetes para evitar que surta efecto el ataque o, incluso, desconectar de la red el servidor para evitar el agotamiento de sus recursos.

A continuación, se puede observar en la figura 6.1 una representación de la detección por el modelo de aprendizaje automático de rachas de paquetes maliciosos. En esta representación gráfica, se identifican las secuencias consecutivas de paquetes clasificados como maliciosos, siendo cada punto de la gráfica el final de una racha. Por ejemplo, una racha de tres paquetes maliciosos consecutivos se representa con el valor de tres en la gráfica.

Asimismo, la línea roja discontinua indica el umbral de detección de rachas establecido en 5 paquetes maliciosos consecutivos.

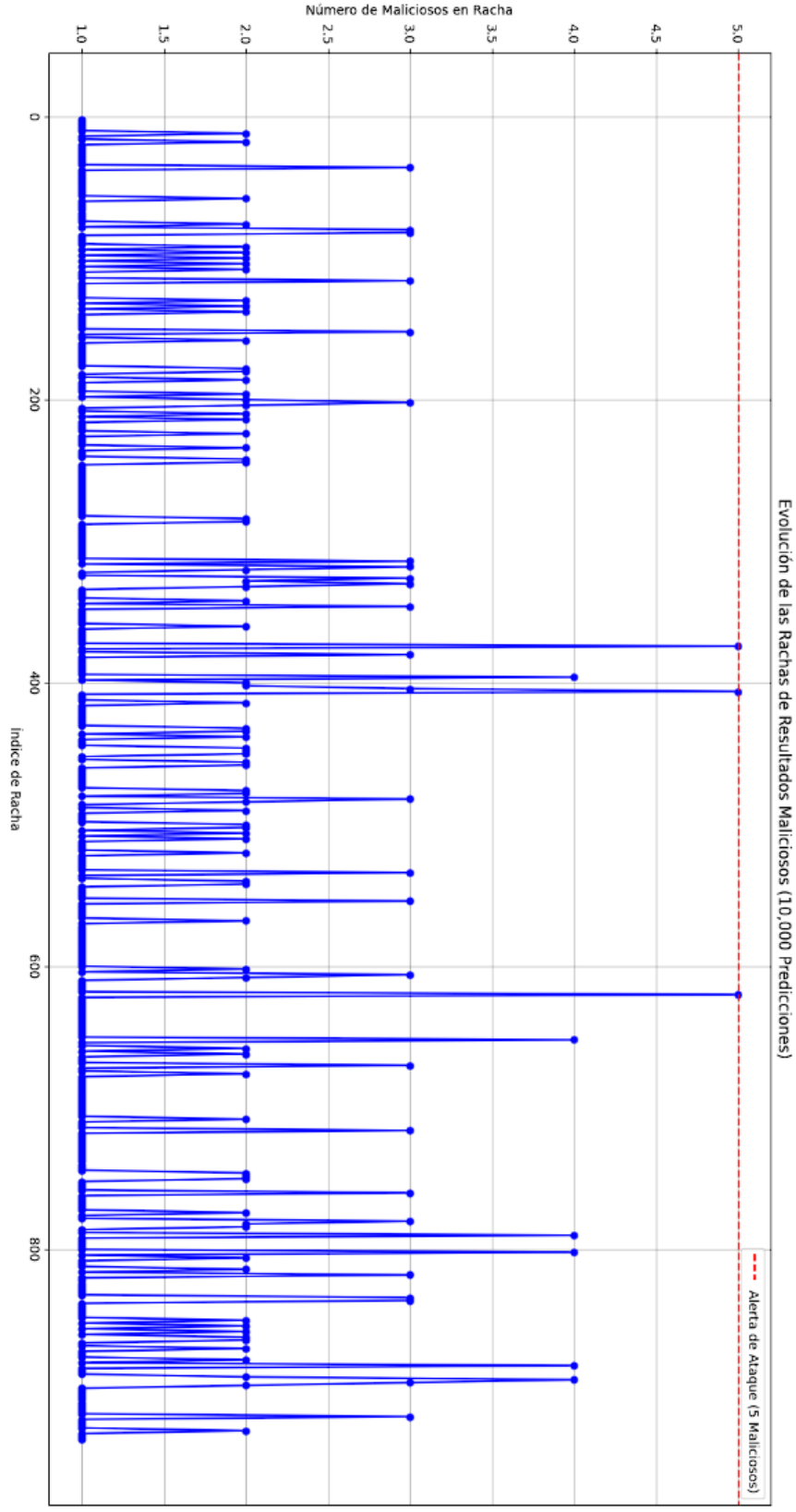



Figura 6.1: Detección de ataques de Denegación de Servicio

# 7

# Lecciones Aprendidas

A lo largo del desarrollo de este trabajo para el desarrollo y evaluación de un modelo de clasificación para la predicción de ataques de Denegación de Servicio, se han identificado varios puntos clave que son importantes para el entendimiento y mejora de los sistemas de detección de ataques. En este capítulo, se presenta un análisis de las lecciones aprendidas durante este proceso.

## 7.1. Importancia del EDA

La utilización del EDA durante este trabajo ha supuesto una mayor comprensión de los datos utilizados para el sistema de detección de ataques de Denegación de Servicio, mediante la investigación de distintos puntos clave, como el análisis de las distribuciones univariantes. Debido al estudio de la distribución de cada una de las variables, se ha identificado que la mayoría de las muestras presentaban una distribución de cola pesada. Por este motivo, se decidió realizar una discretización de las variables numéricas con este problema mediante el uso de árboles de decisión univariantes.

Posteriormente, con el objetivo de obtener un conjunto de datos equilibrado, se procedió al balanceo de carga mediante submuestreo, obteniendo un conjunto de datos de 3.000 datos donde el porcentaje de muestras legítimas y maliciosas era de un 50 % y 50 %, respectivamente. Se eligió este balanceo de carga debido a que es un estándar en muchos proyectos de Ciencia de Datos, sin embargo, como trabajo a futuro podría resultar de interés la experimentación con otros tipos

de balanceos de carga que pudieran ser más adecuados cuando el escenario que se presenta existe un desbalanceo notable en la cantidad de muestras legítimas frente a las muestras maliciosas.

Adicionalmente, para evaluar el grado de dependencia entre las variables de entrada y la variable de salida, se emplearon contrastes de hipótesis. Sin embargo, se observó que el uno de Python limitó el alcance de este contraste, ya que no fue posible aplicar el test exacto de Fisher. Por lo tanto, se considera necesario desarrollar el modelo en otros lenguajes de programación en futuros trabajos para estudiar estos aspectos con mayor detalle.

