



UNIVERSIDAD  
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA  
GRADUADO EN INGENIERÍA INFORMÁTICA

## **XAI para la Detección Confiable de Intrusiones: Explicabilidad e Interpretabilidad**

### **XAI for Trustworthy Intrusion Detection: Explainability and Interpretability**

Realizado por  
**JAVIER LUQUE RUEDA**

Tutorizado por  
**MARÍA CRISTINA ALCARAZ TELLO**  
**RODRIGO ROMÁN CASTRO**

Departamento  
**LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN**

UNIVERSIDAD DE MÁLAGA  
MÁLAGA,

Fecha defensa:

# Resumen

En un contexto donde el empleo de tecnologías de la información y las comunicaciones sigue experimentando un crecimiento continuo, las organizaciones se enfrentan a un riesgo creciente de sufrir intrusiones maliciosas que podrían comprometer la seguridad y disponibilidad de sus sistemas y datos. Este Trabajo Fin de Grado (TFG) se enfoca en el ámbito de la ciberseguridad, específicamente en la detección de intrusiones, donde la "interpretabilidad" y "explicabilidad" son aspectos clave. La mayoría de los sistemas de detección de intrusiones, basados en modelos de aprendizaje automático, se enfrentan al desafío de volverse cada vez más complejos, lo que dificulta la comprensión de sus decisiones.

El propósito fundamental de este TFG es abordar esta problemática, inspirándose en trabajos previos como el framework XD-IoT (por sus siglas en inglés, "eXplainable Deep learning for Internet of Things", que significa "Aprendizaje profundo explicable para el internet de las cosas") propuesto en investigaciones anteriores [1]. Adaptando este enfoque al campo de la detección de intrusiones, se busca explorar y evaluar diversas técnicas de inteligencia artificial (IA) y aprendizaje automático con el objetivo de detectar amenazas cibernéticas de manera confiable. Además, se pone énfasis en la parte explicativa de los modelos, buscando transparentar los procesos de detección de intrusiones para mejorar la confianza y comprensión de los resultados obtenidos, asegurando que los sistemas de detección de intrusiones no solo sean efectivos, sino también comprensibles y transparentes para los usuarios, lo que facilita la toma de decisiones informadas y la mejora continua de la seguridad de las organizaciones.

**Palabras clave:** ciberseguridad, inteligencia artificial, explicabilidad, interpretabilidad, detección, aprendizaje automático.

# Abstract

In a context where the use of information and communication technologies is constantly increasing, organizations face a growing risk of suffering malicious intrusions that could compromise the security and availability of their systems and data. This dissertation focuses on the field of cybersecurity, specifically on intrusion detection, where "interpretability" and "explainability" are key aspects. Most intrusion detection systems, based on machine learning models, face the challenge of becoming increasingly complex, making it difficult to understand their decisions.

The fundamental purpose of this TFG is to address this issue, drawing inspiration from previous work such as the XD-IoT (eXplainable Deep Learning for Internet of Things) framework proposed in previous research [1]. Adapting this approach to the field of intrusion detection, we seek to explore and evaluate various artificial intelligence and ML techniques with the objective of reliably detecting cyber threats. In addition, emphasis is placed on the explanatory part of the models, seeking to make intrusion detection processes transparent in order to improve the confidence and understanding of the results obtained, ensuring that intrusion detection systems are not only effective, but also understandable and transparent to users, which facilitates informed decision making and continuous improvement in the of organizations security.

**Keywords:** cybersecurity, Artificial Intelligence, explainability, interpretability, detection, machine learning.

# Índice

## Contenido

<b>Resumen.....</b>	<b>1</b>
<b>Abstract .....</b>	<b>1</b>
<b>Índice .....</b>	<b>1</b>
<b>Introducción.....</b>	<b>6</b>
1.1 Motivación .....	8
1.2 Objetivos .....	9
1.3 Metodología de trabajo.....	10
1.4 Organización de la memoria .....	12
<b>Conceptos básicos .....</b>	<b>14</b>
2.1 Detección de intrusiones e inteligencia artificial.....	14
2.2 Aprendizaje automático .....	16
2.3 Explicabilidad.....	21
2.3.1 SHAP .....	25
2.3.2 LIME .....	27
<b>Análisis preliminares .....</b>	<b>31</b>
3.1 Revisión del estado del arte.....	31
3.2 Conjuntos de datos .....	34
3.2.1 Tipos de ataques.....	37
3.3 Preprocesamiento de datos.....	39
3.3.1 Limpieza de datos .....	39
3.3.2 Selección de características.....	40
3.3.3 Normalización de características.....	42
3.4 Algoritmos de aprendizaje automático utilizados .....	43
3.4.1 Logistic Regression .....	43
3.4.2 Perceptrón multicapa .....	45
3.4.3 SVM.....	47
3.4.4 Random Forest .....	49
3.4.5 XGBoost .....	51
3.4.6 LightBM.....	52
3.5 Algoritmos y técnicas de interpretabilidad utilizadas.....	54
3.5.1 Análisis de residuos .....	54
3.5.2 Análisis de influencia .....	56
3.5.3 Análisis de curva ROC .....	59
3.5.4 Análisis de sensibilidad .....	60
3.5.5 Permutación de características .....	62
3.5.6 LIME .....	64
3.5.7 SHAP .....	67
<b>Tecnologías utilizadas.....</b>	<b>72</b>
4.1 Lenguaje de programación .....	72

4.2 Bibliotecas utilizadas .....	73
4.3 Herramientas y plataformas para desarrollo y análisis interactivo .....	74
<b>Implementación .....</b>	<b>75</b>
5.1 Modelos y carga de datos .....	75
5.2 SHAP .....	79
5.3 LIME .....	82
5.4 Técnicas y evaluaciones adicionales.....	83
5.4.1 Permutación de características .....	84
<b>Análisis de resultados .....</b>	<b>86</b>
6.1 DDoS .....	88
6.1.1 Evaluación de modelos .....	88
6.1.2 Evaluación de residuos e influencia .....	91
6.1.3 SHAP .....	94
6.1.4 LIME .....	109
6.1.5 Evaluación permutación y sensibilidad de características .....	112
6.2 PortScan .....	115
6.2.1 Evaluación de modelos .....	115
6.2.2 Evaluación de residuos e influencia .....	117
6.2.3 SHAP .....	120
6.2.4 LIME .....	133
6.2.5 Evaluación permutación y sensibilidad de características .....	136
<b>Conclusiones y Líneas futuras .....</b>	<b>139</b>
7.1 Conclusiones .....	139
7.2 Líneas futuras .....	141
<b>Referencias .....</b>	<b>145</b>
<b>Manual de Instalación .....</b>	<b>151</b>
Requerimientos: .....	151

# Índice de tablas y figuras

Figura 1: Función sigmoide [37] .....	44
Figura 2: Perceptrón multicapa [40] .....	45
Figura 3: Árbol de clasificación para determinar la supervivencia a bordo del RMS Titanic. Fuente: Rigatti, S. J. (2017) [46] .....	50
Figura 4: Gráfica de análisis de residuos para el modelo Logistic Regression para el ataque DDoS .....	92
Figura 5: Gráfica de análisis de influencia para el modelo SVM Poly 2 para el ataque DDoS .....	93
Figura 6: Valores SHAP para el modelo XGBoost para el ataque DDoS .....	95
Figura 7: Valores SHAP para el modelo Random Forest (Clase 0- Benign) para el ataque DDoS .....	98
Figura 8: Valores SHAP para el modelo LightBM para el ataque DDoS .....	100
Figura 9: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque DDoS- Parte1 .....	101
Figura 10: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque DDoS- Parte2 .....	102
Figura 11: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque DDoS- Parte3 .....	102
Figura 12: Análisis SHAP para instancia DDoS para modelo XGBoost en el ataque DDoS- Parte1.ra modelo XGBoost en el ataque DDoS- Parte1 .....	104
Figura 13: Análisis SHAP para instancia DDoS para modelo XGBoost en el ataque DDoS- Parte2 .....	104
Figura 14: Análisis SHAP para instancia DDoS para modelo XGBoost en el ataque DDoS- Parte3 .....	104
Figura 15: Relación de valores SHAP con valores reales (normalizados) para característica Fwd Packet Length Max con el modelo XGBoost para el ataque DDoS .....	107
Figura 16: Tiempo de ejecución para cálculo de valores SHAP para el modelo SVM Poly 2 en el ataque DDoS .....	109
Figura 17: Análisis LIME para instancia Benign para modelo Perceptrón multicapa en el ataque DDoS .....	110

Figura 18: Análisis LIME para instancia DDoS para modelo Perceptrón multicapa en el ataque DDoS.....	111
Figura 19: Permutación en la característica Fwd Packet Length Max para el modelo XGBoost en el ataque DDoS .....	113
Figura 20: Análisis de sensibilidad para la característica Bwd Packet Length Min para el modelo XGBoost con el ataque DDoS.....	114
Figura 21: Gráfica de análisis de residuos para el modelo Logistic Regression para el ataque PortScan.....	118
Figura 22: Gráfica de análisis de influencia para el modelo SVM Poly 2 para el ataque PortScan.....	119
Figura 23: Valores SHAP para el modelo XGBoost para el ataque PortScan	121
Figura 24: Valores SHAP para el modelo LightBM para el ataque PortScan .	123
Figura 25: Valores SHAP para el modelo SVM Lineal para el ataque PortScan .....	124
Figura 26: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque PortScan- Parte1 .....	127
Figura 27: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque PortScan- Parte1 .....	127
Figura 28: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque PortScan- Parte3.....	127
Figura 29: Análisis SHAP para instancia PortScan para modelo XGBoost en el ataque PortScan- Parte1 .....	129
Figura 30: Análisis SHAP para instancia PortScan para modelo XGBoost en el ataque PortScan- Parte2.....	130
Figura 31: Análisis SHAP para instancia PortScan para modelo XGBoost en el ataque PortScan- Parte3.....	130
Figura 32: Relación de valores SHAP con valores reales (normalizados) para característica Fwd Packet Length Mean con el modelo XGBoost para el ataque PortScan.....	132
Figura 33: Análisis LIME para instancia Benign para modelo Perceptrón multicapa en el ataque PortScan.....	133
Figura 34: Análisis LIME para instancia PortScan para modelo Perceptrón multicapa en el ataque PortScan.....	135

Figura 35: Permutación en la característica Fwd Packet Length Mean para el modelo XGBoost en el ataque PortScan ..... 136

Figura 36: Análisis de sensibilidad para la característica Fwd Packet Length Mean para el modelo XGBoost con el ataque PortScan ..... 137

Tabla 1: Resultados de Evaluación de Modelos para Detección de Ataques DDoS en el Dataset CICIDS-2017 ..... 88

Tabla 2: Resumen de Resultados de la Matriz de Confusión para la Detección de Ataques DDoS..... 90

Tabla 3: Resultados de Evaluación de Modelos para Detección de Ataques PortScan en el Dataset CICIDS-2017 ..... 115

Tabla 4: Resumen de Resultados de la Matriz de Confusión para la Detección de Ataques PortScan ..... 116



# Lista de Acrónimos

**TFG** Trabajo Fin de Grado

**IA** Inteligencia Artificial

**XD-IoT** eXplainable Deep learning for Internet of Things

**XAI** eXplainable Artificial Intelligence

**DDoS** Distributed Denial of Service

**DoS** Denial of Service

**IDS** Intrusion Detection System

**HIDS** Host-based Intrusion Detection System

**NIDS** Network-based Intrusion Detection System

**ML** Machine Learning

**SVM** Support Vector Machines

**k-NN** k-Nearest Neighbors

**MLP** Multilayer Perceptron

**LGB** LightBM (Light Gradient Boosting Machine)

**SHAP** SHapley Additive exPlanations

**LIME** Local Interpretable Model-agnostic Explanations

**DNS** Domain Name System

**HTTPS** Hypertext Transfer Protocol Secure

**XGB** eXtreme Gradient Boosting

**SQL** Structured Query Language

**TLS** Transport Layer Security

**SSL** Secure Sockets Layer

**CSV** Comma-Separated Values

**RBF** Radial Basis Function

**CART** Classification and Regression Trees

**RAM** Random Access Memory

**ROC** Receiver Operating Characteristic

**AUC** Area Under the Curve

**NaN** Not a Number

**TN** True Negative

**TP** True Positive

**FN** False Negative

**FP** False Positive

**TPR** True Positive Rate

**FPR** False Positive Rate

# 1

## Introducción

En la actualidad, vivimos inmersos en una era digital donde el vertiginoso crecimiento de las tecnologías de la información y la comunicación ha transformado radicalmente nuestra manera de relacionarnos con el mundo digital que nos rodea. Este progreso tecnológico ha generado una amplia gama de beneficios que han optimizado nuestras vidas y actividades cotidianas.

Sin embargo, este panorama también ha propiciado un incremento sustancial en las amenazas cibernéticas y las intrusiones maliciosas que representan una seria amenaza para la seguridad y la integridad de los sistemas informáticos y los datos sensibles tanto de organizaciones como de individuos. La interconexión global impulsada por la expansión de las tecnologías digitales ha creado un escenario donde la vulnerabilidad a ataques cibernéticos se ha vuelto una preocupación constante. La sofisticación creciente de los ciberdelincuentes y la diversificación de sus métodos han dado lugar a un aumento en la frecuencia y gravedad de los incidentes de seguridad informática.

Desde ataques de ransomware hasta intrusiones en redes corporativas [2], las amenazas cibernéticas representan un desafío constante que requiere una respuesta proactiva y eficaz para salvaguardar la información confidencial y los activos digitales. En este contexto, la protección de la ciberseguridad se convierte en una prioridad estratégica para organizaciones de todos los

tamaños y sectores, así como para usuarios individuales que dependen cada vez más de entornos digitales seguros. La necesidad de implementar medidas preventivas y soluciones avanzadas para detectar, prevenir y mitigar posibles ataques cibernéticos se vuelve imperativa en un entorno donde la información es uno de los activos más valiosos y vulnerables. Por lo tanto, es crucial adoptar un enfoque integral hacia la ciberseguridad que combine tecnología avanzada, buenas prácticas de gestión de riesgos y concienciación sobre seguridad informática para hacer frente a las crecientes amenazas cibernéticas y proteger la integridad de los sistemas y datos en un mundo digital cada vez más interconectado.

En este escenario digital, la detección de intrusiones desempeña un rol esencial en la defensa contra las crecientes amenazas cibernéticas al posibilitar la identificación temprana y la respuesta proactiva ante actividades sospechosas o maliciosas que acechan en los entornos informáticos [3]. La evolución constante de los sistemas de detección de intrusiones hacia soluciones cada vez más avanzadas y sofisticadas, sustentadas en técnicas de IA y aprendizaje automático, ha representado un hito significativo en la lucha contra los ataques cibernéticos.

Esta evolución ha potenciado considerablemente la capacidad de anticipar y prevenir intrusiones, permitiendo una respuesta más ágil y eficaz ante las amenazas digitales que evolucionan constantemente. La integración de algoritmos inteligentes y capacidades predictivas en los sistemas de detección de intrusiones ha marcado un cambio paradigmático en la forma en que se aborda la seguridad informática. La aplicación de modelos basados en IA no solo ha mejorado la eficiencia en la identificación de patrones anómalos y comportamientos maliciosos, sino que también ha permitido una adaptación dinámica a las nuevas formas de ataques cibernéticos, fortaleciendo así las defensas digitales y reduciendo el tiempo de respuesta ante incidentes de seguridad [4].

A pesar de los avances en la sofisticación de los sistemas de detección de intrusiones, surge un desafío significativo derivado de la opacidad y

complejidad inherentes a estas soluciones. Esta falta de transparencia puede obstaculizar la comprensión de su funcionamiento y generar desconfianza en su eficacia, planteando interrogantes sobre la fiabilidad y la capacidad real de proteger contra las amenazas cibernéticas. Por consiguiente, resulta crucial abordar esta problemática mediante el desarrollo de soluciones innovadoras que no solo destaquen por su efectividad en la detección de intrusiones, sino que también se caractericen por su transparencia y comprensibilidad para los usuarios.

La creación de sistemas que sean accesibles y claros en su operatividad no solo fortalecerá la confianza en la ciberseguridad, sino que también empoderará a los usuarios al permitirles comprender y participar activamente en la protección de sus sistemas y datos. La transparencia en el diseño y funcionamiento de los sistemas de detección de intrusiones no solo contribuirá a disipar dudas y temores sobre su eficacia, sino que también facilitará la colaboración entre los responsables de seguridad y los usuarios finales. Al fomentar una mayor comprensión y confianza en las herramientas de ciberseguridad, se promueve una cultura proactiva de protección digital que resulta fundamental en un entorno donde las amenazas cibernéticas evolucionan constantemente. En resumen, la búsqueda de soluciones innovadoras y transparentes en el ámbito de la detección de intrusiones no solo representa un paso crucial para mejorar la seguridad informática, sino que también constituye un pilar fundamental para fortalecer la resiliencia digital y promover una mayor conciencia sobre la importancia de la ciberseguridad en el mundo actual.