Asimismo, se procedió a la identificación y eliminación de las variables con mayor capacidad predictiva. Si estas variables no hubieran sido eliminadas, el modelo probablemente habría obtenido resultados más satisfactorios, aunque con un mayor riesgo de sobreajuste.

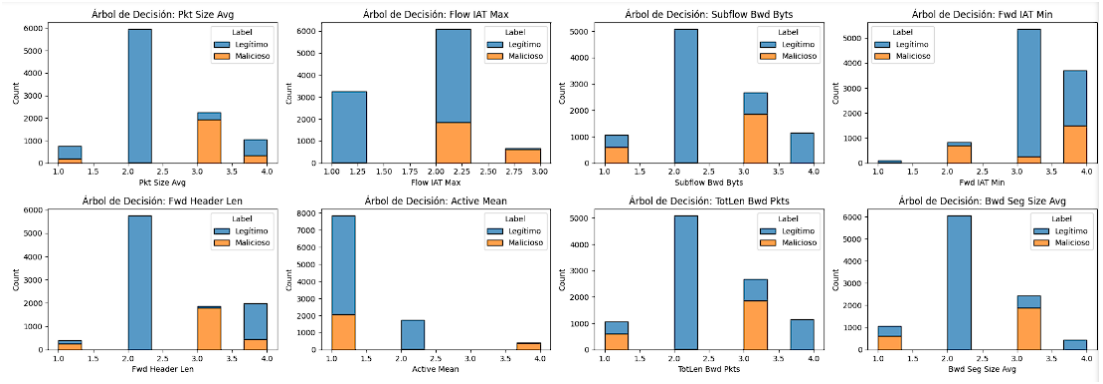
El análisis de multicolinealidad fue otro componente relevante del proceso. Este análisis permitió detectar y gestionar las relaciones lineales entre las variables predictoras, asegurando que el modelo no se viera afectado por redundancias en los datos y, además, reduciendo el número de variables explicativas, que siempre asegura un menor uso de recursos computacionales. La selección de variables resultante de este análisis contribuyó a mejorar la robustez y la interpretabilidad del modelo.

Finalmente, la visualización de la separabilidad entre pares de variables y la variable objetivo proporcionó una visión detallada sobre la capacidad del modelo para diferenciar entre las distintas clases de ataques de Denegación de Servicio. Estas visualizaciones ayudaron a identificar patrones y relaciones que no eran evidentes mediante simples métricas numéricas.

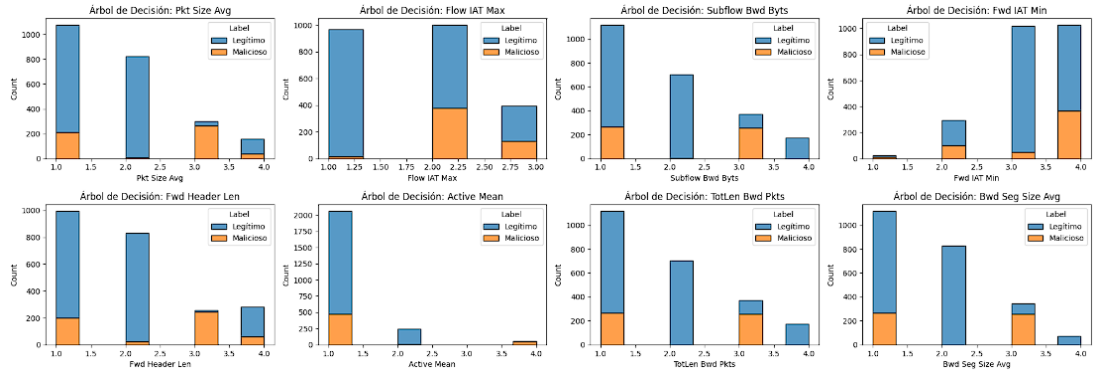
## 7.2. Diferencia de rendimiento del modelo con los datos de entrenamiento y los de la simulación

Uno de los varios puntos importantes a tener en cuenta en futuros trabajos es la discrepancia entre los resultados obtenidos durante el desarrollo del modelo y los resultados de la simulación de un caso real. Específicamente, en este trabajo se ha obtenido que el *F1 score* de la simulación es inferior al obtenido durante el entrenamiento y la validación del modelo.

Al analizar con más detenimiento cuál era la causa del problema, se ha identificado que las distribuciones entre el conjunto utilizado durante el entrenamiento y la validación, y el utilizado en la simulación del despliegue real, difieren en




(a) Distribución de las variables más importantes en el conjunto utilizado durante el modelado y la validación.




(b) Distribución de las variables más importantes en el conjunto utilizado durante la simulación.

Figura 7.1: Comparación de distribuciones

la proporción de clases. En concreto, se ha observado que el conjunto de datos de la simulación tras su preprocesado presenta más muestras con la clase 1 en la mayoría de variables, a diferencia de las variables de los datos preprocesados del conjunto de desarrollo. A continuación, en la figura 7.1 se puede observar una comparativa entre las variables utilizadas por el árbol de decisión durante el desarrollo del modelo y durante la simulación del escenario real.

Este problema radica en la etapa de la discretización de los datos. Durante el desarrollo del modelo, se utilizaron árboles de decisión univariantes para realizar la discretización de cada una de las variables numéricas con distribución de cola pesada. Sin embargo, a la vista de los resultados, puede ser que los árboles univariantes para discretizar las variables numéricas sufrieran un sobreajuste, ajustándose demasiado a los datos de entrenamiento y capturando el ruido en ellos. Esto ha dado como resultado en una menor capacidad de generalización del modelo. Al aplicar manualmente la discretización basada en los puntos de corte de los árboles univariantes en las variables numéricas de la simulación, esta discretización no ha podido adaptarse correctamente a los nuevos datos, lo que

ha provocado un cambio de distribución en las variables de la muestra de simulación, afectando al modelo de clasificación ajustado, que habría aprendido con otra distribución.

Esto explica el menor rendimiento obtenido en la muestra de simulación por el árbol de decisión ajustado para la predicción de ataques maliciosos, en comparación con los resultados obtenidos en la muestra de entrenamiento y de test.

A partir de este problema, se han aprendido varias lecciones. La discretización es una etapa muy importante que puede afectar en el rendimiento del modelo. Una de las posibles soluciones a este problema puede ser la utilización de un modelo que generalice correctamente los datos no vistos. Un modelo que se desempeña notablemente bien en los datos del desarrollo, pero falla en un escenario real no es aplicable en la práctica. Es relevante plantear el uso de técnicas para evitar el sobreajuste, como la validación cruzada.