## **1.1 Motivación**

Con el aumento del uso de tecnologías de la información y comunicación, se ha transformado la manera en la que interactúan las organizaciones, a pesar de los beneficios, este progreso ha provocado un mayor riesgo de sufrir intrusiones maliciosas que pueden comprometer la integridad, confidencialidad y disponibilidad de sus sistemas y datos.

En este escenario, la ciberseguridad se ha vuelto una preocupación esencial para empresas, usuarios e instituciones. La detección de intrusiones desempeña un papel crucial en la defensa contra estas amenazas al permitir identificar y responder proactivamente a actividades sospechosas en sistemas informáticos. No obstante, el reto radica en que los sistemas de detección de intrusiones, cada vez más avanzados y basados en IA y aprendizaje automático [5,6], pueden volverse complejos y difíciles de comprender. La falta de transparencia en su funcionamiento puede generar desconfianza y obstaculizar la toma de decisiones informadas por parte de los responsables de seguridad.

Por ello, este proyecto busca desarrollar soluciones innovadoras que no solo sean efectivas en la detección de intrusiones, sino también comprensibles y transparentes, la capacidad de interpretar y explicar el funcionamiento de estos sistemas es esencial por varias razones, para garantizar un mínimo de confianza, auditoría, rendimiento y facilitar la interacción humana.

## **1.2 Objetivos**

El presente Trabajo de Fin de Grado (TFG) se enmarca en la búsqueda de soluciones en el campo de ciberseguridad, con un enfoque específico en la detección de intrusos, para ello se inspira en trabajos previos, como el framework de XD-IoT propuesto en [1]. El propósito inicial del XD-IoT es abordar problemas en el ámbito del Internet de las cosas (IoT) mediante tecnologías de aprendizaje profundo, para luego analizar detalladamente los diversos componentes que influyen en las salidas del IoT. Este enfoque será adaptado y aplicado específicamente al campo de la detección de intrusos.

Se explorarán y evaluarán diversas técnicas de IA y aprendizaje automático para determinar su fiabilidad en la detección de amenazas cibernéticas, enfocándose también en la parte explicativa de los modelos, buscando comprender y transparentar los procesos de detección de intrusos para mejorar la confianza y comprensión de los resultados obtenidos.

La IA se refiere a la simulación de procesos de inteligencia humana por parte de sistemas computacionales, permitiendo a las máquinas realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento y la toma de decisiones.

El objetivo principal es **desarrollar un sistema innovador de detección de intrusiones que incorpore técnicas avanzadas de IA y aprendizaje automático**, específicamente dentro del marco de la inteligencia artificial explicable (XAI, por sus siglas en inglés). Este sistema estará especialmente **diseñado para mejorar la comprensión y explicación de sus decisiones, facilitando a los usuarios una mejor comprensión de las amenazas detectadas y la adopción de medidas más efectivas para mitigarlas.**

El enfoque principal del proyecto será el diseño, implementación y evaluación de un sistema efectivo y escalable que identifique actividades maliciosas o anómalas en redes informáticas. Esto implica investigar las últimas tendencias en detección de intrusiones y técnicas de IA, desarrollar un sistema modular y adaptable, crear un software robusto y eficiente y evaluar el rendimiento del sistema en pruebas experimentales.

### **1.3 Metodología de trabajo**

En la realización de este Trabajo de Fin de Grado (TFG), se ha adoptado una metodología que integra diversos enfoques consolidados en los ámbitos de la inteligencia artificial explicable y la detección de intrusiones o amenazas cibernéticas. Este enfoque sistemático asegura una exploración integral y una implementación meticulosa de los sistemas de detección de intrusos, así como una evaluación exhaustiva de su capacidad para explicar y comprender sus decisiones.

La metodología empleada proporciona flexibilidad y adaptabilidad, permitiendo abordar la complejidad y diversidad de los retos asociados al análisis de la explicabilidad e interpretabilidad en la detección de intrusos mediante técnicas de aprendizaje automático.

La estructura de esta metodología se organiza en varias fases clave:

**1. Revisión de la literatura:**

- Exploración de estudios previos y enfoques actuales en inteligencia artificial explicable y detección de intrusiones.
- Identificación de las mejores prácticas y las técnicas más efectivas utilizadas en el campo.

**2. Recopilación y evaluación de conjuntos de datos:**

- Se realizó una búsqueda exhaustiva y una evaluación crítica de conjuntos de datos disponibles, seleccionando aquellos que proporcionen resultados confiables y que se asemejen a un entorno real de comunicaciones.
- La selección de los conjuntos de datos se basó en la opinión de expertos en ciberseguridad, garantizando su adecuación para los objetivos del proyecto.

**3. Desarrollo de sistemas de detección de intrusos:**

- Se desarrollaron sistemas de detección de intrusos utilizando diversos algoritmos de aprendizaje automático.
- Los conjuntos de datos seleccionados en la fase anterior se aplicaron para entrenar y evaluar estos sistemas, asegurando una base sólida y relevante para el análisis.

**4. Implementación de técnicas de evaluación y explicabilidad:**

- Se implementaron técnicas avanzadas tanto para la evaluación del rendimiento como para la explicabilidad e interpretabilidad de los sistemas de detección de intrusos desarrollados.
- Este enfoque permitió no solo medir la efectividad de los modelos, sino también comprender las decisiones tomadas por los mismos.



## 5. Replicación para otros tipos de ataques:

- Se replicó el proceso de entrenamiento y evaluación para otros tipos de ataques utilizando el mismo conjunto de datos.
- Los modelos se volvieron a entrenar y ajustar específicamente para estos nuevos ataques, asegurando que cada modelo estuviera optimizado para detectar las diferentes intrusiones de manera precisa.
- Esta fase permitió realizar más comparaciones de las explicaciones de los modelos y generalizar algunas conclusiones, proporcionando una comprensión más amplia y detallada de su capacidad para detectar distintos tipos de intrusiones.

## 6. Análisis de resultados y formulación de conclusiones:

- Se realizó un análisis exhaustivo de los resultados obtenidos, contrastando los distintos modelos, características y tipos de ataques.
- Las conclusiones se formularon basándose en el desempeño comparativo de los sistemas, proporcionando una visión integral de su efectividad y utilidad en contextos reales de ciberseguridad.

Se ha sostenido una comunicación constante con los tutores mediante el uso de mensajería. Informándose sobre los avances realizados y abordándose las inquietudes y preguntas oportunas.

## 1.4 Organización de la memoria

La memoria se organiza en las siguientes siete secciones principales:

- 1) Introducción:** se presenta una introducción del proyecto con sus motivaciones, objetivos, metodología y la propia organización de este.
- 2) Conceptos básicos:** se explican conceptos básicos para poder entender el resto del proyecto, desde las definiciones de inteligencia artificial o detección de intrusiones hasta los algoritmos de aprendizaje automático o técnicas de explicabilidad utilizadas.

- 3) Análisis preliminares:** se explica la revisión del estado del arte, que abarca los estudios y metodologías actuales en inteligencia artificial explicable y detección de intrusiones. Se describe el conjunto de datos utilizado, seleccionado por su relevancia y adecuación para el proyecto, y se detalla el preprocesamiento de datos realizado para asegurar su calidad. Además, se enumeran los algoritmos de aprendizaje automático implementados para la detección de intrusos y se presentan las técnicas de interpretabilidad aplicadas para explicar y comprender los resultados obtenidos.
- 4) Tecnologías utilizadas:** se describen las diversas tecnologías utilizadas en el proyecto.
- 5) Implementación:** se detalla la implementación en código del proyecto.
- 6) Análisis de resultados:** se analizan los resultados obtenidos para los ataques evaluados: DDoS y PortScan.
- 7) Conclusiones y líneas futuras:** se recopilan las conclusiones derivadas del trabajo realizado, junto con posibles direcciones para investigaciones futuras.

Además de incluir las referencias correspondientes, se proporciona un apéndice que servirá como manual de instalación para facilitar la ejecución del código fuente.

# Conceptos básicos

## 2.1 Detección de intrusiones e inteligencia artificial

Los Sistemas de Detección de Intrusos (IDS, siglas en inglés de Intrusion Detection System) son herramientas esenciales en la seguridad de redes, diseñadas para identificar actividades sospechosas o malintencionadas y violaciones de políticas a través del monitoreo del tráfico de red. Debido al aumento de datos sensibles tanto personales como empresariales que transitan por las redes, la necesidad de medidas de seguridad sólidas ha crecido, haciendo de los IDS un componente crucial para prevenir ciberataques y minimizar los daños y costos asociados [7].

Los IDS se utilizan para monitorear y analizar el tráfico de red, detectando intrusiones que podrían comprometer la seguridad y confidencialidad de los datos. Estos sistemas permiten a los administradores de red mantener una vigilancia constante sobre las amenazas actuales, actuando como una primera línea de defensa contra intentos de acceso no autorizado. Al identificar actividades anómalas o firmas de ataques conocidos, los IDS pueden alertar a los administradores sobre posibles violaciones de seguridad antes de que ocurran daños significativos.

Los IDS se pueden clasificar en dos tipos principales según dónde se implementen:

- **IDS basados en Host (HIDS, siglas en inglés de Host-based Intrusion Detection System):** se instalan en un dispositivo específico, monitoreando los datos que llegan a ese dispositivo. La desventaja principal es que se necesita un IDS en cada dispositivo que se desea proteger, lo que puede aumentar el tiempo de procesamiento y afectar el rendimiento del sistema.
- **IDS basados en red (NIDS, del inglés Network-based Intrusion Detection System):** se despliegan en toda la red para monitorear, capturar y analizar el tráfico en busca de actividades maliciosas. Este enfoque permite una supervisión más amplia de la red, aunque puede ser más complejo de gestionar.

Los IDS utilizan dos enfoques principales para detectar ataques [8]:

- **Detección basada en firmas:** también llamada "detección de uso indebido" o "detección basada en conocimiento", este método es reactivo y utiliza una base de datos de firmas y patrones de ataques conocidos. Cuando el IDS encuentra una coincidencia entre los datos recibidos y las firmas almacenadas, marca la actividad como maliciosa. Este método es eficiente para detectar ataques conocidos con pocos falsos positivos, pero no puede identificar ataques nuevos o desconocidos.
- **Detección basada en anomalías:** también conocida como "detección basada en comportamiento", este método crea un modelo del comportamiento normal del sistema y busca actividades que se desvíen de este modelo para identificar ataques. Su principal ventaja es la capacidad de detectar ataques nuevos y desconocidos, aunque generalmente tiene una mayor tasa de falsos positivos.

La IA se está incorporando cada vez más en los IDS para mejorar su capacidad de detección y reducir los falsos positivos. Algoritmos de aprendizaje automático, como las redes neuronales y los métodos de ensamblaje, se

utilizan para analizar grandes volúmenes de datos de red y aprender patrones de comportamiento normal y anómalo.

Esta integración permite a los IDS adaptarse a nuevas amenazas y mejorar continuamente su precisión y eficiencia.

## **2.2 Aprendizaje automático**

El machine learning (ML), o aprendizaje automático, es una subcategoría de la IA centrada en el desarrollo de algoritmos que pueden identificar patrones en diversos tipos de datos. Estos datos pueden ser numéricos, textuales, visuales o estadísticos, y se almacenan digitalmente para ser procesados por los algoritmos. La capacidad de estos algoritmos para descubrir patrones subyacentes en los datos les permite aprender de forma autónoma y mejorar su rendimiento en la ejecución de tareas específicas con el tiempo.

En esencia, el machine learning permite que los sistemas aprendan y se adapten sin intervención humana directa. A medida que se exponen a más datos, los algoritmos refinan sus habilidades para predecir, clasificar y tomar decisiones basadas en la información procesada previamente. Esta capacidad de adaptación y mejora continua ha sido ampliamente estudiada y documentada, destacando su impacto en sectores como la salud, las finanzas, el transporte y la manufactura [9].

El desarrollo de un modelo de machine learning generalmente sigue cuatro etapas principales. Estas etapas son cruciales para asegurar que el modelo sea preciso, eficiente y aplicable a los problemas específicos que se desean resolver.

### **1- Selección y preparación de datos**

La primera etapa implica la selección y preparación de un conjunto de datos de entrenamiento. Estos datos pueden estar etiquetados (aprendizaje supervisado) o sin etiquetar (aprendizaje no supervisado). Los datos etiquetados incluyen información adicional que guía al modelo sobre qué características debe identificar, mientras que los datos no

etiquetados requieren que el modelo detecte patrones de forma autónoma.

La preparación de datos es un proceso crítico que incluye la limpieza, organización y transformación de los datos para asegurar que estén en un formato adecuado para el entrenamiento del modelo. La limpieza de datos implica eliminar valores erróneos, rellenar datos faltantes y normalizar los datos para evitar sesgos. La organización y transformación pueden incluir la codificación de variables categóricas, la escalación de características y la división del conjunto de datos en subconjuntos de entrenamiento y prueba.

## **2- Selección del algoritmo**

El segundo paso es la selección de un algoritmo adecuado para ejecutar sobre el conjunto de datos de entrenamiento. La elección del algoritmo depende del tipo y volumen de datos, así como del problema específico que se intenta resolver. Algunos algoritmos son más adecuados para tareas de clasificación, mientras que otros se especializan en regresión o clustering.

La selección del algoritmo también considera factores como la interpretabilidad, la precisión, el tiempo de entrenamiento y la capacidad para manejar grandes volúmenes de datos.

## **3- Entrenamiento del algoritmo**

Una vez seleccionado el algoritmo, se procede al entrenamiento de este. Este es un proceso iterativo donde el algoritmo ajusta sus parámetros (pesos y sesgos) para minimizar la diferencia entre sus predicciones y los resultados esperados. Durante el entrenamiento, se ejecutan las variables a través del algoritmo repetidamente, ajustando los parámetros basados en el error de predicción.

El objetivo es encontrar el conjunto de parámetros que optimice la precisión del modelo en el conjunto de datos de entrenamiento. Este

proceso puede incluir técnicas de validación cruzada para evaluar el rendimiento del modelo y evitar el sobreajuste. El entrenamiento continúa hasta que el modelo alcance un nivel aceptable de precisión y generalización.

#### **4- Uso y mejora del modelo**

La etapa final es el uso del modelo entrenado para hacer predicciones sobre nuevos datos. Este modelo se aplica a datos no vistos previamente para evaluar su rendimiento y utilidad práctica. La mejora continua del modelo es fundamental, ya que los datos y las condiciones del entorno pueden cambiar con el tiempo.

La mejora del modelo se logra mediante el reentrenamiento con nuevos datos, ajuste de parámetros adicionales y la implementación de nuevas características. Además, se puede realizar un monitoreo constante del rendimiento del modelo para detectar y corregir cualquier desviación en su precisión y efectividad [10].

#### **Algoritmos supervisados vs no supervisados**

En el ámbito del aprendizaje automático, los algoritmos supervisados se destacan por su capacidad para aprender a partir de datos etiquetados, lo que les permite predecir resultados futuros con precisión. Estos algoritmos, son fundamentales en la clasificación y regresión, donde se necesitan respuestas conocidas para entrenar modelos. Por otro lado, los algoritmos no supervisados operan sin etiquetas y se centran en la exploración de la estructura intrínseca de los datos, como en el clustering y la reducción de dimensionalidad, lo que permite descubrir patrones subyacentes y segmentar datos en grupos similares.

La diferencia fundamental entre ambos radica en la disponibilidad de etiquetas en los datos de entrenamiento. Mientras que los algoritmos supervisados requieren datos etiquetados para su aprendizaje, los no supervisados pueden trabajar con datos sin etiquetar, lo que los hace útiles en situaciones donde la información de las etiquetas es escasa o inexistente. A pesar de esta

distinción, ambos tipos de algoritmos desempeñan roles cruciales en el análisis de datos: los supervisados para hacer predicciones precisas en tareas específicas, y los no supervisados para explorar y comprender la estructura inherente de conjuntos de datos complejos. Además, una revisión rápida de los algoritmos de ML indica que estos pueden resolver desafíos de clasificación, regresión y agrupamiento, adaptándose bien a diferentes condiciones y situaciones [11].

Al tener un conjunto de datos etiquetados por lo tanto usaremos algoritmos de supervisados hay varios tipos de algoritmos para hacer esta clasificación:

- **Redes Bayesianas:** estas redes son capaces de interpretar problemas al identificar relaciones estructurales entre predictores. Son eficientes en términos de tiempo computacional y no requieren la configuración de parámetros. Sin embargo, su desempeño se deteriora con el aumento de datos y no son adecuadas para datos de alta dimensión. Se utilizan principalmente en la clasificación de documentos y en sistemas de diagnóstico médico [12].
- **Regresión Logística:** la regresión logística produce salidas en forma de probabilidades, lo cual es útil para una variedad de aplicaciones. Puede manejar no linealidades y efectos de interacción. Necesita grandes muestras de datos para obtener resultados confiables y puede verse afectada por la multicolinealidad. Se usa comúnmente para analizar tipos de accidentes, la severidad de lesiones y la clasificación de votantes [12].
- **Árboles de decisión:** los árboles de decisión son flexibles y pueden manejar interacciones complejas entre características, así como datos faltantes y redundantes. Ofrecen buena capacidad de generalización y son robustos al ruido con un esfuerzo computacional relativamente bajo. Sin embargo, tienen dificultades con datos de alta dimensión, pueden sobreajustarse, y requieren tiempo considerable para construir. También pueden tener problemas de fragmentación de datos y propagación de errores. Son útiles en aplicaciones como dispositivos implantables, control de calidad en soldadura, análisis de drogas y teledetección [12].



- **Bosques aleatorios:** los bosques aleatorios son rápidos, escalables y robustos al ruido. Evitan el sobreajuste y permiten una visualización y explicación claras de los resultados sin necesidad de ajustes de parámetros. Sin embargo, su eficiencia disminuye a medida que se incrementa el número de árboles. Se utilizan para agrupar pacientes, clasificar datos de micromatrices y en la detección de objetos [12].
- **Máquinas de Vectores de Soporte (SVM, del inglés Support Vector Machines):** las SVM son precisas y evitan el sobreajuste. Permiten una selección flexible de núcleos para manejar no linealidades y su rendimiento no depende del número de características, lo que las hace versátiles. No obstante, son complejas, con tiempos de entrenamiento más lentos y su rendimiento depende en gran medida de los parámetros seleccionados. Se aplican principalmente en la clasificación de textos [12].
- **k-Nearest Neighbors (k-NN, en español significa “k vecinos más cercanos”):** este algoritmo es adecuado para clasificar múltiples tipos de datos y no depende de cómo están distribuidos los puntos de muestra. Sin embargo, es menos eficiente y su rendimiento depende de elegir el valor correcto de 'k'. Además, puede verse afectado por el ruido y características innecesarias, y su eficiencia varía según el tamaño del conjunto de datos. Se usa en la estimación de densidad, visión y geometría computacionales.
- **Redes neuronales:** las redes neuronales son buenas para manejar relaciones complejas y no lineales, y no están limitadas por suposiciones estrictas de linealidad o normalidad. Son resistentes a entradas irrelevantes y al ruido. Sin embargo, suelen ser más lentas para entrenar, su rendimiento depende del tamaño de las capas ocultas y los valores de los parámetros elegidos, y son difíciles de interpretar. Se utilizan principalmente en la clasificación de imágenes [12].
  - **Autoencoders:** un tipo específico de red neuronal utilizada comúnmente en técnicas no supervisadas es el autoencoder. Los autoencoders son útiles para aprender representaciones eficientes de los datos, típicamente para la reducción de

dimensionalidad o la extracción de características. Consisten en dos partes: un codificador que reduce las dimensiones de los datos de entrada y un decodificador que intenta reconstruir los datos originales a partir de la representación reducida. Los autoencoders funcionan entrenando la red para minimizar la diferencia entre los datos de entrada y la salida reconstruida. Este proceso permite al autoencoder capturar las características más importantes de los datos.

En el apartado 3.4, se describirán los principales algoritmos supervisados de aprendizaje automático utilizados en este proyecto: Random Forest, Support Vector Machines (SVM), XGBoost, Perceptrón Multicapa (MLP), LightGBM (LGB) y Regresión Logística. Cada uno de estos algoritmos tiene características y ventajas únicas que los hacen adecuados para diferentes tipos de problemas y conjuntos de datos. Se presentará una breve introducción a cada método, seguida de una explicación detallada de sus fundamentos teóricos, ventajas y desventajas, y cómo se implementaron en el contexto de este estudio.

## 2.3 Explicabilidad

Es común escuchar que los modelos de aprendizaje automático son "cajas negras", ya que, aunque son capaces de hacer buenas predicciones, es difícil entender la lógica detrás de estas. Esta afirmación es cierta en gran medida porque muchos científicos de datos aún no dominan la extracción de perspectivas de los modelos.

No obstante, existen técnicas para extraer las siguientes perspectivas de modelos sofisticados de aprendizaje automático [13]:

- **Identificación de características clave:** ¿qué características en los datos considera el modelo como las más importantes?

- **Análisis de predicciones individuales:** para cualquier predicción individual de un modelo, ¿cómo afecta cada característica en los datos esa predicción en particular?
- **Evaluación del impacto general de las características:** ¿cómo afecta cada característica las predicciones del modelo en un sentido más amplio? Es decir, ¿cuál es su efecto típico cuando se considera un gran número de predicciones posibles?

### **Importancia de estas perspectivas**

Estas perspectivas son valiosas por varias razones, incluyendo:

- Depuración de modelos
- Optimización de la ingeniería de características
- Dirección en la recolección de datos futuros
- Apoyo a la toma de decisiones humanas
- Generación de confianza en el modelo

### **Depuración de modelos**

En el ámbito real, los datos suelen ser poco fiables, desorganizados y, en general, sucios. Es común que se introduzcan errores al escribir el código de preprocesamiento, y la fuga de objetivos es una preocupación constante. Por ello, es más habitual encontrar errores en algún punto de un proyecto de ciencia de datos que no hacerlo. Debido a la frecuencia y las posibles consecuencias serias de los errores, la depuración es una de las habilidades más importantes en la ciencia de datos. Identificar los patrones que un modelo detecta te permitirá notar cuándo estos patrones no se alinean con tu conocimiento del mundo real, lo que a menudo es el primer paso para localizar y corregir errores [13].

### **Optimización de la ingeniería de características**

La ingeniería de características es a menudo la forma más efectiva de mejorar la precisión de un modelo. Este proceso generalmente implica crear nuevas características mediante transformaciones de los datos brutos o de características ya existentes.

Aunque en algunos casos puedes confiar en la intuición sobre el tema subyacente, necesitarás más dirección cuando trabajes con cientos de características brutas o cuando carezcas de conocimiento previo sobre el tema. Como comenta Dansbecker en [13] un ejemplo extremo de esto se observó en una competencia de Kaggle para predecir incumplimientos de préstamos. Esta competencia incluía cientos de características brutas, nombradas de manera críptica como  $f_1$ ,  $f_2$ ,  $f_3$ , etc., para simular un escenario donde la intuición sobre los datos era mínima. Un competidor descubrió que la diferencia entre dos características ( $f_{527}$  y  $f_{528}$ ) formaba una nueva característica muy poderosa. Los modelos que incluían esta diferencia superaban a aquellos que no la incluían. Las técnicas adecuadas harían evidente que  $f_{527}$  y  $f_{528}$  son características importantes y que su interacción es crucial, dirigiéndote así a explorar transformaciones de estas variables.

Con el aumento de conjuntos de datos que comienzan con cientos o miles de características, este enfoque es cada vez más importante.

### **Dirección en la recolección de datos futuros**

Aunque no puedes influir en los conjuntos de datos que descargas en línea, muchas empresas y organizaciones tienen la posibilidad de diversificar los tipos de datos que recolectan. Debido a que la recopilación de nuevos datos puede ser costosa o inconveniente, es fundamental saber si realmente vale la pena. Los análisis basados en modelos te brindan una comprensión precisa del valor de las características actuales, lo que te ayudará a identificar qué nuevos datos podrían ser más útiles y valiosos [13].

### **Apoyo a la toma de decisiones humanas**

Algunas decisiones son tomadas automáticamente por modelos, como en el caso de las recomendaciones de productos en Amazon. Sin embargo, muchas decisiones importantes siguen siendo tomadas por humanos. En estos casos, las perspectivas derivadas de los modelos pueden ser más valiosas que las propias predicciones, proporcionando información crítica para la toma de decisiones [3].

## Generación de confianza en el modelo

Para que las personas confíen en un modelo en decisiones críticas, es fundamental verificar ciertos aspectos básicos. Esto es especialmente crucial debido a la frecuencia con la que ocurren errores en los datos. Presentar perspectivas que se alineen con la comprensión general del problema ayudará a generar confianza en el modelo, incluso entre aquellos que no tienen un conocimiento profundo en ciencia de datos.

Las técnicas para extraer perspectivas valiosas de los modelos de aprendizaje automático mejoran considerablemente las habilidades en depuración, ingeniería de características, recolección de datos, toma de decisiones y generación de confianza en los modelos. Estas competencias son esenciales en un entorno donde los conjuntos de datos son cada vez más complejos y extensos [13].

Para obtener estas perspectivas, se han implementado diversas técnicas en los modelos de aprendizaje automático descritos en el apartado 3.5, logrando varios éxitos y enfrentando algunos fracasos. Las técnicas utilizadas incluyen:

- **Análisis de residuos:** evalúa las diferencias entre las predicciones del modelo y los valores reales para identificar patrones de error [14].
- **Análisis de influencia:** determina el impacto de eliminar o modificar observaciones específicas en el conjunto de datos sobre las predicciones del modelo [15].
- **Análisis de sensibilidad:** mide cómo cambian las predicciones del modelo cuando se alteran las características de entrada [16].
- **Permutación de características:** evalúa la importancia de una característica intercambiando aleatoriamente sus valores y observando la reducción en la precisión del modelo [17].

Además, se han utilizado métodos avanzados de explicabilidad como SHAP y LIME:

- **SHAP (SHapley Additive exPlanations):** proporciona explicaciones tanto globales como locales de las predicciones del modelo, asignando valores SHAP a cada característica para entender su impacto [18].

- **LIME (Local Interpretable Model-agnostic Explanations):** ofrece explicaciones locales para predicciones individuales, creando modelos simples y localizados alrededor de cada predicción para interpretar cómo cada característica contribuye al resultado [19].

Con la implementación de análisis de dependencia de características, residuos, influencia, sensibilidad y permutación, junto con métodos avanzados como SHAP y LIME, se ha logrado una comprensión más profunda y detallada del comportamiento y la lógica detrás de las predicciones de los modelos.

### 2.3.1 SHAP

El marco de SHAP se ha consolidado como un estándar de oro para explicaciones locales gracias a su sólido fundamento teórico y aplicabilidad general. SHAP, basado en los valores de Shapley de la teoría de juegos, ofrece una medida unificada de la importancia de las características que es única en su capacidad para cumplir con ciertos criterios deseables para explicaciones de modelos [18].

Es un enfoque de interpretabilidad local unificado desarrollado por Lundberg y Lee en 2017. Este marco se basa en los valores de Shapley, una solución de teoría de juegos que asigna ganancias a los jugadores en un juego cooperativo. En el contexto de machine learning, los "jugadores" son las características del modelo, y los valores de Shapley miden la contribución de cada característica al resultado del modelo.

#### 2.3.1.1 Fundamentos Teóricos

Los valores de Shapley se calculan como la contribución promedio de una característica a todas las posibles combinaciones de características en el modelo. Esto asegura que cada característica sea evaluada en todos los contextos posibles, proporcionando una medida robusta de su importancia.

Los valores de Shapley cumplen con cuatro propiedades deseables:

- **Eficiencia:** la suma de las contribuciones de todas las características es igual al resultado total.
- **Simetría:** si dos características contribuyen de manera igual a cualquier subconjunto de características, sus valores de Shapley son iguales.
- **Dummy:** si una característica no cambia el resultado en ningún subconjunto, su valor de Shapley es cero.
- **Aditividad:** los valores de Shapley de dos juegos combinados son la suma de los valores de Shapley de cada juego por separado.

#### 2.3.1.2 Métodos Basados en SHAP

A continuación, se presentan los distintos métodos basados en SHAP [18] que se han utilizado en el código:

KernelSHAP es una adaptación de LIME que aproxima los valores de SHAP. Este método es agnóstico al modelo, lo que significa que puede aplicarse a cualquier tipo de modelo, pero a costa de una eficiencia computacional reducida.

LinearSHAP específico para modelos lineales, utiliza los coeficientes de ponderación del modelo y, opcionalmente, tiene en cuenta las correlaciones entre características.

TreeSHAP es una extensión eficiente de SHAP para árboles de decisión y ensamblajes de árboles. Este método aprovecha la estructura de los árboles de decisión para calcular exactamente los valores de SHAP con una eficiencia mucho mayor comparada con métodos más generales como KernelSHAP.

DeepSHAP combina los valores de SHAP con métodos de retropropagación como DeepLIFT y LRP para proporcionar explicaciones específicas para redes neuronales profundas. Este método permite calcular las contribuciones de las características en modelos complejos con mayor precisión.

### **2.3.1.3 Aplicaciones**

SHAP se utiliza ampliamente para interpretar modelos de machine learning [20], ofreciendo una visión detallada de cómo cada característica afecta las predicciones del modelo. Esto es especialmente útil en aplicaciones donde la transparencia y la explicabilidad son cruciales, como en la medicina y las finanzas.

Los valores de SHAP permiten también a los usuarios diagnosticar problemas en los modelos de machine learning, como la detección de características que pueden estar introduciendo sesgos o que no están aportando información relevante al modelo.

También es una herramienta poderosa para la visualización de datos. Permite crear gráficos que muestran la importancia de las características y cómo interactúan entre sí, facilitando una mejor comprensión de los datos y del modelo.

SHAP proporciona un marco teórico sólido y herramientas prácticas para la interpretabilidad de modelos de machine learning. Su capacidad para ofrecer explicaciones locales precisas y su aplicabilidad a una amplia gama de modelos lo convierten en una herramienta invaluable para investigadores y profesionales que buscan desarrollar modelos transparentes y confiables.

### **2.3.2 LIME**

LIME, es una técnica utilizada para interpretar modelos de aprendizaje automático complejos. Su objetivo es aproximar el comportamiento de un modelo de caja negra (es decir, un modelo cuyo funcionamiento interno no es fácilmente comprensible) utilizando un modelo más simple y explicable alrededor de un punto de interés específico [19].

#### **2.3.2.1 Fundamentos Teóricos**

La idea principal de LIME es explicar la predicción de un modelo complejo para una instancia específica al aproximar el modelo complejo con un modelo



simple, pero solo en la vecindad de esa instancia. En otras palabras, LIME trata de entender cómo una pequeña perturbación en las características de una instancia afecta su predicción. Este modelo simple es generalmente una regresión lineal o un árbol de decisión, que son fáciles de interpretar.

LIME se implementa a través de varios pasos clave [21]:

**1. Generación de puntos nuevos:**

- LIME comienza generando un conjunto de datos sintéticos alrededor de la instancia de interés  $x$ . Estos nuevos puntos se generan perturbando ligeramente las características de  $x$ .
- Por ejemplo, si la instancia de interés tiene características como la edad, el ingreso y el puntaje crediticio, LIME crea nuevas instancias modificando ligeramente estos valores. Esto se hace para explorar cómo las pequeñas variaciones en las características afectan la predicción del modelo.

**2. Ponderación local:**

- Cada punto sintético generado se pondera de acuerdo con su proximidad a la instancia original  $x$ .
- Se utiliza un kernel Gaussiano para asignar pesos, de manera que los puntos más cercanos a  $x$  tengan un peso mayor que los puntos más alejados. Este paso es crucial porque LIME se enfoca en la vecindad inmediata de  $x$  para construir el modelo explicativo.

**3. Predicción con el modelo complejo:**

- LIME aplica el modelo complejo original para obtener las predicciones para cada uno de los puntos sintéticos generados.
- Esto proporciona un conjunto de datos donde las entradas son los puntos perturbados y las salidas son las predicciones del modelo complejo.

#### **4. Estandarización:**

- Antes de ajustar el modelo explicativo, las características de los puntos sintéticos se estandarizan.
- La estandarización asegura que todas las características se midan en la misma escala, lo cual es importante para evitar que características con escalas mayores dominen la explicación.

#### **5. Selección de características:**

- LIME selecciona un subconjunto de características para incluir en el modelo explicativo para mantener la interpretabilidad.
- Generalmente, se utiliza Lasso (una técnica de regresión que incluye una penalización para reducir el número de características) para seleccionar las características más importantes.

#### **6. Ajuste del modelo explicativo:**

- Con el conjunto de datos sintetizado y estandarizado, LIME ajusta un modelo simple (por ejemplo, una regresión lineal ponderada) para predecir las salidas del modelo complejo.
- Este modelo simple se entrena utilizando los pesos calculados en el paso de ponderación local, dando más importancia a los puntos más cercanos a  $x$ .

El resultado de este proceso es un modelo local simple que explica cómo las características individuales afectan la predicción del modelo complejo para la instancia de interés. Los coeficientes del modelo simple indican la dirección (positiva o negativa) y la magnitud del impacto de cada característica.