## 7.3. Estrategia de detección y mitigación de ataques de Denegación de Servicio

Durante el desarrollo de este trabajo, se ha llegado a una conclusión fundamental de cara a cómo desarrollar el modelo de aprendizaje supervisado y cómo este predice las muestras que son ingestadas contribuyendo a un enfoque más sólido y fundamentado en la detección de ataques de Denegación de Servicio.

Este tipo de ataque requieren la saturación de los recursos del servidor mediante el envío de múltiples paquetes maliciosos. Para lograr este objetivo, es necesario que una gran cantidad de estos paquetes logre llegar al servidor y consumir sus recursos. Si el modelo de detección logra identifica y bloquear una parte significativa de estos paquetes, puede prevenir que el ataque tenga éxito y que el servidor pueda manejar la carga parcial de paquetes maliciosos sin colapsar.

Por el contrario, clasificar paquetes legítimos como maliciosos puede llevar a la pérdida parcial de la información transmitida y la interrupción del servicio para usuarios legítimos si esta detección es muy constante. Esto no solo podría afectar a la experiencia del usuario, sino también puede tener consecuencias económicas y operativas.

De este modo, una estrategia que tolere más falsos negativos equilibra la necesidad de mantener la seguridad del sistema con la continuación operativa del servidor y minimiza el riesgo de bloquear paquetes legítimos.

Esta estrategia se ha llevado a cabo mediante la utilización de un umbral superior a la hora de predecir los datos proporcionados al bosque de decisión entrenado.

# 8

# Conclusiones y trabajo a futuro

Tras la realización del trabajo, se exponen las conclusiones obtenidas tras las fases de definición de objetivos, preparación de los datos, el entrenamiento y las pruebas del modelo de aprendizaje automático preparado para la detección de ataques de Denegación de Servicio.

## 8.1. Conclusiones

A lo largo de este trabajo, se han identificado las distintas nuevas amenazas que surgen a partir de la evolución de la tecnología y de los sistemas. Una de estas amenazas son los ataques de Denegación de Servicio, en concreto, los que se materializan mediante el uso del protocolo HTTP. Debido a esto, se ha planteado una posible solución en el campo de la ciberseguridad mediante el uso de la Ciencia de Datos. Esta solución se ha propuesto como una alternativa a las medidas de seguridad existentes en la actualidad y como un reto de combinar el campo de la ciberseguridad con el campo de la Ciencia de Datos.

Durante los capítulos tres y cuatro, se ha llevado a cabo la preparación de los datos para el desarrollo de un modelo de aprendizaje automático diseñado para la detección de ataques de Denegación de Servicio. Durante estas fases, se ha acometido la recopilación de los datos, su preprocesado y un análisis exploratorio con el objetivo de obtener más información sobre las características del conjunto de datos. A lo largo de estos capítulos, se han identificado múltiples problemas relacionados con los datos del conjunto que han llevado a la toma algunas deci-

siones para poder transformar de forma satisfactoria los datos y que estuvieran preparados para su modelado. Entre estos problemas, se destaca la distribución de cola pesada que presentaba la mayoría de variables numéricas del conjunto de datos. Como solución a este problema, al principio se propuso la aplicación de la transformación logarítmica. Sin embargo, debido a que la distribución de cola pesada persistía, se decidió realizar una discretización a cada una de las variables numéricas con este problema.

Adicionalmente, durante el capítulo cinco, se procedió con el entrenamiento de distintos modelos de aprendizaje automático para predecir ataques de Denegación de Servicio. En particular, se ha experimentado con un árbol de decisión y con un *Random Forest*. La evaluación de estos modelos ha permitido cumplir con los objetivos establecidos al principio del trabajo, resaltando el alto rendimiento de ambos modelos durante el entrenamiento y la validación. Sin embargo, debido a que el árbol de decisión presenta una complejidad, un gasto de tiempo y de recursos menor frente al *Random Forest*, se ha elegido este primero como modelo para el desarrollo del sistema de detección.

Finalmente, se ha realizado una simulación de un escenario real en el cual los datos son entregados al modelo de manera secuencial, imitando en la medida de lo posible el flujo de tráfico normal, con el objetivo de evaluar el modelo seleccionado. Adicionalmente, se ha propuesto una posible aplicación del sistema de detección en el cual se identifican paquetes maliciosos consecutivos y se determina la existencia de un ataque de Denegación de Servicio. No obstante, tras observar los resultados obtenidos tras la simulación, se ha observado que el rendimiento del modelo ha disminuido respecto a la etapa de entrenamiento y validación. Este problema ha llevado a cabo el planteamiento de distintas líneas de trabajo futuro que se expondrán a continuación en la sección 7.2.

## 8.2. Líneas de trabajo a futuro

A continuación, se sugieren algunas líneas de trabajos a futuro que podrían mejorar el rendimiento del modelo de aprendizaje automático para la detección de ataques de Denegación de Servicio. Estas líneas se basan en las lecciones aprendidas explicadas anteriormente en el capítulo 7.

Esto incluye explorar diferentes técnicas de discretización que generalicen mejor los datos no vistos. Se ha observado que la discretización es una fase muy importante que afecta al rendimiento del modelo. Es por ello, que se propone investigar otras técnicas en lugar de los árboles de decisión univariantes para lograr este objetivo.

Otra línea de trabajo a futuro es el uso de otros tipos de balanceos de carga. Si bien balancear la carga con el 50 % asignado a las muestras legítimas y el

otro 50 % a las muestras maliciosas es una técnica bastante utilizada, podría no ser la más adecuada en escenarios desequilibrados. Es por ello, que utilizar otras técnicas más avanzadas podría mejorar el rendimiento y fiabilidad del modelo de aprendizaje automático.

Por otro lado, se propone la implementación del modelo mediante diferentes lenguajes de programación con el objetivo de proporcionar un mayor alcance y flexibilidad en el análisis de las variables del conjunto de datos. Los lenguajes como R o Julia, podrían ofrecer librerías y funciones adicionales para el análisis estadístico y una optimización del modelo de clasificación.

Finalmente, se propone la evaluación periódica del modelo en escenarios reales con el objetivo de adaptar el modelo de clasificación a las condiciones cambiantes del entorno en producción y mejorar su rendimiento con nuevos datos.

# Bibliografía


[1] A. F. Isabel and I. M. De Diego, *Ciencia de datos para la ciberseguridad*. Ra-Ma Editorial, 2020.