#### **2.3.2.2 Aplicaciones**

LIME es utilizado en diversas áreas para proporcionar interpretaciones locales de modelos complejos [22]. Un ejemplo concreto es su aplicación en el campo del riesgo crediticio, donde se usa para explicar las predicciones de modelos como el Gradient Boosting Model (GBM) en términos de características financieras clave, como el puntaje de crédito y el historial de pagos.

Un problema significativo con LIME es su inestabilidad. Al aplicar LIME varias veces bajo las mismas condiciones, se pueden obtener diferentes explicaciones, lo que dificulta confiar en sus resultados. Esta inestabilidad se debe principalmente a la aleatoriedad en la generación de puntos nuevos y en la selección de características.

Para abordar la inestabilidad de LIME, se han propuesto índices de estabilidad, como el Índice de Estabilidad de Variables (VSI) y el Índice de Estabilidad de Coeficientes (CSI). Estos índices miden la consistencia de las características y los coeficientes de los modelos explicativos de LIME en múltiples ejecuciones, proporcionando una evaluación cuantitativa de la estabilidad de las explicaciones [19].

LIME es una herramienta poderosa para interpretar modelos de aprendizaje automático complejos al proporcionar explicaciones locales interpretables. Sin embargo, su inestabilidad puede ser un obstáculo significativo para su adopción. Las propuestas recientes para medir y mejorar la estabilidad de LIME son pasos importantes hacia la creación de explicaciones más confiables y consistentes.

# Análisis preliminares

## 3.1 Revisión del estado del arte

A continuación, se presenta un análisis del estado actual de la técnica en relación con el problema planteado en este proyecto, centrándonos en investigaciones relevantes en el ámbito de la detección de intrusos con capacidades XAI. A pesar de que se han abordado sistemas de detección de intrusos, se ha observado que muchos de los estudios disponibles presentan limitaciones significativas en términos de contenido y metodología. Si bien existen varios trabajos que han implementado sistemas de detección de intrusos con capacidades XAI, se ha notado una falta de profundidad en el análisis y la evaluación de estos sistemas.

Entre los estudios previos que podrían mencionarse se encuentran los ya comentados en la introducción [5] y [6], así como el comentado en los objetivos [1], en el que se inspira este trabajo. Este último se basa en el propósito inicial del XD-IoT, que busca abordar problemas en el ámbito del Internet de las cosas (IoT) mediante tecnologías de aprendizaje profundo, para luego analizar detalladamente los diversos componentes que influyen en las salidas del IoT, y será adaptado a la detección de intrusos.

Además de los mencionados, existen otros artículos relevantes que han abordado la detección de intrusiones utilizando enfoques similares basados en IA y aprendizaje automático. Estos artículos se han obtenido mediante una

revisión exhaustiva de documentos científicos publicados en revistas especializadas en ciberseguridad. La búsqueda se ha realizado desde el año 2020 hasta el presente, utilizando términos de búsqueda como "detección de intrusiones", "inteligencia artificial", "aprendizaje automático", "explicabilidad" entre otros relacionados. A continuación, se presentan estos artículos por orden de publicación.

Wang et al. [23] presentaron un marco propuesto que se compone de dos partes: la estructura tradicional del IDS y una sección para mejorar la interpretabilidad. Para el IDS tradicional, se incluyen conjunto de datos, modelos entrenados y predicciones, con clasificadores uno contra todos y multiclase. El enfoque del marco se centra en la interpretabilidad, generando explicaciones locales y globales junto con el uso de SHAP. Dos métodos de explicación global analizan características importantes y relaciones, mientras que la explicación local detalla la salida del IDS y la relevancia de las características.

Aunque tiene interpretabilidad local y global, tiene bastante margen de mejoras, conjuntos de datos algo obsoleto.

T. Dias et al. [24] proporcionaron un sistema de detección de anomalías mediante Reglas y Aprendizaje Automático, para el tráfico malicioso de DNS sobre el protocolo HTTPS, los datos se procesan para calcular evidencias dinámicas que representan la información de la secuencia de paquetes. Estas evidencias se envían al motor de reglas Drools, donde se generan hipótesis y conclusiones mediante inferencia. La verificación del motor concluye cuando se alcanza una conclusión o se prueban todas las reglas. Posteriormente, se obtiene un árbol de reglas coincidentes, lo que permite generar una justificación y modificar el estado de los activos peligro si hay conclusión, advertencia si se formula al menos una hipótesis, y normal si no se alcanza ninguna predicción. En el pipeline de generación de reglas, las evidencias dinámicas se procesan en un contenedor activando la generación de reglas. Estas evidencias se envían a un modelo de árbol de decisión para predecir posibles anomalías. La implementación actual genera reglas basadas en el

número de nodos hoja de clase de ataque, cada una con las condiciones de la rama de ataque.

Tiene una base de conocimientos escasa, no experimentan con otros modelos de aprendizaje automático más prometedores para hacerlo más robusto como sistema de detección de intrusiones (IDS), permitiendo así detectar una variedad más amplia de ataques.

Jian et al. [25] propusieron un autoencoder para la detección de anomalías industriales en imágenes, mejorando su capacidad mediante una pérdida de similitud estructural (SSIM) que evalúa la similitud local de intensidades de píxeles, a través de un algoritmo que detalla la generación del "Mapa de Atención Consciente de la Interpretabilidad" (IAAM) y la detección de anomalías, brindando una metodología integral para identificar defectos en aplicaciones industriales mediante el análisis de imágenes.

Pueden requerir el ajuste de hiperparámetros, lo cual debe considerarse junto con la selección de un umbral apropiado y complejidad computacional adicional.

Maricar et al. [26] desarrollaron un modelo utilizando tres clasificadores diferentes: Regresión Logística, el clasificador XGB de machine learning y el clasificador LGBM, de los cuales el XGB resultó ser el más efectivo. Integraron el modelo LIME, que convierte cada valor de característica en la contribución del predictor, permitiendo que, para cada muestra de datos, un intérprete obtenga una perspectiva local y aclare problemas del clasificador. Este método proporciona una visión del funcionamiento del modelo de caja negra utilizado en machine learning; según su algoritmo, se requieren múltiples instancias de interrupción con cambios mínimos en los valores para justificar una explicación. El método utiliza estos datos manipulados para desarrollar un modelo lineal local alrededor de la observación alterada, predice los resultados de los datos alterados evaluando la distancia entre cada observación afectada y la original para calcular un puntaje de similitud, y determina cómo mejorar la representación de las predicciones utilizando técnicas de preprocesamiento de datos. Estos datos preprocesados se emplean para ajustar un modelo a los

datos alterados, y los resultados se definen en función de los coeficientes del modelo, también conocidos como pesos.

Considerando el resultado del estado actual del arte, parece apropiado proponer una alternativa en cuanto a la metodología de trabajo, el enfoque de entrenamiento y el tipo de modelo utilizado.

### **3.2 Conjuntos de datos**

Tras una minuciosa evaluación de distintos conjuntos de datos buscando uno que facilite la obtención de resultados confiables que se asemejen a un entorno real de comunicaciones, se ha optado por el conjunto de datos CICIDS-2017 [27], tomando en cuenta varios factores cruciales. Uno de los aspectos determinantes es su actualidad, siendo del año 2017, lo que lo diferencia de otras opciones que datan de años anteriores. Esta actualidad es esencial para asegurar la relevancia y vigencia de los datos en un contexto contemporáneo de ciberseguridad. Además, el conjunto de datos CICIDS-2017 está disponible públicamente a través de la plataforma Kaggle [27], lo que facilita su acceso y verificación por parte de otros investigadores y profesionales del campo.

El conjunto de datos CICIDS 2017 destaca por su riqueza en cuanto a la variedad y representatividad de los ataques que contiene. Desde ataques de denegación de servicio (DoS) hasta ataques de fuerza bruta e infiltración, abarca una amplia gama de amenazas modernas y relevantes en el ámbito de la ciberseguridad. Esta diversidad lo posiciona como una opción más adecuada en comparación con conjuntos de datos más antiguos como KDD Cup 99 [28], que pueden carecer de la complejidad y sofisticación de los ataques contemporáneos. Al tener una gran cantidad de datos, proporciona una base sólida y diversa para el entrenamiento de modelos de Deep Learning. El Deep Learning requiere conjuntos de datos extensos para aprender patrones complejos y sutiles en los datos. Esta característica facilita la creación de modelos de detección de intrusiones más robustos y precisos, capaces de adaptarse a las complejidades del entorno de seguridad informática actual.

Se evaluaron varios conjuntos de datos en términos de novedad, representatividad, volumen de datos y su aplicabilidad para estudios relacionados con la detección de intrusiones y la inteligencia artificial explicable:

1. **N-BaloT Dataset** [29]

Este dataset está orientado a detectar ataques de botnet en dispositivos IoT. Aunque es útil para el análisis de seguridad en el contexto de IoT, su enfoque específico y la menor presencia en la literatura lo hacen menos adecuado para nuestro objetivo general de análisis de ataques de red.

2. **CSE-CIC-IDS2018 Dataset** [30], [31]

Este dataset es muy similar al CICIDS 2017 y contiene un volumen de datos significativo. Sería otra muy buena elección para estudiar, pero aun así contiene algo menos de variedad de ataques que CICIDS 2017.

3. **KDD Cup 1999 Data** [28]

Este es uno de los datasets más antiguos y ampliamente utilizados en la detección de intrusiones. Sin embargo, la variedad y complejidad de los ataques representados no es tan rica como en datasets más recientes, lo que limita su aplicabilidad a los desafíos actuales de ciberseguridad.

4. **CIRA-CIC-DoHBrw-2020 Dataset** [32]

Este conjunto de datos se centra en ataques relacionados con el tráfico DNS. Aunque es valioso para estudios específicos de tráfico DNS, su enfoque limitado lo hace menos adecuado para un análisis más amplio de seguridad de red.

5. **NSL-KDD Dataset** [33]

NSL-KDD es una versión mejorada del KDD Cup 1999 con varias mejoras en la calidad de los datos. Sin embargo, sigue teniendo limitaciones similares en términos de variedad y representatividad de ataques modernos.



Es por esto por lo que el dataset elegido es el CICIDS 2017 porque como ya se ha comentado es el que mejor engloba los aspectos de novedad y representatividad de ataques, calidad y volumen de datos y aplicabilidad a la IA explicable.

En cuanto a los estudios relacionados con la inteligencia artificial explicable (XAI, del inglés eXplainable Artificial Intelligence), si bien se han realizado análisis del conjunto de datos CICIDS 2017 desde esta perspectiva, es importante destacar que estos estudios no han profundizado lo suficiente en el análisis y la evaluación de modelos de detección de intrusiones interpretables. Aunque existen investigaciones que exploran la aplicabilidad de la XAI en este contexto, aún queda un amplio espacio para investigaciones más exhaustivas que analicen en profundidad la interpretabilidad de los modelos y su impacto en la detección de amenazas cibernéticas.

En términos de calidad y representatividad del conjunto de datos, el CICIDS 2017 se destaca por contener datos de tráfico de red real y un mapeo completo de la infraestructura. A diferencia de otros conjuntos de datos más sintéticos, esta autenticidad y fidelidad a escenarios reales proporciona una base sólida para la evaluación de modelos de detección de intrusiones. Además, la cantidad suficiente de instancias y atributos facilita una evaluación efectiva de estos modelos, lo que contribuye a su utilidad y aplicabilidad en entornos de ciberseguridad prácticos.

Además de sus cualidades previamente mencionadas, el conjunto de datos CICIDS 2017 también se destaca por su relativa actualidad. Esta actualidad asegura que los resultados y las conclusiones derivadas del análisis que sean más pertinentes y aplicables a los desafíos actuales que enfrentan los sistemas de seguridad informática.

### **3.2.1 Tipos de ataques**

Este conjunto de datos abarca una variedad de tipos de ataques, entre los cuáles se incluyen los siguientes.

#### **3.2.1.1 DDoS y DoS**

La Denegación de Servicio Distribuido (DDoS) es un método de ataque que busca inundar un sistema, servicio o red con una avalancha de solicitudes, con el fin de bloquear el acceso para los usuarios autorizados. Por ejemplo, un grupo de atacantes podría emplear una red de bots para enviar una oleada de solicitudes a un servidor web, sobrecargándolo y dejándolo incapaz de atender las peticiones legítimas de los usuarios.

La Denegación de Servicio (DoS) es similar al ataque DDoS, pero realizado por un solo dispositivo o sistema.

#### **3.2.1.2 PortScan**

El escaneo de puertos es una táctica de ataque, en la que el atacante examina los puertos de un sistema en busca de aquellos que estén abiertos y puedan representar vulnerabilidades susceptibles a ataques. Por ejemplo, un atacante podría emplear herramientas automatizadas para escanear los puertos de un servidor, con el propósito de identificar aquellos que estén abiertos y así poder acceder al sistema de forma no autorizada.

#### **3.2.1.3 BotNet**

Un bot es un software que ejecuta automáticamente tareas repetitivas en línea. En un ataque de botnet, un agresor aprovecha una red de bots controlados de manera remota para llevar a cabo acciones maliciosas, como enviar correos no deseados, perpetrar ataques de denegación de servicio o sustraer datos. Por ejemplo, un ciberdelincuente podría utilizar una botnet para desatar un ataque de denegación de servicio distribuido (DDoS) contra un sitio web.

#### **3.2.1.4 Infiltration**

El ataque por infiltración implica el intento de un atacante de acceder a un sistema o red de manera no autorizada. Por ejemplo, un atacante podría intentar acceder a una red corporativa utilizando credenciales robadas o explotando una vulnerabilidad en el sistema de autenticación.

#### **3.2.1.5 Web Attack**

Los ataques web son aquellos dirigidos a aplicaciones web y servicios en línea abarcan una diversidad de acciones maliciosas. Estas pueden implicar ataques de inyección SQL, ataques de cross-site scripting (XSS), manipulación de cookies, entre otros. Por ejemplo, mediante un ataque de inyección SQL, un atacante podría manipular una base de datos web para acceder de manera no autorizada a información confidencial.

#### **3.2.1.6 Brute Force**

En una táctica de fuerza bruta, un atacante busca descifrar una contraseña probando todas las combinaciones posibles hasta dar con la correcta. Por ejemplo, un atacante podría intentar acceder a una cuenta de usuario probando diferentes combinaciones de contraseñas hasta encontrar la correcta.

#### **3.2.1.7 Heartbleed**

Heartbleed representó una falla crítica de seguridad en la biblioteca de cifrado OpenSSL, permitiendo a un intruso acceder a la memoria de un servidor web y extraer datos sensibles, como claves privadas, tokens de sesión y datos de usuario. Este ataque se basaba en una deficiencia en la implementación de la extensión de latido de TLS/SSL. Por ejemplo, un atacante podría aprovechar la vulnerabilidad de Heartbleed para sustraer información confidencial de un servidor web.

### 3.3 Preprocesamiento de datos

Debido a la gran cantidad de datos contenidos en cada archivo CSV correspondiente a diversos tipos de ataques y considerando los recursos actuales disponibles, se ha tomado la decisión de no analizar exhaustivamente todos los tipos de ataques presentes en el conjunto de datos. En su lugar, se ha optado por focalizar el análisis en dos tipos específicos de ataques:

#### **PortScan y DDoS.**

Antes de proceder al análisis detallado de los ataques seleccionados, se han realizado una serie de modificaciones y preprocesamientos a los archivos CSV para asegurar la calidad y la manejabilidad de los datos.

#### 3.3.1 Limpieza de datos

La limpieza de datos es una fase crucial en el preprocesamiento, ya que garantiza la uniformidad, integridad y calidad de los datos que se van a analizar. En este apartado se detallan los pasos específicos que se llevaron a cabo para limpiar los datos del conjunto relacionado con los ataques PortScan y DDoS.

En muchos conjuntos de datos, los nombres de las columnas pueden contener espacios en blanco debido a errores durante la recopilación o entrada de datos. Estos espacios pueden causar problemas durante el análisis y la manipulación de datos, ya que muchos algoritmos y funciones de procesamiento no manejan bien los nombres de columnas con espacios. Para solucionar este problema, se eliminaron todos los espacios en blanco de los nombres de las columnas.

Los valores infinitos y nulos en los datos pueden surgir por diversas razones, como errores de medición, entradas incorrectas o valores fuera de los rangos esperados. Estos valores pueden afectar negativamente los análisis estadísticos y los modelos de aprendizaje automático, ya que muchos algoritmos no pueden procesar valores infinitos o nulos.

Para ello se ha realizado:

- Reemplazo de valores infinitos con valores nulos.
- Eliminación de filas con valores nulos.

La limpieza de datos es fundamental para uniformidad y consistencia, integridad y calidad de los datos y preparación para el análisis y modelado, realizado en código en [34].

### **3.3.2 Selección de características**

El objetivo principal es reducir la cantidad de características (o variables) en el conjunto de datos, manteniendo aquellas que son más relevantes para el problema que se está analizando. Esto no solo simplifica el modelo y reduce el tiempo de computación, sino que también puede mejorar el rendimiento del modelo al eliminar el ruido y evitar el sobreajuste.

En los dos ficheros .csv de los que se parte y se van a analizar, tanto el de PortScan y DDoS, cuentan con aproximadamente 250.000 instancias y cada uno con 78 características, lo que hace muy difícil el estudio de todas ellas computacionalmente, por lo que se procede de la siguiente manera.

Se emplea un modelo de Random Forest que es un conjunto de árboles de decisión que funcionan de manera conjunta para mejorar la precisión de la predicción y controlar el problema del sobreajuste. Cada árbol en el bosque se entrena utilizando un subconjunto aleatorio del conjunto de datos original y un subconjunto aleatorio de características. Esta diversidad en los árboles reduce la varianza del modelo y mejora su capacidad de generalización.

Una vez entrenado, el modelo calcula la importancia de cada característica. La importancia de una característica se mide en términos de cómo contribuye a reducir la impureza (como el índice de Gini o la entropía) en los nodos de los árboles de decisión. Las características que logran grandes reducciones en la impureza reciben mayores puntuaciones de importancia.

Se seleccionan las características con las mayores puntuaciones de importancia. En este caso, se seleccionan las **20 características más importantes**. Este conjunto reducido de características es el que se utilizará en los análisis posteriores. Se ha intentado con un mayor número de características, incluso con todas en un principio, pero el impacto en el rendimiento es muy grande y la precisión de los modelos, como se ve en el apartado de análisis de resultados, es una precisión excelente.

Esto da una serie de beneficios como son la reducción de la dimensionalidad del conjunto, mejora del rendimiento al reducirse el ruido de los datos y previene el sobreajuste al tener menos características.

Una vez hecho esto con el conjunto de datos perteneciente al ataque DDoS, se cogerán esas 20 mismas características y serán las utilizadas también en el de PortScan, para poder así comparar mejor los resultados y ver equivalencias y diferencias entre un ataque y otro, esto puede afectar a algunos modelos en sus precisiones, aunque siguen teniendo precisiones muy buenas.

Definición de desviación estándar: una medida de cuánto varían los valores de un conjunto de datos en relación con el promedio. Es una de las medidas de dispersión más utilizadas en estadística.

A continuación, se describe cada una de ellas:

1. **Fwd Packet Length Max**: longitud máxima de los paquetes enviados hacia adelante en una conexión.
2. **Fwd Packet Length Mean**: longitud media de los paquetes enviados hacia adelante.
3. **Avg Fwd Segment Size**: tamaño medio de los segmentos enviados hacia adelante.
4. **Init\_Win\_bytes\_forward**: tamaño inicial de la ventana de bytes en la dirección hacia adelante.
5. **Act\_data\_pkt\_fwd**: número de paquetes de datos activos enviados hacia adelante.
6. **Subflow Fwd Bytes**: total de bytes enviados en subflujos hacia adelante.

7. **Fwd IAT Total**: tiempo total entre llegadas de paquetes enviados hacia adelante.
8. **Fwd IAT Max**: máximo tiempo entre llegadas de paquetes enviados hacia adelante.
9. **Destination Port**: puerto de destino de la conexión.
10. **Total Length of Fwd Packets**: longitud total de todos los paquetes enviados hacia adelante.
11. **Fwd IAT Std**: desviación estándar del tiempo entre llegadas de paquetes enviados hacia adelante.
12. **Fwd Header Length**: longitud total de las cabeceras de los paquetes enviados hacia adelante.
13. **Total Fwd Packets**: número total de paquetes enviados hacia adelante.
14. **Subflow Fwd Packets**: número de paquetes enviados en subflujos hacia adelante.
15. **Fwd Packet Length Std**: desviación estándar de la longitud de los paquetes enviados hacia adelante.
16. **Init\_Win\_bytes\_backward**: tamaño inicial de la ventana de bytes en la dirección inversa.
17. **Bwd Packet Length Min**: longitud mínima de los paquetes recibidos desde atrás.
18. **Fwd Header Length.1**: longitud total de las cabeceras de los paquetes enviados hacia adelante (parece ser una repetición del 12).
19. **Bwd Header Length**: longitud total de las cabeceras de los paquetes recibidos desde atrás.
20. **Fwd IAT Mean**: tiempo medio entre llegadas de paquetes enviados hacia adelante.

### 3.3.3 Normalización de características

La razón principal para normalizar los datos es que muchas técnicas de aprendizaje automático, especialmente aquellas que implican medidas de distancia (como SVM, KNN y regresión logística), funcionan mejor cuando los datos están en la misma escala. La normalización mejora la convergencia de los algoritmos y la precisión de los modelos.

Se utiliza Standard Scaler que es una herramienta de la biblioteca scikit-learn en Python que se utiliza para estandarizar características. Este método transforma las características para que tengan una media de 0 y una desviación estándar de 1. Esto se logra restando la media y dividiendo por la desviación estándar de cada característica.

Primero, el Standard Scaler se ajusta a los datos. Durante este ajuste, se calculan la media y la desviación estándar de cada característica en el conjunto de datos. Una vez ajustado, el Standard Scaler transforma los datos aplicando la fórmula.

$$z = \frac{x - \mu}{\sigma}$$

Dónde  $x$  es el valor original,  $\mu$  es la media y  $\sigma$  es la desviación estándar. Esta transformación asegura que todas las características tengan una media de 0 y una desviación estándar de 1 [35].

Con esto se consigue mejora de la eficacia de los algoritmos al converger más rápido, comparabilidad de las características no domina ninguna sobre el modelo y estabilidad numérica en los cálculos.

### 3.4 Algoritmos de aprendizaje automático utilizados

#### 3.4.1 Logistic Regression

Una técnica fundamental en el análisis estadístico y el aprendizaje automático [36]. A diferencia de la Regresión Lineal, que establece relaciones cuantitativas entre variables, la Regresión Logística se centra en problemas de clasificación, prediciendo la probabilidad de que una observación pertenezca a una categoría específica. Su aplicación es crucial en campos como la ciencia de datos, la medicina y la investigación de mercado. La Regresión Logística se basa en la función sigmoide como se puede ver en la figura 1, que transforma la salida en un rango entre 0 y 1, facilitando la clasificación binaria.

Utiliza el método de descenso de gradiente para optimizar los coeficientes del modelo, minimizando una función de error y ajustando los parámetros para mejorar la precisión de la predicción. Esta técnica es especialmente útil cuando



las relaciones entre variables son no lineales y se requiere una clasificación clara de los datos.

**Figura 1**

$$f(x) = \frac{1}{1 + e^{-x}}$$

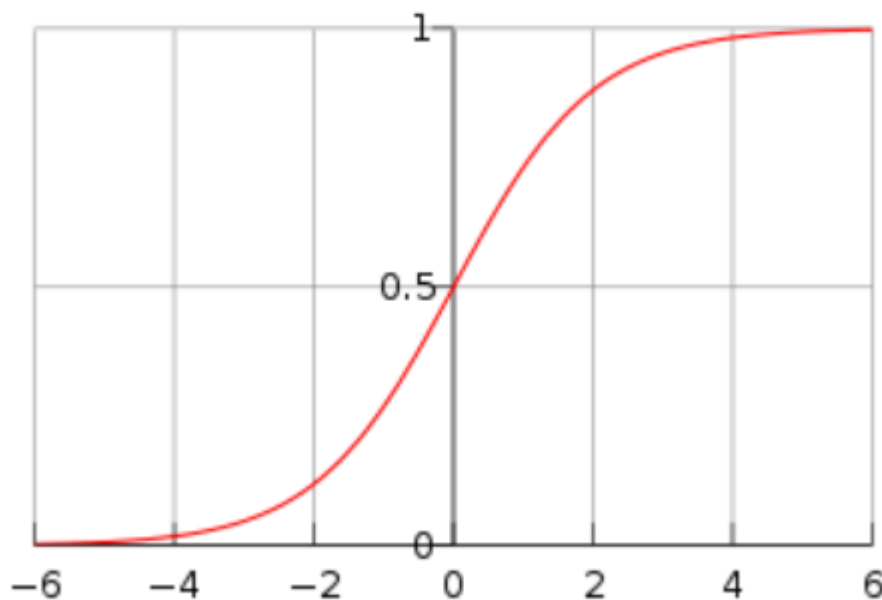


Figura 1: Función sigmoide [37]

Un ejemplo práctico propuesto por Zou, Hu, Tian y Shen en [38] de aplicación de la Regresión Logística es su uso en la evaluación de vehículos para predecir si los consumidores aceptarán un determinado automóvil. Se emplean conjuntos de datos de entrenamiento y prueba para verificar la precisión del modelo, lo que demuestra su eficacia en la predicción de resultados binarios. La comprensión de la Regresión Logística y su metodología subyacente es esencial para aquellos que deseen utilizar esta técnica en sus análisis y proyectos de aprendizaje automático.

En conclusión, la Regresión Logística es una herramienta poderosa y versátil en el análisis de datos y el aprendizaje automático. Su capacidad para modelar relaciones no lineales y predecir probabilidades la convierte en una técnica invaluable en una amplia gama de aplicaciones, desde la investigación médica hasta la toma de decisiones empresariales [39].

Explorar técnicas avanzadas dentro de la Regresión Logística puede mejorar aún más la precisión y la robustez de los modelos, permitiendo un análisis más profundo y preciso de los datos.

### 3.4.2 Perceptrón multicapa

El perceptrón multicapa (MLP, por sus siglas en inglés, *Multilayer perceptron*) es un modelo fundamental dentro de las redes neuronales artificiales.

Se caracteriza por su capacidad de procesar datos a través de una estructura de múltiples capas de neuronas, las cuales están organizadas en una secuencia específica: una capa de entrada, una o más capas ocultas y una capa de salida como se puede ver en la figura 2.

**Figura 2**

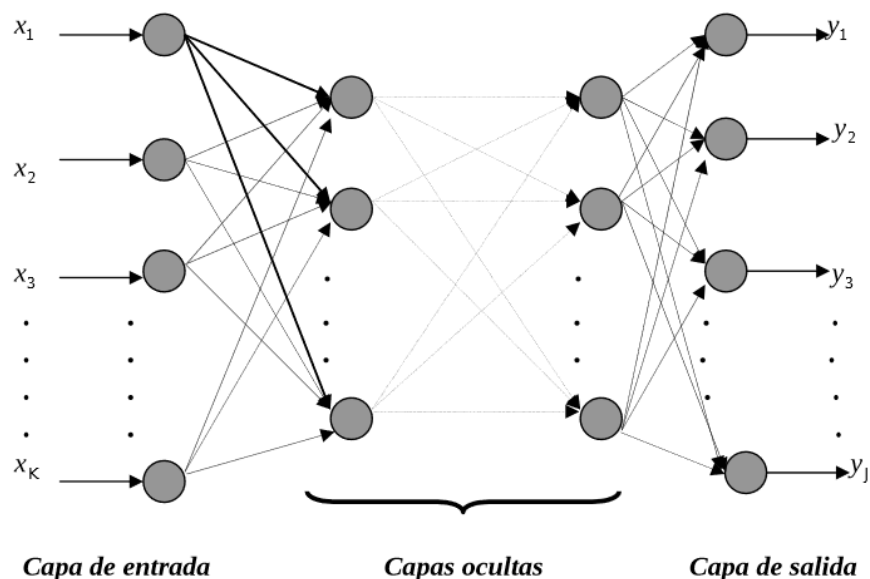


Figura 2: Perceptrón multicapa [40]

En un MLP, la capa de entrada recibe las señales iniciales y las transmite sin alterarlas a la primera capa oculta. Las capas ocultas, a su vez, son cruciales porque las neuronas en estas capas aplican funciones de activación no lineales, como la función sigmoide o la tangente hiperbólica. Estas funciones son esenciales porque introducen no linealidad, permitiendo al MLP capturar y modelar relaciones complejas en los datos [41].

El proceso de aprendizaje de un MLP se realiza mediante el algoritmo de retropropagación, que es un método supervisado. Este algoritmo ajusta los pesos de las conexiones neuronales para minimizar la diferencia entre la salida real de la red y la salida deseada. Durante el entrenamiento, las señales se propagan hacia adelante a través de la red para producir una salida. Luego, el error entre esta salida y la salida esperada se calcula y se propaga hacia atrás, ajustando los pesos en función del gradiente de las funciones de activación. Este proceso se repite hasta que el error se reduce a un nivel aceptable.

Popescu, Balas, Perescu-Popescu y Mastorakis en [42] proporcionaron una explicación detallada sobre la estructura y el funcionamiento de los MLP, destacando la importancia de las funciones de activación no lineales y el algoritmo de retropropagación para el entrenamiento efectivo de estas redes. Para mejorar la eficiencia del entrenamiento, se pueden utilizar técnicas como el método del momento y la tasa de aprendizaje variable. El método del momento ayuda a suavizar las actualizaciones de los pesos y a prevenir oscilaciones, mientras que la tasa de aprendizaje variable ajusta la velocidad de aprendizaje en función del comportamiento del error, acelerando el proceso cuando el error decrece consistentemente y desacelerándolo cuando el error comienza a aumentar.

Este modelo ha demostrado ser muy eficaz en diversas aplicaciones, como la clasificación, la regresión y el reconocimiento de patrones, gracias a su habilidad para capturar la complejidad intrínseca de los datos [43].

### 3.4.3 SVM

Como señalan Wang y Hu [44] las máquinas de vectores de soporte (SVM) fueron inicialmente diseñadas para la clasificación y la estimación de funciones no lineales, y desde entonces han sido ampliamente investigadas y aplicadas en varios campos. La técnica SVM para regresión se plantea como un problema de optimización convexa, en concreto, un problema de programación cuadrática (QP).

Este enfoque emplea la función de pérdida insensible de Vapnik y formula el problema de aproximación como un problema de optimización con restricciones de desigualdad, aprovechando la condición de Mercer para relacionar el mapeo del espacio de características no lineal con la función kernel seleccionada.

El método SVM es muy efectivo para gestionar espacios de entrada de alta dimensión. No obstante, uno de sus mayores inconvenientes es la alta carga computacional derivada de la programación de optimización requerida. Para abordar este problema, se desarrolló una versión de SVM basada en mínimos cuadrados (LS-SVM). En las LS-SVM, se utilizan restricciones de igualdad en lugar de desigualdad y se emplea una función de costo de error cuadrático (SSE), lo que simplifica notablemente el problema y permite que la solución se caracterice por un sistema lineal.

En la tarea de modelar conjuntos de datos complejos, nos encontramos con la necesidad de emplear técnicas que vayan más allá de la simple regresión lineal, las SVM ofrecen una solución poderosa al permitirnos aproximar conjuntos de datos utilizando funciones no lineales. Este enfoque es especialmente útil cuando los datos exhiben patrones complejos que no pueden ser capturados por una función lineal simple.

En esencia, la idea detrás de SVM es encontrar una función que pueda ajustarse a nuestros datos de manera precisa pero también generalizable. Esto se logra mediante la introducción de una función no lineal que mapea los datos a un espacio de características de mayor dimensión. Este mapeo nos permite

encontrar un hiperplano en este espacio que maximice el margen entre las clases de datos.

El proceso de optimización de las SVM implica minimizar una función objetivo que combina la norma del vector de pesos con una penalización por errores de aproximación. Para manejar desviaciones de los datos, se emplea una función de pérdida epsilon-insensible. Esta función de pérdida permite tolerar cierto grado de error en las predicciones, lo que hace que las SVM sean robustas ante datos ruidosos o atípicos.

Una característica distintiva de las SVM es su capacidad para generar soluciones dispersas, donde solo un subconjunto de los datos de entrenamiento, denominados vectores de soporte contribuyen significativamente a la definición del modelo. Estos vectores de soporte se encuentran en las fronteras de decisión entre las diferentes clases o, en el caso de la regresión, en las regiones donde los datos tienen una mayor influencia en la estimación de la función.

En resumen, las SVM son una herramienta poderosa para la estimación de funciones no lineales, que aprovecha la idea de mapear los datos a un espacio de características de alta dimensión y utiliza técnicas de optimización para encontrar una solución robusta y dispersa que pueda generalizar bien a nuevos datos [45].

Además, es importante destacar que las SVM ofrecen flexibilidad en la elección del kernel, lo que permite adaptarse a diferentes tipos de datos y problemas. Los kernels más comunes incluyen el kernel lineal, que es efectivo para problemas linealmente separables; el kernel polinomial, que introduce no linealidades adicionales mediante la elevación de los datos a potencias; el kernel de función de base radial (RBF), que utiliza una función de similitud gaussiana para capturar relaciones no lineales más complejas; y el kernel de perceptrón multicapa, que emplea una función de activación no lineal para modelar datos altamente no lineales.