[2] A. R. Garnacho, "Panorama actual de la ciberseguridad," *Economía industrial*, no. 410, pp. 13–26, 2018.

[3] T. Bourke, *Server load balancing*. .°'Reilly Media, Inc.", 2001.

[4] J. D. Kelleher and B. Tierney, *Ciencia de datos: La serie de conocimientos esenciales de MIT Press*. Ediciones UC, 2021.

[5] K. M. Elleithy, D. Blagovic, W. K. Cheng, and P. Sideleau, "Denial of service attack techniques: analysis, implementation and comparison," 2005.

[6] D. Narváez, C. Romero, and M. Núñez, "Evaluación de ataques de denegación de servicio dos y ddos, y mecanismos de protección," *GEEKS DECC-REPORTS*, vol. 2, no. 1, 2010.

[7] RSnake and J. Kinsella. Slowloris HTTP DoS. [Online]. Available: https://web.archive.org/web/20150426090206/http://ha.ckers.org/slowloris

[8] Goldeneye DDos tool in kali linux. Section: Linux-Unix. [Online]. Available: https://www.geeksforgeeks.org/goldeneye-ddos-tool-in-kali-linux/

[9] W. L. Martinez, A. R. Martinez, and J. Solka, *Exploratory data analysis with MATLAB*. Chapman and Hall/CRC, 2017.

[10] E. Camizuli and E. J. Carranza, "Exploratory data analysis (eda)," *The encyclopedia of archaeological sciences*, pp. 1–7, 2018.

[11] J. I. Daoud, "Multicollinearity and regression analysis," in *Journal of Physics: Conference Series*, vol. 949, no. 1. IOP Publishing, 2017, p. 012009.

[12] C. Arana, "Modelos de aprendizaje automoático mediante árboles de decisión," Serie Documentos de Trabajo, Tech. Rep., 2021.

[13] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90.

[14] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[15] University of New Brunswick. IDS 2018 intrusion CSVs (CSE-CIC-IDS2018). [Online]. Available: https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv

[16] IDS 2018 | datasets | research | canadian institute for cybersecurity | UNB. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[17] Using decision trees to bin numerical vs categorical variables - CleverTap. [Online]. Available: https://clevertap.com/blog/numerical-vs-categorical-variables-decision-trees/

[18] A. Dubey. Discretisation using decision trees. [Online]. Available: https://towardsdatascience.com/discretisation-using-decision-trees-21910483fa4b

[19] H.-Y. Kim, "Statistical notes for clinical researchers: Chi-squared test and fisher's exact test," *Restorative dentistry & endodontics*, vol. 42, no. 2, p. 152, 2017.

[20] M. Cuesta, C. Lancho, A. Fernández-Isabel, E. L. Cano, and I. Martín De Diego, "Cs-viz: Class separability visualization for high-dimensional datasets," *Applied Intelligence*, vol. 54, no. 1, pp. 924–946, 2024.

# Apéndice

# A

# Código utilizado durante el trabajo

## A.1. Librerías utilizadas

```
# Tratamiento de datos
# ==============================================================================
import pandas as pd
import numpy as np

# Gráficos
# ==============================================================================
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
pd.set_option("display.max_rows", None)
pd.set_option("display.max_columns", None)

# Preprocesado y modelado
# ==============================================================================
from sklearn.tree import DecisionTreeClassifier, plot_tree
from scipy.stats import ttest_ind, chi2_contingency
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

## A.2. Obtención y visualización de los datos

```
dataframe = pd.read_csv("C:/Users/rodri/Documentos/Universidad/TFG/Dataset/02-15-2018.csv")
datos, caso_real = train_test_split(dataframe, test_size=0.1, stratify=dataframe["Label"], random_state=12)

print("Número de filas: %s" % str((datos.shape[0])))
print("Número de columnas: %s" % str((datos.shape[1])))

# Número de muestras por etiqueta
# ================================================================================
num_etiqueta = datos["Label"].value_counts()
print(num_etiqueta)
print("\n-------------------\n")
porcentaje_etiqueta = (num_etiqueta / len(datos))
print(porcentaje_etiqueta)

# Gráfico de barras
# ================================================================================
colores = ["green", "red", "orange"]
plt.bar(num_etiqueta.index, num_etiqueta.values, color = colores)

for i, etiqueta in enumerate (num_etiqueta.index):
    plt.text(i, num_etiqueta[etiqueta]+5, str(num_etiqueta[etiqueta]), ha = "center", va = "bottom")

plt.xticks(range(len(num_etiqueta.index)), num_etiqueta.index)
plt.title("Número de registros según su categoría", fontweight="bold")
plt.xlabel("Tráfico", fontweight="bold")
plt.ylabel("Número de registros", fontweight="bold")
plt.show()
```

## A.3. Preprocesado

```
# Eliminación de variables irrelevantes
# ================================================================================
datos.drop("Timestamp", axis=1, inplace = True)

# Eliminación de muestras irrelevantes
# ================================================================================
datos = datos[datos["Dst Port"] == 80]

# Eliminación de muestras duplicadas
# ================================================================================
datos = datos.drop_duplicates()

# Eliminación de variables constantes
# ================================================================================
var_cons = datos.columns[datos.nunique() == 1]
```

```
datos.drop(var_cons, axis=1, inplace = True)

# Eliminación de valores faltantes
# ================================================================================
datos = datos.replace(["Infinity", "infinity"], np.inf)
datos = datos.replace([np.inf, -np.inf], np.nan)

# Eliminación valores erróneos
# ================================================================================
colum_neg = [col for col in datos.select_dtypes(include=['number']).columns if
↪  (datos[col]<0).any()]
for col in colum_neg:
    datos = datos[datos[col]>=0]

# Transformación de la variable objetivo a binario
# ================================================================================
datos["Label"] = datos["Label"].apply(lambda x: "Legítimo" if x == "Benign"
↪  else "Malicioso")
print(datos["Label"].value_counts()) # Recuento
```

## A.4. Análisis Exploratorio de los Datos

```
def distribuciones(datos):
    columnas = datos.columns.drop("Label")
    variables_discretas = []

    for columna in columnas:
        n_valores = datos[columna].nunique()
        if n_valores <= 10:
            variables_discretas.append(columna)

    variables_continuas = datos.columns.drop(variables_discretas + ["Label"])
    return variables_continuas, variables_discretas

y = datos["Label"]
y = y.map({"Legítimo": 0, "Malicioso": 1})

# Transformaciones logarítmicas y discretizacion
# ================================================================================
variables_continuas, variables_discretas = distribuciones(datos)

for columna in variables_continuas:
    if columna in ["Fwd IAT Tot", "Fwd IAT Mean", "Fwd IAT Std", "Fwd IAT
    ↪  Max"]:
        datos[columna] = np.log(datos[columna] + 0.5)
    else:
        X = datos[[columna]]
        arbol_discretizador = DecisionTreeClassifier(max_depth=2,
        ↪  random_state=12)
        arbol_discretizador.fit(X, y)
        plt.figure(figsize=(20,10))
```

```
28          plot_tree(arbol_discretizador, filled=True, feature_names=[columna],
            ↪  class_names=np.unique(y).astype(str))
29          plt.title(f"Árbol de decisión para la columna {columna}")
30          plt.show()
31          nodos_hoja = arbol_discretizador.apply(X)
32          etiquetas_unicas = np.unique(nodos_hoja)
33          mapeo_hojas = {etiqueta: i+1 for i, etiqueta in
            ↪  enumerate(etiquetas_unicas)}
34          datos[columna] = [mapeo_hojas[nodo] for nodo in nodos_hoja]
35
36  variables_continuas, variables_discretas = distribuciones(datos)
37
38  # Visualización de distribuciones
39  # ================================================================================
40  def boxplot(datos, variables_continuas):
41      plt.figure(figsize=(10, len(variables_continuas) * 4))
42
43      for i, columna in enumerate(variables_continuas, 1):
44          plt.subplot(len(variables_continuas), 1, i)
45          sns.boxplot(x="Label", y=columna, data=datos)
46          plt.title(f"Boxplot de {columna} diferenciado por Label")
47
48      plt.tight_layout()
49      plt.show()
50
51  def histograma (datos, variables_discretas):
52      plt.figure(figsize=(10, len(variables_discretas) * 4))
53      for columna in variables_discretas:
54          datos_agrupados = datos.groupby([columna,
            ↪  "Label"]).size().unstack(fill_value=0)
55
56          datos_agrupados.plot(kind="bar", stacked=True, figsize=(10, 6))
57
58          plt.xlabel(f"{columna}")
59          plt.ylabel("Número de Muestras")
60          plt.title(f"Distribución de Benigno y Malicioso en {columna}")
61          plt.legend(title="Label")
62          plt.show()
63
64  boxplot(datos, variables_continuas)
65  histograma(datos, variables_discretas)
66
67  # Ajuste de las variables discretizadas
68  # ================================================================================
69  datos["Flow Duration"] = datos["Flow Duration"].replace([1, 2], 2)
70  datos["Flow Byts/s"] = datos["Flow Byts/s"].replace([3, 4], 3)
71  datos["Flow Pkts/s"] = datos["Flow Pkts/s"].replace([1, 2], 1)
72  datos["Flow IAT Mean"] = datos["Flow IAT Mean"].replace([3, 4], 3)
73  datos["Flow IAT Std"] = datos["Flow IAT Std"].replace([3, 4], 3)
74  datos["Flow IAT Max"] = datos["Flow IAT Max"].replace([3, 4], 3)
75  datos["Bwd IAT Mean"] = datos["Bwd IAT Mean"].replace([2, 3], 2)
76  datos["Bwd IAT Std"] = datos["Bwd IAT Std"].replace([2, 3], 2)
77  datos["Bwd IAT Max"] = datos["Bwd IAT Max"].replace([2, 3], 2)
```

```
78  datos["Bwd Pkts/s"] = datos["Bwd Pkts/s"].replace([1, 2], 1)
79  datos["Down/Up Ratio"] = datos["Down/Up Ratio"].replace([2, 3], 2)
80  datos["Init Fwd Win Byts"] = datos["Init Fwd Win Byts"].replace([3, 4], 3)
81  datos["Active Mean"] = datos["Active Mean"].replace([3, 4], 4)
82  datos["Active Std"] = datos["Active Std"].replace([3, 4], 3)
83  datos["Active Max"] = datos["Active Max"].replace([3, 4], 3)
84  datos["Active Min"] = datos["Active Min"].replace([3, 4], 4)
85  datos["Idle Mean"] = datos["Idle Mean"].replace([3, 4], 4)
86  datos["Idle Std"] = datos["Idle Std"].replace([3, 4], 3)
87  datos["Idle Max"] = datos["Idle Max"].replace([3, 4], 4)
88  datos["Idle Min"] = datos["Idle Min"].replace([1, 2], 1)
89
90  histograma(datos, variables_discretas)
91
92  datos.drop("Fwd Pkt Len Min", axis=1, inplace = True)
93  datos.drop("Bwd Pkt Len Min", axis=1, inplace = True)
94  datos.drop("Fwd PSH Flags", axis=1, inplace = True)
95  datos.drop("FIN Flag Cnt", axis=1, inplace = True)
96  datos.drop("SYN Flag Cnt", axis=1, inplace = True)
97  datos.drop("PSH Flag Cnt", axis=1, inplace = True)
98  datos.drop("ACK Flag Cnt", axis=1, inplace = True)
99  datos.drop("URG Flag Cnt", axis=1, inplace = True)
100
101 variables_continuas, variables_discretas = distribuciones(datos)
102
103 print("Número de filas: %s" % str((datos.shape[0])))
104 print("Número de columnas: %s" % str((datos.shape[1])))
105
106 # Balanceo de carga
107 # ========================================================================
108 good_datos = datos[datos["Label"] == "Legítimo"].sample(1500, random_state=12)
109 bad_datos = datos[datos["Label"] == "Malicioso"].sample(1500, random_state=12)
110 muestras = pd.concat([good_datos, bad_datos])
111 muestras = muestras.sample(frac=1, random_state = 12)
112 print(muestras["Label"].value_counts())
113
114 # Contrastes de Hipótesis
115 # ========================================================================
116 variables_fisher = []
117
118 for variable in variables_discretas:
119     tabla_contingencia = pd.crosstab(muestras["Label"], muestras[variable])
120
121     print(f"Tabla de contingencia para {variable}:\n")
122     print(tabla_contingencia)
123     print("\n")
124
125     if (tabla_contingencia <= 5).any().any():
126         variables_fisher.append(variable)
127
128 for variable in variables_fisher:
129     print(f"{variable}\n")
130
```