#### 3.4.4 Random Forest

El algoritmo Random Forest, creado por Breiman, según Rigatti [46] representa un avance en los modelos de árboles de clasificación y regresión (CART). A diferencia de los modelos CART, que son simples, pero no muy adecuados para datos complejos, Random Forest utiliza la aleatorización para generar numerosos árboles de decisión, que luego se combinan para obtener una predicción más precisa.

Esta aleatorización se aplica tanto al conjunto de datos, mediante el muestreo de bootstrap, como a la selección de características en cada nodo de decisión. Una de las grandes ventajas de Random Forest es su capacidad para descubrir interacciones complejas y efectos no lineales entre las variables, sin necesidad de especificarlos previamente. Una vez construido el modelo, este puede usarse para hacer predicciones evaluando cada árbol y combinando sus resultados mediante votación o promedio, dependiendo del problema.

Aunque el modelo final puede resultar complejo y difícil de interpretar, existen medidas matemáticas para evaluar la importancia de las variables en el modelo, como el coeficiente Gini o pruebas de exclusión de variables.

Random Forest ha demostrado ser útil en una amplia gama de aplicaciones, desde la investigación médica hasta análisis de grandes conjuntos de datos [46]. En la práctica médica, se ha utilizado para predecir riesgos de salud, como el riesgo de accidente cerebrovascular o el inicio del Alzheimer, así como para automatizar tareas como la diferenciación entre tipos de lesiones cutáneas o la detección temprana de ciertos tipos de cáncer.

En la figura 3 podemos ver un ejemplo de un árbol de clasificación utilizado para predecir la supervivencia a bordo del Titanic. Cada nodo del árbol representa una pregunta sobre las características de los pasajeros, como el sexo, la edad o la clase de pasajero.

Dependiendo de la respuesta a esa pregunta, el árbol se divide en ramas que representan diferentes posibilidades. Las cajas sombreadas indican grupos de

pasajeros con una alta probabilidad de mortalidad, basándose en la información recopilada durante el hundimiento del Titanic. Esto sugiere que, según las características de los pasajeros, el modelo predice si una persona sobrevivió o no al desastre.

Para cada punto de decisión, la rama izquierda representa una respuesta "Sí" al criterio de decisión. Las cajas sombreadas indican una mortalidad de >50% para ese grupo, por lo que, en el conjunto de pruebas, este modelo predeciría la muerte para ese grupo.

**Figura 3**

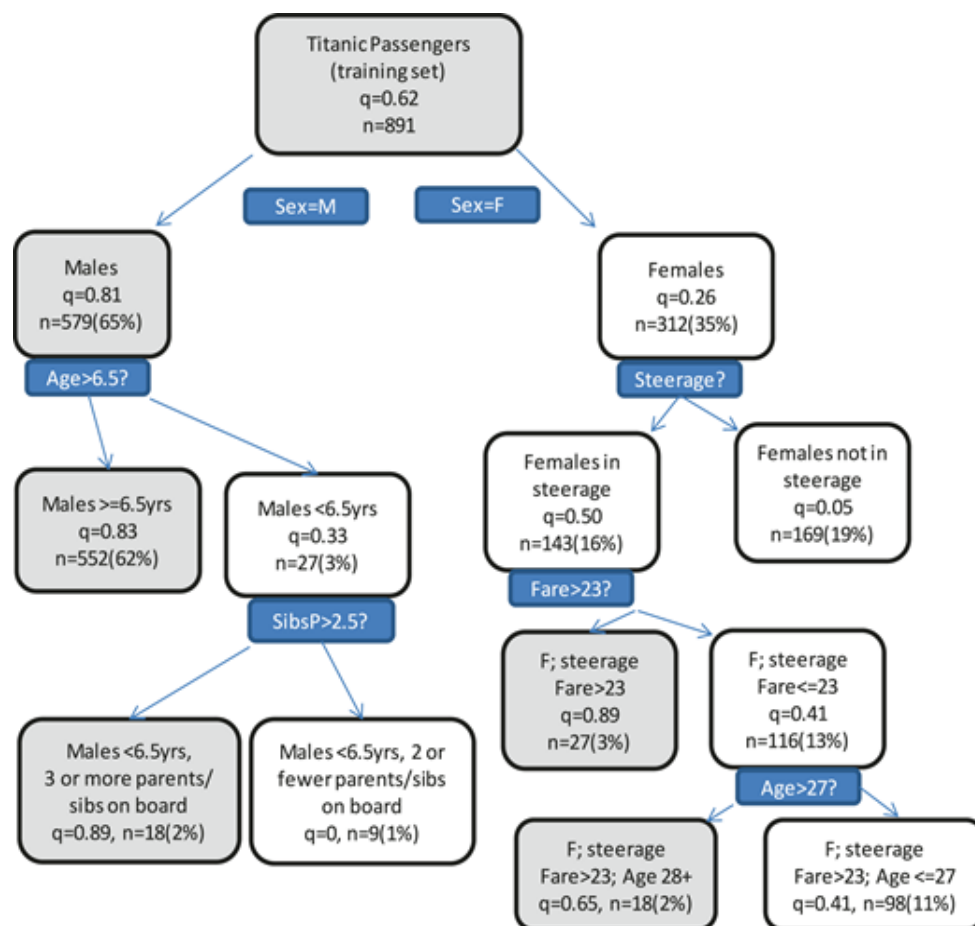


Figura 3: Árbol de clasificación para determinar la supervivencia a bordo del RMS Titanic. Fuente: Rigatti, S. J. (2017) [46]

### 3.4.5 XGBoost

El modelo XGBoost (eXtreme Gradient Boosting) es una implementación avanzada del método de boosting de árboles de decisión, reconocida por su alta eficiencia y escalabilidad en aplicaciones de aprendizaje automático. XGBoost se utiliza ampliamente en competencias de ciencia de datos y en la industria debido a su capacidad para manejar grandes volúmenes de datos y ofrecer resultados de alta precisión [47].

XGBoost es un sistema de boosting de árboles que mejora iterativamente las predicciones de los modelos anteriores al agregar nuevos árboles que corrigen los errores de predicción. Este enfoque se basa en el método de boosting, donde múltiples modelos débiles se combinan para formar un modelo fuerte. Utiliza una serie de árboles de decisión aditivos para realizar predicciones. Cada árbol se construye para corregir los errores de los árboles anteriores, mejorando gradualmente la precisión del modelo. XGBoost optimiza una función objetivo regularizada, lo que ayuda a evitar el sobreajuste. La regularización controla la complejidad del modelo, asegurando un equilibrio entre el ajuste a los datos de entrenamiento y la capacidad de generalización del modelo.

Implementa el algoritmo de gradient boosting, que ajusta los árboles para minimizar los residuos de los modelos anteriores utilizando técnicas de gradiente. Incluye un algoritmo consciente de la escasez de datos, lo que permite manejar eficientemente datos con muchos valores faltantes o nulos. Utiliza un procedimiento de sketch de cuantiles ponderados para manejar pesos de instancias en el aprendizaje aproximado de árboles, optimizando la selección de divisiones en los árboles. Está diseñado para aprovechar la computación paralela y distribuida, acelerando significativamente el proceso de entrenamiento. Esto permite explorar rápidamente diferentes configuraciones de modelos y manejar grandes conjuntos de datos. Además, puede realizar cálculos fuera de núcleo, permitiendo procesar cientos de millones de ejemplos utilizando recursos de computación limitados. Esta característica es crucial para escalar el aprendizaje a grandes volúmenes de datos sin necesidad de grandes cantidades de memoria RAM.



Entre las principales innovaciones de XGBoost se destacan el algoritmo consciente de la escasez, que optimiza el aprendizaje para datos con muchas celdas vacías o valores faltantes; el sketch de cuantiles ponderados, que mejora la eficiencia en el cálculo de propuestas de divisiones en los árboles; y la estructura de bloques consciente de la caché, que optimiza el aprendizaje fuera de núcleo, utilizando eficientemente los recursos de almacenamiento y memoria caché. XGBoost ha demostrado ser altamente efectivo en una amplia gama de problemas, incluyendo la predicción de ventas en tiendas, clasificación de eventos en física de alta energía, clasificación de textos web y predicción del comportamiento del cliente, entre otros.

Como señalan Chen y Guestrin en [48], XGBoost ha sido el modelo preferido en numerosas competencias de Kaggle y ha sido fundamental para muchos equipos ganadores en competencias como el KDDCup.

En resumen, XGBoost es un sistema de boosting de árboles altamente escalable y eficiente que se destaca por su capacidad para manejar grandes conjuntos de datos y producir modelos de alta precisión, convirtiéndose en una herramienta indispensable para científicos de datos y profesionales del aprendizaje automático.

#### **3.4.6 LightBM**

El algoritmo LightGBM, desarrollado por Microsoft, es una versión optimizada del método de Gradient Boosting Decision Tree (GBDT) [49]. Este algoritmo está diseñado para ser eficiente y escalable, especialmente cuando se trata de manejar grandes volúmenes de datos. LightGBM destaca por su capacidad para gestionar características categóricas y su eficiencia en términos de memoria y velocidad de entrenamiento. En este informe, se ofrece una visión general del funcionamiento teórico de LightGBM, sus ventajas y un caso de uso específico en la detección de fraudes en clics en línea.

LightGBM actúa como una técnica de ensamblaje que crea un modelo robusto a partir de la combinación de múltiples modelos más débiles, concretamente árboles de decisión. Utiliza un enfoque basado en el método de Boosting,

donde cada nuevo árbol se construye para corregir los errores de los árboles anteriores. Emplea un método de crecimiento de árboles denominado "leaf-wise", a diferencia del crecimiento "depth-wise" tradicional. En el crecimiento leaf-wise, el algoritmo selecciona las hojas con mayor pérdida y las expande, lo que conduce a una mayor precisión y optimización. Sin embargo, este enfoque puede resultar en sobreajuste si no se maneja adecuadamente.

Para mejorar la eficiencia en el uso de la memoria, LightGBM convierte los valores continuos en valores discretos mediante el uso de binning. Esto no solo reduce el uso de memoria, sino que también acelera el proceso de entrenamiento. LightGBM ofrece varios parámetros que permiten la optimización del modelo. Algunos de los parámetros clave incluyen el número de hojas por árbol (`num_leaves`), la tasa de aprendizaje del algoritmo (`learning_rate`), la profundidad máxima de los árboles (`max_depth`), el número mínimo de datos en una hoja (`min_data_in_leaf`) y la proporción de características utilizadas en cada iteración (`feature_fraction`).

LightGBM admite tanto el paralelismo de características como el paralelismo de datos, lo que permite una aceleración significativa en el entrenamiento cuando se trabaja con grandes conjuntos de datos. Además, ofrece soporte para GPU, aumentando aún más la velocidad de entrenamiento. Según estudios realizados por Xiaojun et al. [50] y Wei Min et al. [51], LightGBM presenta varias ventajas significativas. La alta velocidad de entrenamiento se debe a su enfoque de binning y paralelismo, mientras que el bajo consumo de memoria se logra mediante la conversión de valores continuos a discretos. La estrategia de crecimiento leaf-wise permite que LightGBM genere árboles más complejos y precisos, y su capacidad para manejar grandes volúmenes de datos lo hace ideal para aplicaciones a gran escala.

En el artículo de Minastireanu y Mesnita [52], se utilizó LightGBM para detectar fraudes en clics en línea. El estudio se centró en un conjunto de datos públicos que maneja 200 millones de clics durante cuatro días. Los autores realizaron un análisis de características para identificar patrones y desarrollaron un modelo utilizando LightGBM, logrando una precisión del 98%. Este caso de uso

demuestra la efectividad de LightGBM en la identificación de actividades fraudulentas en grandes volúmenes de datos.

En conclusión, LightGBM es un algoritmo potente y eficiente para la clasificación y regresión, especialmente cuando se trabaja con grandes conjuntos de datos. Sus características de optimización de memoria, velocidad de entrenamiento y alta precisión lo hacen ideal para aplicaciones en diversas industrias. Como señalan Minastireanu y Mesnita (2019), LightGBM ha demostrado ser una herramienta eficaz en la detección de fraudes en clics en línea, destacando su relevancia en el ámbito de la publicidad digital.

### **3.5 Algoritmos y técnicas de interpretabilidad utilizadas**

Todas las gráficas y salidas de las ejecuciones del código se verán en detalle con ejemplos para los distintos modelos, ataques y características en el apartado de análisis de resultados (5), aunque en este apartado se comenten las gráficas y salidas a nivel general como se pueden interpretar y que es lo que muestran.

#### **3.5.1 Análisis de residuos**

Los residuos, en este contexto, son la diferencia entre los valores reales y los valores predichos por el modelo. Analizar estos residuos puede proporcionar información valiosa sobre cómo se desempeña el modelo y dónde podría estar fallando [14].

El principal objetivo del análisis de residuos es identificar patrones en los errores de predicción del modelo. Si los residuos se distribuyen aleatoriamente, sugiere que el modelo se ajusta bien a los datos. Sin embargo, si hay patrones sistemáticos en los residuos, podría indicar problemas en el modelo, como un ajuste insuficiente o excesivo, la presencia de outliers (datos atípicos), o la necesidad de transformar algunas características.

En el código se realiza un análisis de residuos siguiendo estos pasos generales:

**1. Codificación de etiquetas:**

- Transforma las etiquetas del conjunto de test utilizando LabelEncoder, lo cual es necesario para trabajar con valores numéricos en el análisis de residuos.

**2. Carga del modelo:**

- Carga el modelo calibrado correspondiente en función del nombre del modelo. Esto asegura que el modelo adecuado se utilice para las predicciones.

**3. Predicciones de probabilidad:**

- En lugar de predecir clases, el modelo realiza predicciones de probabilidad de pertenencia a la clase positiva. Esto es importante para el cálculo de residuos.

**4. Cálculo de residuos:**

- Calcula los residuos como la diferencia entre las etiquetas codificadas reales y las probabilidades predichas.

**5. Visualización de residuos:**

- Se generan varios gráficos para analizar los residuos:
  - **Gráfico de dispersión:** muestra la relación entre las probabilidades predichas y los residuos (menos representativo).
  - **Histograma:** ilustra la distribución de los residuos.
  - **Gráfico de densidad de Kernel:** proporciona una estimación de la densidad de probabilidad de los residuos.

El análisis de residuos es crucial por varias razones [53]:

- **Evaluación del ajuste del modelo:** permite evaluar si el modelo se ajusta adecuadamente a los datos. Residuos grandes o sistemáticamente sesgados pueden indicar un mal ajuste.
- **Identificación de outliers:** ayuda a identificar datos atípicos que podrían estar afectando el rendimiento del modelo.
- **Validación de suposiciones:** verifica si se cumplen las suposiciones del modelo, como la homocedasticidad (varianza constante de los residuos) y la independencia de errores.

El histograma de residuos muestra la distribución de los residuos. Es útil para evaluar si los residuos siguen una distribución normal, lo cual es un supuesto clave en muchos modelos estadísticos. La forma del histograma puede indicar sesgo o problemas en el modelo, una distribución centrada en cero y simétrica sugiere que el modelo está bien ajustado.

Si el histograma muestra residuos sesgados hacia valores positivos o negativos, puede indicar que el modelo tiende a sobreestimar o subestimar las probabilidades, al igual con el gráfico de densidad de kernel.

### 3.5.2 Análisis de influencia

Este análisis es fundamental para identificar puntos de datos que tienen un impacto significativo en las métricas de rendimiento y, por ende, en la robustez del modelo [15]. A continuación, se presenta una descripción general del propósito y proceso del análisis de influencia.

El principal objetivo del análisis de influencia es determinar la sensibilidad del modelo a la eliminación de un subconjunto de datos del conjunto de prueba. Al hacer esto, se pueden identificar datos que tienen una alta influencia en el desempeño del modelo.

Esto es especialmente útil para:

- Detectar y mitigar la dependencia del modelo en datos específicos.
- Identificar instancias que podrían estar sesgando los resultados.
- Evaluar la estabilidad y generalización del modelo.

En el código realiza el análisis de influencia siguiendo los pasos generales que se describen a continuación:

### 1. Predicciones y métricas originales:

- **Predicción:** se utilizan los datos de prueba completos para realizar predicciones utilizando el modelo dado.
- **Cálculo de métricas:** se calculan varias métricas de rendimiento, como la exactitud (accuracy), precisión (precision), recall y F1-score. Estas métricas proporcionan una línea base de rendimiento del modelo con el conjunto de datos completo.

#### **Recall (Sensibilidad)**

El Recall, también conocido como Sensibilidad o Tasa de Verdaderos Positivos, mide la capacidad de un modelo para identificar todos los ejemplos positivos en el conjunto de datos [54]. Se calcula como:

$$Recall = \frac{Verdaderos\ Positivos\ (TP)}{Verdaderos\ Positivos\ (TP) + Falsos\ negativos\ (FN)}$$

El Recall es particularmente importante en situaciones donde la detección de positivos es crucial, como en diagnósticos médicos o detección de fraudes.

#### **F1-Score**

El F1-Score es una medida de precisión utilizada en la clasificación que combina la precisión (Precision) y el Recall (Sensibilidad) en una única métrica [54]. Es la media armónica de estas dos medidas y se calcula como:

$$F1 - Score = 2 \times \left( \frac{Precisión \times Recall}{Precisión + Recall} \right)$$

Donde la precisión es la proporción de verdaderos positivos sobre el total de positivos predichos, y el Recall es la proporción de verdaderos positivos sobre el total de positivos reales. El F1-Score es útil cuando se necesita un equilibrio entre precisión y Recall y hay una distribución desigual de clases.

## 2. Exclusión de datos:

- **Eliminación de instancias:** se excluye un rango específico de instancias del conjunto de datos de prueba.
- **Predicciones con datos excluidos:** se realizan predicciones utilizando el conjunto de datos modificado (con las instancias excluidas).

## 3. Cálculo de métricas con datos excluidos:

- **Métricas de rendimiento:** se vuelven a calcular las métricas de rendimiento para el conjunto de datos modificado. Estas métricas reflejan el desempeño del modelo sin las instancias excluidas.

## 4. Comparación de métricas:

- **Comparación visual:** se comparan las métricas originales y las métricas después de la exclusión mediante visualizaciones gráficas (barras).
- **Análisis de la influencia:** al observar las diferencias en las métricas, se puede evaluar cuánto influyen las instancias excluidas en el rendimiento del modelo.

Las gráficas generadas en este análisis permiten visualizar y comparar fácilmente el rendimiento del modelo antes y después de la exclusión de los datos.

- Dos conjuntos de barras se presentan para cada métrica. Unas barras representan las métricas calculadas con el conjunto de datos completo, mientras y otras representan las métricas calculadas después de excluir las instancias especificadas.

- Cambios significativos: si se observan cambios significativos en las métricas, indica que las instancias excluidas tenían una alta influencia en el rendimiento del modelo. Esto puede sugerir la presencia de datos atípicos o puntos de datos que el modelo está sobreajustando.
- Cambios menores: si las métricas no cambian significativamente, se puede concluir que el modelo es robusto y generaliza bien, sin depender excesivamente de un subconjunto específico de datos.

### 3.5.3 Análisis de curva ROC

La curva ROC es una técnica esencial para evaluar y comparar modelos de clasificación binaria. Representa gráficamente la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos en diferentes umbrales de decisión, ayudando a identificar el equilibrio óptimo entre sensibilidad y especificidad [55].

- **Definición y propósito:** empleada para determinar la eficacia de un modelo de clasificación, la curva ROC es clave para identificar el balance óptimo.
- **Área bajo la Curva (AUC):** el área bajo la curva (AUC) es un indicador cuantitativo del rendimiento del modelo. Un AUC cercano a 1 indica un modelo excelente, mientras que un AUC de 0.5 sugiere un rendimiento aleatorio.
- **Aplicaciones:** se utiliza en campos como la medicina y la detección de fraudes, donde es crucial evaluar la precisión de los modelos predictivos.
- **Comparación de Modelos:** permite comparar múltiples modelos de clasificación, facilitando la elección del más adecuado para el contexto del problema.
- **Selección de Puntos de Corte:** optimiza el rendimiento del modelo, ajustando las prioridades entre sensibilidad y especificidad según las necesidades específicas.



### *Funcionamiento General:*

- 1) **Preparación del Gráfico:** Configura el gráfico para mostrar las curvas ROC de todos los modelos.
- 2) **Cálculo de Curvas ROC:** Para cada modelo, calcula las tasas de verdaderos y falsos positivos.
- 3) **Trazado y Configuración:** Dibuja las curvas en el gráfico, agrega etiquetas y una línea diagonal de referencia.
- 4) **Visualización:** Muestra el gráfico con todas las curvas ROC para comparar el rendimiento de los modelos.

Esta visualización ayuda a identificar qué modelo tiene el mejor rendimiento general en términos de clasificación.

#### **3.5.4 Análisis de sensibilidad**

El análisis de sensibilidad es una técnica utilizada para determinar cómo la variación en los valores de las características de entrada de un modelo afecta las predicciones del modelo. Al evaluar cómo las predicciones del modelo cambian con la variación de una característica específica, se puede identificar las características más influyentes y detectar posibles puntos débiles en el modelo [16].

El objetivo principal del análisis de sensibilidad es identificar cómo las predicciones del modelo cambian en respuesta a las alteraciones en una característica específica, se pueden ver sus distintas aplicaciones y lo que aporta en [56].

Esto ayuda a:

- **Identificar características críticas:** determinar cuáles características tienen el mayor impacto en las predicciones del modelo.
- **Evaluar la robustez del modelo:** evaluar cómo de sensible es el modelo a los cambios en los datos de entrada.

- **Guiar la interpretación del modelo:** proporcionar información sobre la relación entre las características y las predicciones del modelo, lo cual es crucial para la interpretabilidad.

El análisis de sensibilidad en el código se realiza mediante el siguiente proceso:

- **Selección de la característica:** se elige una característica específica para analizar su impacto en las predicciones del modelo.
- **Generación de rango de valores:** se genera un rango de valores aleatorios posibles para la característica seleccionada, desde su mínimo hasta su máximo valor observado en los datos de prueba.
- **Modificación de datos de prueba:** para cada valor en el rango generado, se modifica el conjunto de datos de prueba, asignando el valor actual a la característica seleccionada mientras se mantienen constantes las demás características.
- **Predicciones del modelo:** se realizan predicciones con el modelo para cada conjunto de datos modificado.
- **Cálculo de predicciones medias:** se calcula la predicción media para cada valor de la característica modificada. En este caso, la predicción media se refiere a la probabilidad predicha de la clase positiva ('DDoS').
- **Visualización de resultados:** se grafica la predicción media en función de los valores de la característica, mostrando cómo varía la probabilidad predicha de la clase positiva con cambios en la característica.

La gráfica muestra la influencia que tiene una característica específica y si es significativa o no en las predicciones del modelo. La variación en la predicción media podría sugerir que el modelo depende en gran medida de esta característica para diferenciar entre las clases.

Podría pasar que hubiese una relación negativa entre el valor de la característica y la probabilidad predicha de la clase positiva (a medida que el valor de la característica aumenta, la probabilidad de que el modelo prediga la clase positiva disminuye).

Estos resultados pueden ser utilizados para entender mejor cómo el modelo utiliza las características para hacer predicciones. Por ejemplo, en un contexto específico, se podría investigar más a fondo por qué ciertos valores de la característica están asociados con una alta probabilidad de una determinada clase y ajustar las estrategias de análisis y respuesta en consecuencia.

Mediante la modificación sistemática de los valores de las características y la observación de los cambios en las predicciones, este análisis proporciona información crucial para la interpretación, evaluación y mejora del modelo.

Aunque el conjunto de datos esté normalizado, al ser un conjunto de datos tan grande hay algunas características que sus valores mínimos y máximos son algo dispares, puede ser por valores atípicos también, en estos casos el análisis de sensibilidad es algo complejo de interpretar y no es muy ilustrativo en la gráfica. Por varias razones:

- **Rango extenso:** un rango de valores muy extenso puede hacer que las variaciones en la predicción media sean menos evidentes o más difíciles de interpretar en una gráfica.
- **Poca relevancia de valores extremos:** los valores extremos pueden no ser representativos de la mayoría de los datos y pueden dar lugar a conclusiones engañosas sobre la sensibilidad del modelo a esa característica.

### 3.5.5 Permutación de características

Una de las preguntas fundamentales en el análisis de modelos de aprendizaje automático es: ¿qué características tienen el mayor impacto en las predicciones del modelo? Esta pregunta se aborda a través del concepto de importancia de características, que mide la relevancia de cada característica en la capacidad del modelo para hacer predicciones precisas.

Existen múltiples métodos para medir la importancia de características, cada uno con sus ventajas y desventajas. Algunos enfoques proporcionan diferentes perspectivas sobre la importancia de las características, y algunos tienen

limitaciones documentadas. En este apartado, se enfoca en la importancia de la permutación, un método que destaca por ser rápido de calcular, ampliamente comprendido y consistente con las propiedades deseables de una medida de importancia de características.

La importancia de la permutación evalúa la importancia de las características al medir cómo se deteriora el rendimiento del modelo cuando los valores de una característica específica se barajan aleatoriamente [17]. La idea subyacente es que, si una característica es crucial para las predicciones del modelo, el barajado de sus valores debería llevar a una disminución significativa en la precisión del modelo. Por otro lado, si una característica tiene poca o ninguna importancia, su barajado no afectará significativamente las predicciones.

- **Permutación de columnas:** se selecciona una columna del conjunto de datos de prueba y se barajan aleatoriamente sus valores, manteniendo las demás columnas y la variable objetivo intactas.
- **Predicciones con datos barajados:** se realizan predicciones usando el conjunto de datos con la columna barajada.
- **Evaluación del deterioro del rendimiento:** se compara el rendimiento del modelo antes y después del barajado usando una métrica como la precisión.
- **Características importantes:** las que causan una gran disminución en el rendimiento al ser barajadas.
- **Características menos importantes:** las que causan poca o ninguna disminución en el rendimiento.
- **Valores negativos:** ocasionalmente, los valores negativos indican que las predicciones con datos barajados fueron, por azar, más precisas que con los datos reales. Esto es más común en conjuntos de datos pequeños debido a la mayor variabilidad aleatoria.

La permutación de características tiene varios beneficios:

- **Simplicidad y rapidez:** es un método sencillo y rápido de calcular, adecuado para una rápida evaluación de la importancia de las características.

- **Consistencia:** proporciona resultados consistentes y es ampliamente comprendido en la comunidad de aprendizaje automático.
- **Aplicabilidad general:** puede aplicarse a una amplia variedad de modelos y tipos de datos.

#### Limitaciones

- **Dependencia de la aleatoriedad:** los resultados pueden variar debido a la aleatoriedad inherente en el proceso de barajado, especialmente en conjuntos de datos pequeños.
- **Impacto del tamaño del conjunto de datos:** en conjuntos de datos pequeños, la variabilidad aleatoria puede afectar más los resultados, lo que puede llevar a interpretaciones incorrectas (no es el caso en el conjunto de datos utilizado).

La permutación de características es una herramienta poderosa y versátil para evaluar la importancia de las características en un modelo de aprendizaje automático. Permite identificar características clave, mejorar la interpretabilidad del modelo y optimizar su rendimiento. Al comprender la importancia relativa de cada característica, los científicos de datos pueden construir modelos más eficientes, precisos y comprensibles, lo cual es crucial para la toma de decisiones informada y la explicación de los resultados del modelo [17].

#### 3.5.6 LIME

LIME se centra en proporcionar interpretaciones locales, lo que significa que explica las predicciones de instancias individuales en lugar de proporcionar una visión global del modelo [19].

El propósito en el código es realizar un análisis de instancias y de características utilizando LIME para comprender mejor cómo el modelo toma decisiones para instancias específicas. Se realizan dos funciones principales:

**Análisis de instancias con LIME:** proporciona una explicación detallada de por qué el modelo tomó una decisión específica para una instancia particular.

**Análisis de importancia de características con LIME:** muestra la importancia de las características para una instancia concreta, destacando cuáles características influyeron más en la predicción del modelo.

El análisis de instancias con LIME se realiza mediante los siguientes pasos generales:

- 1) **Selección del modelo:** el código selecciona el modelo adecuado a partir de varios modelos calibrados, como Random Forest, lgb, XGBoost, mlp, y logisticRegression.
- 2) **Creación del explicador LIME:** se crea un explicador LIME (LimeTabularExplainer) que está ajustado al conjunto de entrenamiento y las características.
- 3) **Explicación de la instancia:** se genera una explicación para una instancia específica utilizando el modelo seleccionado. El explicador de LIME perturba la instancia y analiza cómo las perturbaciones afectan la predicción del modelo.
- 4) **Visualización de la explicación:** la explicación se muestra de una manera que sea fácilmente comprensible, a menudo en un notebook Jupyter.

El análisis de importancia de características se realiza de manera similar, pero con un enfoque en las características que más influyen en una predicción específica. Los primeros 3 pasos son los mismos al análisis de instancias con LIME (1) Selección del modelo, 2) Creación del explicador LIME, 3) Explicación de la instancia), y posteriormente se realizan los siguientes pasos:

- 4) **Extracción de pesos:** se extraen las características y sus pesos (importancia) de la explicación de LIME.

- 5) **Visualización de la importancia de las características:** se crea un gráfico de barras que muestra las características más importantes y sus pesos, proporcionando una visión clara de qué características influyeron más en la predicción.

La salida del análisis de LIME permite a los usuarios entender cómo el modelo utiliza las características para hacer predicciones específicas. Esto es particularmente útil en aplicaciones donde la interpretabilidad es crucial, como en la seguridad cibernética o la medicina [19],[21].

Comparado con otros métodos, LIME ofrece una perspectiva más detallada a nivel de instancia individual. Mientras que la dependencia de características y el análisis de sensibilidad, por ejemplo, proporcionan una visión global de cómo las características afectan las predicciones en general, LIME permite desglosar las predicciones para casos específicos, proporcionando explicaciones más granulares y localizadas.

Los resultados visuales proporcionados por LIME permiten interpretar de manera efectiva cómo cada característica influye en las predicciones del modelo para instancias individuales, a continuación, se explica de manera general la salida del análisis de una instancia con LIME.

1. **Gráfica de predicción para una instancia específica:**

- **Probabilidades de predicción:** la gráfica muestra las probabilidades predichas para cada clase (por ejemplo, BENIGN y DDoS), proporcionando una visión clara de la confianza del modelo en su predicción.
- **Contribución de características:** se desglosa la predicción mostrando cómo cada característica contribuye a la decisión del modelo. Las características se listan junto con sus valores específicos para la instancia, indicando si apoyan o contradicen la clase predicha.
- **Valor de las características:** los valores específicos de las características para la instancia proporcionan contexto sobre

cómo se comportan las características en relación con la predicción del modelo.

### **Beneficios adicionales de LIME**

Detección de sesgos y anomalías, como es obvio y es su objetivo mejora la interpretabilidad, facilita la validación de modelos y mejora comunicación con No-Expertos (aspecto clave) por la capacidad de desglosar y explicar las predicciones de manera comprensible facilita la comunicación de los resultados del modelo a partes interesadas que no son expertas en machine learning.

### **3.5.7 SHAP**

Los valores SHAP se utilizan para explicar la predicción de un modelo mostrando la contribución de cada característica a la predicción final. Los valores SHAP proporcionan una medida consistente y equitativa de la importancia de cada característica, sumando las contribuciones de cada característica para obtener la predicción final del modelo [18],[20].

### **Valores SHAP globales**

El propósito del código es calcular y visualizar los valores SHAP para varios modelos, permitiendo una comprensión detallada de cómo cada característica influye en las predicciones del modelo y dando una visión general de la importancia y el efecto de cada característica. El código maneja diferentes tipos de modelos, incluyendo modelos lineales, bosques aleatorios (Random Forest), LightGBM (lgb), XGBoost y regresión logística.

#### *Funcionamiento general del código para valores SHAP globales*

##### **1. Inicialización de SHAP:**

- `shap.initjs()`: inicializa la visualización interactiva de SHAP en un entorno Jupyter Notebook.

##### **2. Carga de valores SHAP:**

- Dependiendo del tipo de modelo se calculan los valores SHAP con una función distinta de SHAP y después se guardan.



### **3. Generación de gráficas resumen de SHAP:**

- `shap.summary_plot()`: esta función genera una gráfica resumen que visualiza los valores SHAP de manera global para todas las características del modelo. La gráfica muestra:
  - La importancia de cada característica (medida por la magnitud de los valores SHAP).
  - La distribución de los valores SHAP para cada característica.
  - Cómo cada característica afecta las predicciones del modelo.

#### *Interpretación de los valores SHAP globales*

##### **1. Importancia global de las características:**

- La gráfica resumen permite identificar cuáles características tienen la mayor influencia en las predicciones del modelo. Las características más importantes se listan en la parte superior de la gráfica.

##### **2. Distribución de los valores SHAP:**

- Cada punto en la gráfica representa un valor SHAP para una característica específica en una instancia del conjunto de datos de prueba.
- El color de los puntos indica el valor de la característica (por ejemplo, azul para valores bajos y rojo para valores altos).

##### **3. Efecto de las características en las predicciones:**

- La posición horizontal de los puntos muestra la magnitud y dirección del impacto de la característica en la predicción. Los puntos hacia la derecha indican un aumento en la predicción, mientras que los puntos hacia la izquierda indican una disminución.