```
131 resultados = []
132
133 for variable in muestras.columns.tolist():
134     a = good_datos[variable]
135     b = bad_datos[variable]
136
137     p_valor = None
138
139     if variable in variables_continuas:
140         t_stat, p_valor = ttest_ind(a, b)
141     elif variable in variables_discretas:
142         tabla_contingencia = pd.crosstab(muestras[variable],
            ↪  muestras["Label"])
143         if variable in variables_fisher:
144             chi2, p_valor, dor, _ = chi2_contingency(tabla_contingencia,
                ↪  correction=True)
145         else:
146             chi2, p_valor, dor, _ = chi2_contingency(tabla_contingencia)
147
148     if p_valor is not None:
149         resultados.append({"Variable": variable, "P-valor":p_valor})
150
151 filas_nueva_tabla = []
152
153 num_filas = len(resultados) // 3 + (1 if len(resultados) % 3 > 0 else 0)
154
155 for i in range(0, len(resultados), 3):
156     fila_actual = resultados[i:i+3]
157
158     fila_nueva = []
159     for resultado in fila_actual:
160         fila_nueva.extend([resultado["Variable"], resultado["P-valor"]])
161
162     while len(fila_nueva) < 6:
163         fila_nueva.extend([None, None])
164
165     filas_nueva_tabla.append(fila_nueva)
166
167 columnas = ["Variable 1", "P-valor 1", "Variable 2", "P-valor 2", "Variable
    ↪  3", "P-valor 3"]
168 df_nueva_tabla = pd.DataFrame(filas_nueva_tabla, columns=columnas)
169
170 print(df_nueva_tabla)
171
172 # Eliminación de variables hiperpredictoras
173 # ============================================================================
174 hiperpredictoras = ["Fwd Seg Size Min", "Init Fwd Win Byts", "Init Bwd Win
    ↪  Byts", "Flow IAT Min"]
175 muestras.drop(hiperpredictoras, axis=1, inplace=True)
176
177 # VIF
178 # ============================================================================
179
```

```
variables_continuas, variables_discretas = distribuciones(muestras)

def calc_vif(X):
    vif = pd.DataFrame()
    vif["variables"] = X.columns
    vif["VIF"] = [variance_inflation_factor(X.values, i) for i in
    ↪  range(X.shape[1])]
    return vif

# Iteración 1
X = muestras[variables_discretas]
calc_vif(X)

# Iteración 2
X = X.drop("Flow IAT Std", axis=1)
calc_vif(X)

# Iteración 3
X = X.drop("Flow Duration", axis=1)
calc_vif(X)

muestras.drop(["Flow IAT Std"] + ["Flow Duration"], axis=1, inplace=True)

print("Número de filas: %s" % str((muestras.shape[0])))
print("Número de columnas: %s" % str((muestras.shape[1])))
```

## A.5. Modelado

```
del muestras
muestras_maliciosas = datos[datos["Label"] == "Malicioso"].sample(n=2486,
↪  random_state=12)
muestras_legitimas = datos[datos["Label"] == "Legítimo"].sample(n=7514,
↪  random_state=12)
muestras = pd.concat([muestras_maliciosas, muestras_legitimas])
muestras = muestras.sample(frac=1, random_state=12)

train, test = train_test_split(muestras, test_size = 0.2,
↪  stratify=muestras["Label"], random_state=12)

train_maliciosos = train[train["Label"] == "Malicioso"].sample(n=1500,
↪  random_state=12)
train_legitimos = train[train["Label"] == "Legítimo"].sample(n=1500,
↪  random_state=12)
train = pd.concat([train_maliciosos, train_legitimos])
train = train.sample(frac=1, random_state=12)

train.drop(["Fwd Seg Size Min", "Init Fwd Win Byts", "Init Bwd Win Byts",
↪  "Flow IAT Min"], axis = 1, inplace = True)
test.drop(["Fwd Seg Size Min", "Init Fwd Win Byts", "Init Bwd Win Byts", "Flow
↪  IAT Min"], axis = 1, inplace = True)
variables_continuas, variables_discretas = distribuciones(train)
```

```
train.drop(["Flow IAT Std", "Flow Duration"], axis = 1, inplace = True)
test.drop(["Flow IAT Std", "Flow Duration"], axis = 1, inplace = True)

X_train = train.drop(["Label"], axis=1)
X_test = test.drop(["Label"], axis=1)
y_train = train["Label"]
y_test = test["Label"]

# Árbol de decisión
# ==============================================================================

# Grid Search
dt = DecisionTreeClassifier(random_state=12)

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': range(1, 10),
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

CV_grid = GridSearchCV(estimator = dt, param_grid = param_grid, cv=5)

CV_grid.fit(X_train, y_train)

print(f"Mejores parámetros: {CV_grid.best_params_}")
print(f"Exactitud: {CV_grid.best_score_}")

results = CV_grid.cv_results_
depths = [params['max_depth'] for params in results['params']]
scores = results['mean_test_score']

# Representación gráfica de las exactitudes
plt.figure(figsize=(10, 6))
plt.plot(depths, scores, marker='o')
plt.xlabel('Profundidad del Árbol')
plt.ylabel('Puntuación Media de Validación')
plt.title('Puntuación Media de Validación vs Profundidad del Árbol de
↪  Decisión')
plt.grid(True)
plt.show()

# Entrenamiento del modelo
arbol = DecisionTreeClassifier(criterion = 'gini', max_depth = 4,
↪  min_samples_leaf = 1, min_samples_split = 2, random_state=12)
arbol.fit(X_train, y_train)
plt.figure(figsize=(20,10))
plot_tree(arbol, filled=True, rounded=True, feature_names=X_train.columns)
plt.title('Árbol de decisión')
plt.show()

# Umbrales de predicción
pred_probs = arbol.predict_proba(X_train)[:, 1]
```

```
68  threshold = []
69  F1 = []
70
71  for u in np.arange(0.05, 1.05, 0.05):
72      threshold.append(u)
73      y_pred = (pred_probs >= u).astype(int)
74      y_pred_labels = np.where(y_pred == 1, 'Malicioso', 'Legítimo')
75      f1 = f1_score(y_train, y_pred_labels, pos_label='Malicioso')
76      F1.append(f1)
77      print(f'Umbral: {u:.2f}, F1-Score: {f1:.4f}')
78
79  plt.plot(threshold, F1, marker='o')
80  plt.xlabel('IoU Threshold')
81  plt.ylabel('F1 - Score')
82  plt.title('F1-Score vs IoU Threshold')
83  plt.grid(True)
84  plt.show()
85
86  # Validación
87  threshold = 0.7
88  pred = arbol.predict_proba(X_test)[:, 1]
89  y_pred = (pred >= threshold).astype(int)
90  y_pred_labels = np.where(y_pred == 1, 'Malicioso', 'Legítimo')
91  print("F1 score: ", f1_score(y_test, y_pred_labels, pos_label='Malicioso'))
92
93  y_pred = arbol.predict(X_test)
94  print('Accuracy score: ',accuracy_score(y_test, y_pred))
95
96  # Random Forest
97  # ================================================================================
98
99  # Grid Search
100 rf = RandomForestClassifier(random_state = 12)
101
102 param_grid ={'n_estimators':[50, 100, 150],
103              'max_features':[5, 10, 20],
104              'max_depth':[None, 3, 10, 20],
105              'criterion':['gini', 'entropy']
106 }
107
108 CV_grid = GridSearchCV(estimator = rf, param_grid = param_grid, cv=5)
109
110 CV_grid.fit(X_train, y_train)
111
112 print(f"Mejores parámetros: {CV_grid.best_params_}")
113 print(f"Exactitud: {CV_grid.best_score_}")
114
115 # Entrenamiento del modelo
116 bosque = RandomForestClassifier(criterion = 'entropy', max_depth = None,
    ↪  max_features = 5, n_estimators = 50, random_state=12)
117 bosque.fit(X_train, y_train)
118
119 # Umbrales de predicción
```

```
pred_probs = bosque.predict_proba(X_train)[:, 1]
threshold = []
F1 = []

for u in np.arange(0.05, 1.05, 0.05):
    threshold.append(u)
    y_pred = (pred_probs >= u).astype(int)
    y_pred_labels = np.where(y_pred == 1, 'Malicioso', 'Legítimo')
    f1 = f1_score(y_train, y_pred_labels, pos_label='Malicioso')
    F1.append(f1)
    print(f'Umbral: {u:.2f}, F1-Score: {f1:.4f}')

plt.plot(threshold, F1, marker='o')
plt.xlabel('IoU Threshold')
plt.ylabel('F1 - Score')
plt.title('F1-Score vs IoU Threshold')
plt.grid(True)
plt.show()

# Validación
threshold = 0.65
pred = bosque.predict_proba(X_test)[:, 1]
y_pred = (pred >= threshold).astype(int)
y_pred_labels = np.where(y_pred == 1, 'Malicioso', 'Legítimo')
print("F1 score: ", f1_score(y_test, y_pred_labels, pos_label='Malicioso'))

y_pred = bosque.predict(X_test)
print('Accuracy score: ',accuracy_score(y_test, y_pred))
```

## A.6. Nuevos datos

```
descarte, simulacion = train_test_split(caso_real, test_size=0.1,
↪  stratify=caso_real["Label"], random_state=3)
simulacion = simulacion[simulacion["Dst Port"] == 80]
simulacion["Label"] = simulacion["Label"].apply(lambda x: "Legítimo" if x ==
↪  "Benign" else "Malicioso")
caso_real = []
for index, muestra in simulacion.iterrows():
    row_df = muestra.to_frame().T
    caso_real.append(row_df)

resultados = []
etiquetas = []
columnas_a_eliminar = ["Timestamp", "Dst Port", "Protocol", "Bwd PSH Flags",
                       "Fwd URG Flags", "Bwd URG Flags", "Pkt Len Min", "CWE
                       ↪  Flag Count",
                       "Fwd Byts/b Avg", "Fwd Pkts/b Avg", "Fwd Blk Rate Avg",
                       "Bwd Byts/b Avg", "Bwd Pkts/b Avg", "Bwd Blk Rate Avg",
                       "Bwd Pkt Len Min", "Fwd Pkt Len Min", "Fwd PSH Flags",
                       "FIN Flag Cnt", "SYN Flag Cnt", "PSH Flag Cnt", "ACK
                       ↪  Flag Cnt",
```

```
                        "URG Flag Cnt", "Fwd Seg Size Min", "Init Fwd Win
                        ↪  Byts",
                        "Init Bwd Win Byts", "Flow IAT Min", "Flow Duration",
                        "Flow IAT Std"]
intervalos = {
    "Tot Fwd Pkts": [-np.inf, 2.50, 3.50, 15.50, np.inf],
    "Tot Bwd Pkts": [-np.inf, 1.50, 3.50, 4.50, np.inf],
    "TotLen Fwd Pkts": [-np.inf, 19.00, 214.00, 426.50, np.inf],
    "TotLen Bwd Pkts": [-np.inf, 53.00, 661.50, 974.00, np.inf],
    "Fwd Pkt Len Max": [-np.inf, 10.50, 214.00, 426.50, np.inf],
    "Fwd Pkt Len Mean": [-np.inf, 8.01, 55.19, 141.76, np.inf],
    "Fwd Pkt Len Std": [-np.inf, 0.07, 97.76, 144.81, np.inf],
    "Bwd Pkt Len Max": [-np.inf, 53.00, 971.50, 975.00, np.inf],
    "Bwd Pkt Len Mean": [-np.inf, 5.11, 165.31, 324.15, np.inf],
    "Bwd Pkt Len Std": [-np.inf, 19.11, 330.77, 561.58, np.inf],
    "Flow Byts/s": [-np.inf, 80.86, 27134.98, 1585003.62, np.inf],
    "Flow Pkts/s": [-np.inf, 0.07, 0.18, 154.39, np.inf],
    "Flow IAT Mean": [-np.inf, 7118.43, 6024116.50, 18495743.50, np.inf],
    "Flow IAT Max": [-np.inf, 44389.00, 12397799.50, 53772394.00, np.inf],
    "Fwd IAT Min": [-np.inf, 0.50, 83.50, 212.50, np.inf],
    "Bwd IAT Tot": [-np.inf, 187.50, 45178.00, 109217628.00, np.inf],
    "Bwd IAT Mean": [-np.inf, 269.42, 10378389.50, 11577567.50, np.inf],
    "Bwd IAT Std": [-np.inf, 28918.76, 7150261.00, 10781661.50, np.inf],
    "Bwd IAT Max": [-np.inf, 45111.50, 13076493.50, 71153360.00, np.inf],
    "Bwd IAT Min": [-np.inf, 214.50, 278.50, 20980159.00, np.inf],
    "Fwd Header Len": [-np.inf, 68.00, 134.00, 170.00, np.inf],
    "Bwd Header Len": [-np.inf, 66.00, 98.00, 138.00, np.inf],
    "Fwd Pkts/s": [-np.inf, 0.78, 71.55, 363.86, np.inf],
    "Bwd Pkts/s": [-np.inf, 0.03, 0.05, 47.94, np.inf],
    "Pkt Len Max": [-np.inf, 231.00, 971.50, 973.00, np.inf],
    "Pkt Len Mean": [-np.inf, 29.77, 102.32, 161.71, np.inf],
    "Pkt Len Std": [-np.inf, 146.04, 222.88, 373.17, np.inf],
    "Pkt Len Var": [-np.inf, 21329.14, 49677.47, 139259.33, np.inf],
    "Down/Up Ratio": [-np.inf, 0.50, 2.50, np.inf],
    "Pkt Size Avg": [-np.inf, 34.02, 116.22, 184.81, np.inf],
    "Fwd Seg Size Avg": [-np.inf, 8.01, 55.19, 141.76, np.inf],
    "Bwd Seg Size Avg": [-np.inf, 5.11, 165.31, 324.15, np.inf],
    "Subflow Fwd Pkts": [-np.inf, 2.50, 3.50, 15.50, np.inf],
    "Subflow Fwd Byts": [-np.inf, 19.00, 214.00, 426.50, np.inf],
    "Subflow Bwd Pkts": [-np.inf, 1.50, 3.50, 4.50, np.inf],
    "Subflow Bwd Byts": [-np.inf, 53.00, 661.50, 974.00, np.inf],
    "Fwd Act Data Pkts": [-np.inf, 10.50, 11.50, np.inf],
    "Active Mean": [-np.inf, 11206.50, 3300225.75, 5475217.75, np.inf],
    "Active Std": [-np.inf, 0.35, 1610949.88, 2734172.62, np.inf],
    "Active Max": [-np.inf, 0.50, 6616434.50, 8011653.50, np.inf],
    "Active Min": [-np.inf, 10633.50, 1001249.50, 4047356.50, np.inf],
    "Idle Mean": [-np.inf, 8840425.50, 12019526.00, 22559388.00, np.inf],
    "Idle Std": [-np.inf, 1.06, 3530430.00, 24846924.00, np.inf],
    "Idle Max": [-np.inf, 8276650.00, 12633244.50, 21052106.00, np.inf],
    "Idle Min": [-np.inf, 5003205.00, 5112022.50, 7003462.00, np.inf]
}

def discretizar_columna(columna, intervalos):
```

```
69      discretizada = pd.cut(columna, bins=intervalos, labels=False)+1
70      return discretizada
71
72  # Árbol de decisión
73  # ================================================================================
74  iter = 0
75  for indice, muestra in enumerate(caso_real):
76      muestra = muestra.drop(columnas_a_eliminar, axis=1)
77
78      if (muestra.select_dtypes(include=[np.number]) < 0).any().any():
79          continue
80
81      muestra = muestra.replace(["Infinity", "infinity"], np.inf)
82      muestra = muestra.replace([np.inf, -np.inf], np.nan)
83
84      for columna in ["Fwd IAT Tot", "Fwd IAT Mean", "Fwd IAT Std", "Fwd IAT
        ↪  Max"]:
85          muestra[columna] = np.log(muestra[columna] + 0.5)
86
87      for var, bins in intervalos.items():
88          muestra[var] = discretizar_columna(muestra[var], bins)
89
90      muestra["Flow Byts/s"] = muestra["Flow Byts/s"].replace([3, 4], 3)
91      muestra["Flow Pkts/s"] = muestra["Flow Pkts/s"].replace([1, 2], 1)
92      muestra["Flow IAT Mean"] = muestra["Flow IAT Mean"].replace([3, 4], 3)
93      muestra["Flow IAT Max"] = muestra["Flow IAT Max"].replace([3, 4], 3)
94      muestra["Bwd IAT Mean"] = muestra["Bwd IAT Mean"].replace([2, 3], 2)
95      muestra["Bwd IAT Std"] = muestra["Bwd IAT Std"].replace([2, 3], 2)
96      muestra["Bwd IAT Max"] = muestra["Bwd IAT Max"].replace([2, 3], 2)
97      muestra["Bwd Pkts/s"] = muestra["Bwd Pkts/s"].replace([1, 2], 1)
98      muestra["Down/Up Ratio"] = muestra["Down/Up Ratio"].replace([2, 3], 2)
99      muestra["Active Mean"] = muestra["Active Mean"].replace([3, 4], 4)
100     muestra["Active Std"] = muestra["Active Std"].replace([3, 4], 3)
101     muestra["Active Max"] = muestra["Active Max"].replace([3, 4], 3)
102     muestra["Active Min"] = muestra["Active Min"].replace([3, 4], 4)
103     muestra["Idle Mean"] = muestra["Idle Mean"].replace([3, 4], 4)
104     muestra["Idle Std"] = muestra["Idle Std"].replace([3, 4], 3)
105     muestra["Idle Max"] = muestra["Idle Max"].replace([3, 4], 4)
106     muestra["Idle Min"] = muestra["Idle Min"].replace([1, 2], 1)
107
108     if muestra.isna().any().any():
109         print(muestra)
110         break
111     else:
112         X = muestra.drop("Label", axis=1)
113         y = muestra["Label"]
114         etiquetas.append(y)
115
116         prediccion = arbol.predict_proba(X)[:, 1]
117         y_prediccion = (prediccion >= threshold).astype(int)
118         y_prediccion_labels = np.where(y_prediccion == 1, "Malicioso",
            ↪  "Legítimo")
119         resultados.append(y_prediccion_labels[0])
```

```
120         iter = iter+1
121
122 F1 = f1_score(etiquetas, resultados, pos_label="Malicioso")
123 print(f"F1 score: {F1}")
124 print(iter)
125
126 # Detección de ataques
127 # ================================================================================
128 df = pd.DataFrame(resultados, columns=["Prediccion"])
129
130 df["Malicioso"] = (df["Prediccion"] == "Malicioso").astype(int)
131
132 df["Grupo"] = (df["Malicioso"] != df["Malicioso"].shift(1)).cumsum()
133
134 df_rachas = df[df["Malicioso"] == 1].groupby("Grupo").size()
135
136 ataque_detectado = df_rachas[df_rachas >= 5].any()
137
138 if ataque_detectado:
139     print("Ataque detectado")
140
141 plt.figure(figsize=(20, 10))
142 plt.plot(df_rachas.index, df_rachas.values, marker="o", linestyle="-",
    ↪  color="b", markersize=5)
143 plt.title("Evolución de las Rachas de Resultados Maliciosos (10,000
    ↪  Predicciones)")
144 plt.xlabel("Índice de Racha")
145 plt.ylabel("Número de Maliciosos en Racha")
146 plt.grid(True)
147
148 plt.axhline(y=5, color="r", linestyle="--", label="Alerta de Ataque (5
    ↪  Maliciosos)")
149 plt.legend()
150
151 plt.show()
```