### *Modelos soportados*

El código está diseñado para manejar diferentes tipos de modelos, cada uno con su propia manera de calcular los valores SHAP:

- **Modelos Lineales:** utilizan `shap.LinearExplainer`.
- **Random Forest:** utilizan `shap.TreeExplainer` y se centran en una clase específica (ejemplo, `clase = 0`).
- **LightGBM y XGBoost:** también utilizan `shap.TreeExplainer`, adecuado para modelos basados en árboles.
- **Regresión Logística:** utilizan `shap.LinearExplainer`.

### *Limitaciones*

Problemas con el resto de modelos, SVM no lineal, y redes neuronales (perceptrón multicapa) porque usan otra manera de calcular los valores de SHAP, lo hacen mediante el `shap.KernelExplainer` el cuál ha presentado muchos problemas de rendimiento siendo incapaz de calcular los valores para estos modelos.

### **Explicaciones SHAP locales**

Además de proporcionar una visión global de cómo las características influyen en las predicciones del modelo, SHAP también permite generar explicaciones locales para instancias específicas al igual que LIME. Esto es particularmente útil para entender por qué el modelo tomó una decisión particular en un caso concreto.

El código genera explicaciones locales utilizando valores SHAP para instancias específicas del conjunto de datos de prueba. Esto se realiza a través de visualizaciones interactivas que detallan cómo cada característica contribuye a la predicción para una instancia específica.

### *Funcionamiento general del código para explicaciones SHAP locales*

#### **1. Inicialización de SHAP:**

- `shap.initjs()`: inicializa la visualización interactiva de SHAP en un entorno Jupyter Notebook.

## **2. Carga del explicador y los valores SHAP:**

- Dependiendo del nombre del modelo, el código carga el explicador SHAP y los valores SHAP previamente calculados.

## **3. Generación de gráficas de explicación Local:**

- `shap.force_plot()`: esta función genera una gráfica de fuerza que visualiza la contribución de cada característica a la predicción de una instancia específica. La gráfica muestra:
  - El valor esperado (baseline) del modelo.
  - La contribución de cada característica a la desviación de la predicción respecto al valor esperado.
  - Una visualización acumulativa de cómo se llega a la predicción final.

### *Interpretación de la explicación local*

#### **1. Contribución de las características:**

- La gráfica de fuerza muestra cómo cada característica específica influye en la predicción para la instancia seleccionada. Las características que empujan la predicción hacia arriba (positivamente) y hacia abajo (negativamente) se visualizan claramente.

#### **2. Valor esperado y predicción:**

- El valor esperado es el promedio de las predicciones del modelo en el conjunto de datos de entrenamiento. La predicción final para la instancia específica se muestra como la suma del valor esperado y las contribuciones de las características.

#### **3. Interactividad y comprensión:**

- La visualización es interactiva, permitiendo a los usuarios explorar cómo pequeñas modificaciones en las características pueden afectar la predicción. Esto es útil para identificar características críticas y comprender la lógica detrás de la decisión del modelo.

### *Relación de Valores SHAP con Valores Reales*

En el código también se permite explorar la relación entre los valores SHAP y los valores reales de las características a través de gráficos de dependencia. Este análisis es crucial para comprender cómo una característica específica influye en la predicción del modelo a lo largo de su rango de valores y cómo interactúa con otras características.

La función principal utilizada es `shap.dependence_plot()`, que genera un gráfico de dependencia que visualiza la relación entre los valores SHAP de una característica seleccionada y sus valores reales.

El gráfico de dependencia muestra cómo los valores SHAP varían con respecto a los valores reales de la característica seleccionada, proporcionando una visualización clara de la importancia y el efecto de esa característica en la predicción del modelo.

Este análisis es útil para detectar patrones y tendencias en los datos, validar la consistencia del modelo y mejorar el diseño del modelo al identificar características y combinaciones de características que tienen un impacto significativo en las predicciones.

# 4

## Tecnologías utilizadas

En esta sección se presentan los recursos y entornos utilizados para desarrollar y probar los algoritmos de aprendizaje automático, incluyendo herramientas de programación, plataformas y bibliotecas.

### 4.1 Lenguaje de programación

Se eligió Python como lenguaje de programación principal para el desarrollo de este proyecto. Python es valorado por las comunidades de ciencia de datos y aprendizaje automático por su facilidad de uso, flexibilidad y amplia variedad de bibliotecas y herramientas especializadas [57]. La sintaxis clara y fácil de leer lo convierte en una opción ideal tanto para la enseñanza como para el desarrollo académico.

Además, Python es un lenguaje dinámico, interpretado, orientado a objetos y de código abierto, lo que lo hace altamente adaptable a una variedad de aplicaciones. En el contexto del aprendizaje automático y el análisis de datos, Python destaca por su capacidad para integrar bibliotecas especializadas, procesar grandes cantidades de datos y contar con una comunidad activa de desarrolladores que aportan constantemente nuevas herramientas y técnicas. Estas características hacen de Python una excelente opción para cualquiera que quiera desarrollar modelos predictivos y analizar datos de manera eficiente y precisa. La flexibilidad de Python y su amplia gama de recursos lo convierten en una herramienta poderosa y versátil para resolver una variedad de

problemas de ciencia de datos y aprendizaje automático. Su facilidad de uso y sintaxis intuitiva lo hacen accesible tanto para principiantes como para expertos, lo que lo convierte en un lenguaje de programación ideal para desarrollar soluciones innovadoras y eficientes.

## 4.2 Bibliotecas utilizadas

- **Matplotlib:** se utiliza para visualizar datos y le permite crear gráficos estáticos, animados e interactivos en Python [58].
- **NumPy:** brinda soporte para grandes conjuntos de datos y matrices y una variedad de funciones matemáticas para trabajar con estos datos [59].
- **Pandas:** se utiliza para la manipulación y el análisis de datos estructurados, proporcionando estructuras de datos flexibles y herramientas para trabajar con datos tabulares [60].
- **LIME:** explica todas las predicciones del clasificador y proporciona una interpretación local y transparente [61].
- **Scikit-learn (sklearn):** es una biblioteca de aprendizaje automático que incluye herramientas y algoritmos para clasificación, regresión y agrupación, así como herramientas para preprocesamiento de datos, entrenamiento y evaluación de modelos [62].
- **Joblib:** guarde y reutilice fácilmente modelos entrenados serializando y deserializando objetos Python grandes [63].
- **Seaborn:** esta es una biblioteca basada en Matplotlib que proporciona una interfaz de visualización de datos de alto nivel especialmente adecuada para crear atractivos gráficos estadísticos [64].
- **XGBoost:** se utiliza para implementar algoritmos de aumento de gradiente, especialmente efectivo para problemas de clasificación [65].
- **LightGBM:** una biblioteca de aprendizaje automático que utiliza algoritmos de aumento de gradiente optimizados para brindar velocidad y eficiencia [66].
- **shap:** ayuda a explicar las predicciones del modelo de aprendizaje automático utilizando valores de explicaciones aditivas de SHapley (SHAP) [67].

### 4.3 Herramientas y plataformas para desarrollo y análisis interactivo

El entorno de desarrollo principal utilizado para este proyecto fue Visual Studio Code [68], un IDE (entorno de desarrollo integrado) que proporciona una variedad de funciones para facilitar la escritura de código en Python. Visual Studio Code proporciona un entorno de desarrollo sólido y altamente personalizable para proyectos de ciencia de datos y aprendizaje automático, con herramientas como resaltado de sintaxis, autocompletar y depuración integrada.

Además, los cuadernos interactivos, como los cuadernos de Jupyter [69], se utilizan para análisis de datos exploratorios, experimentos de modelos y documentación interactiva.

Estos cuadernos permiten combinar código, visualizaciones y texto explicativo en un solo documento, facilitando la comunicación de resultados y los procesos de desarrollo colaborativo. La integración de Visual Studio Code con Notebooks a través de extensiones como Jupyter permite flujos de trabajo fluidos y eficientes que combinan el poder de un entorno de desarrollo con la interactividad y las capacidades narrativas de Notebooks.

Además, las extensiones de Jupyter eran importantes para la función `shap.initjs()`, que permite la visualización interactiva en los cuadernos de Jupyter. Esta inicialización habilita diagramas generados por SHAP. Los valores SHAP y los gráficos de dependencia se muestran correctamente y son interactivos. Esto proporciona una comprensión más profunda de cómo los diferentes atributos influyen en las predicciones del modelo, aumentando la interpretabilidad y transparencia de los resultados obtenidos.

# 5

## Implementación

En esta sección, se detalla la implementación a nivel de código de la detección del ataque DDoS utilizando técnicas avanzadas de aprendizaje automático. El proceso incluye la preparación de datos, la construcción de los modelos y la evaluación mediante métricas de rendimiento clave y técnicas de explicabilidad, se explicará con detalle las partes más relevantes del código.

### 5.1 Modelos y carga de datos

*Función para cargar datos:*

```
def cargar_datos():
    # Verificar si los archivos con los datos ya existen

    if os.path.exists('datos_X_DDoS.csv') and
os.path.exists('datos_Y_DDoS.csv'):
        X = pd.read_csv('datos_X_DDoS.csv')
        # y = pd.read_csv('datos_y.csv', header=None, squeeze=True) #
Usar squeeze=True para obtener una Serie
        y = pd.read_csv('datos_Y_DDoS.csv', header=None,
skiprows=1).iloc[:, 0] # Obtener la Serie de una sola columna
    else:
        # Recopilación de archivos CSV de un directorio y
concatenación en un DataFrame
        csv_files = []
        for dirname, _, filenames in
os.walk('dataset\MachineLearningCSV_reducido\MachineLearningCVE'):
            for filename in filenames:
                csv_file = os.path.join(dirname, filename)
                print(os.path.join(dirname, filename))
                csv_files.append(csv_file)
        df = pd.concat([pd.read_csv(file) for file in csv_files],
ignore_index=True)

        # Limpieza de datos: eliminación de espacios en nombres de
columnas y valores nulos o infinitos
```



```

df.columns = df.columns.str.strip()
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df.dropna(inplace=True)

# Preprocesamiento de la columna 'Label' para agrupar
categorías de ataques similares
df_experiment = df.copy(deep=True)
df_experiment.Label.replace("Web.*", "Web Attack", regex=True,
inplace=True)
df_experiment.Label.replace(r'.*Patator$', "Brute Force",
regex=True, inplace=True)
df_experiment.Label.replace(["DoS GoldenEye", "DoS Hulk", "DoS
Slowhttptest", "DoS slowloris"], "DDoS", inplace=True)

# División de los datos en conjuntos de entrenamiento y prueba
y = df_experiment.Label
X = df_experiment.drop(columns='Label')

# Guardar los datos en archivos CSV
X.to_csv('datos_X_DDOS.csv', index=False)
y.to_csv('datos_Y_DDOS.csv', index=False)

return X, y

```

La función `cargar_datos` realiza la carga y preprocesamiento de datos para la clasificación de ataques DDoS. A continuación, se describen brevemente sus pasos:

1. **Verificación y carga de archivos:**

- Si existen los archivos `datos_X_DDOS.csv` y `datos_Y_DDOS.csv`, se cargan directamente. Si no existen, se recopilan y concatenan archivos CSV de un directorio específico.

2. **Limpieza de datos:**

- Se eliminan espacios en los nombres de las columnas y se reemplazan valores infinitos con NaN, eliminando luego las filas con valores nulos.

3. **Preprocesamiento de etiquetas:**

- Las etiquetas de ataques se agrupan en categorías comunes: "Web Attack", "Brute Force" y "DDoS".

4. **División y guardado de datos:**

- Se separan las características (X) de las etiquetas (y) y se guardan en archivos CSV para futuras ejecuciones.

5. **Retorno de datos:**

- La función retorna los DataFrames X (características) y y (etiquetas).

Esta función asegura que los datos estén listos para su uso en modelos de clasificación, optimizando el proceso de carga y preprocesamiento.

#### *Entrenamiento de modelos:*

```
def test_svm_models(X_train, X_test, y_train, y_test, kernel,
degree=0):

    if kernel == 'poly':
        svm_model = SVC(kernel=kernel, degree=degree)
        svm_model.fit(X_train, y_train)
        joblib.dump(svm_model,
f'svm_model_{kernel}_degree_{degree}DDOS.pkl')
        score = svm_model.score(X_test, y_test)
        print(f"Score del SVM con kernel {kernel}: {score:.4f}: ")
    else:
        svm_model = SVC(kernel=kernel)
        svm_model.fit(X_train, y_train)
        joblib.dump(svm_model, f'svm_model_{kernel}DDOS.pkl')
        score = svm_model.score(X_test, y_test)
        print(f"Score del SVM con kernel {kernel}: {score:.4f}: ")

try:
    randomForest_modelDDOS =
joblib.load('randomForest_model_DDOS.pkl')
except FileNotFoundError:
    randomForest_modelDDOS = RandomForestClassifier()
    randomForest_modelDDOS.fit(X_train, y_train)
    joblib.dump(randomForest_modelDDOS, 'randomForest_model_DDOS.pkl')

try:
    lgb_model_DDOS = joblib.load('lgb_model_DDOS.pkl')
except FileNotFoundError:
    lgb_model = lgb.LGBMClassifier(verbosity=-1)
    lgb_model.fit(X_train, y_train)
    joblib.dump(lgb_model, 'lgb_model_DDOS.pkl')

try:
    label_encoder = LabelEncoder()
    y_train_encoded = label_encoder.fit_transform(y_train)
    y_test_encoded = label_encoder.transform(y_test)
    y_encoded = label_encoder.transform(y)
    y_test_encoded_series = pd.Series(y_test_encoded)
    xgBoost_model_DDOS = joblib.load('xgBoost_model_DDOS.pkl')
except FileNotFoundError:
    label_encoder = LabelEncoder()
    y_train_encoded = label_encoder.fit_transform(y_train)
    y_test_encoded = label_encoder.transform(y_test)
    xgBoost_model = xgb.XGBClassifier()
    xgBoost_model.fit(X_train, y_train_encoded)
    joblib.dump(xgBoost_model, 'xgBoost_model_DDOS.pkl')

try:
    logisticRegression_model_DDOS =
joblib.load('logisticRegression_model_DDOS.pkl')
except FileNotFoundError:
    logistic_model = LogisticRegression(penalty='l2', solver='saga',
max_iter=10000)
    logistic_model.fit(X_train, y_train)
    joblib.dump(logistic_model, 'logisticRegression_model_DDOS.pkl')
```

```

try:
    mlp_model_DDOS = joblib.load('mlp_model_DDOS.pkl')
except FileNotFoundError:
    mlp_model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=300,
solver='adam', random_state=42)
    mlp_model.fit(X_train,y_train)
    joblib.dump(mlp_model,'mlp_model_DDOS.pkl')

```

Este segmento de código entrena y evalúa varios modelos de clasificación utilizando los métodos de la biblioteca sklearn correspondiente a cada modelo. A continuación, se presenta una breve descripción de las funciones y procedimientos utilizados:

#### 1. Función `test_svm_models`:

- Entrena un modelo SVM con diferentes kernels (lineal, polinomial, RBF) utilizando `sklearn.svm.SVC`.
- Guarda el modelo entrenado en un archivo y calcula su puntuación en el conjunto de prueba.

#### 2. Entrenamiento y evaluación de otros modelos:

- **RandomForest**: utiliza `RandomForestClassifier` también de la librería `sklearn.ensemble` para entrenar el modelo. Intenta cargar un modelo preentrenado; si no existe, lo entrena y guarda.
- **LightGBM**: utiliza `lightgbm.LGBMClassifier` para entrenar el modelo. Procede de manera similar, entrenando y guardando el modelo si no se encuentra uno preexistente.
- **XGBoost**: utiliza `xgboost.XGBClassifier` para entrenar el modelo. Codifica las etiquetas y entrena el modelo, guardándolo después.
- **Regresión Logística**: utiliza `LogisticRegression` de la librería `sklearn.linear_model` para entrenar el modelo. Entrena y guarda un modelo de regresión logística.
- **MLP (Multilayer Perceptron)**: utiliza `MLPClassifier` de la librería `sklearn.neural_network` para entrenar el modelo. Entrena y guarda un modelo de red neuronal MLP.

Este enfoque permite entrenar múltiples modelos de manera eficiente, reutilizando modelos previamente entrenados y almacenados en archivos, lo que acelera el proceso de evaluación y selección del mejor modelo.

## 5.2 SHAP

*Funciones para valores SHAP, análisis de instancias y comparación de valores SHAP con valores reales:*

```
def shap_valores_globales(model,modelName, feature_names, clase = 0):
    shap.initjs()
    if modelName == 'svmLineal':
        try:
            shap_values = joblib.load('shap_values_svmLineal.pkl')
            shap.summary_plot(shap_values, X_test, feature_names =
feature_names)
        except FileNotFoundError:
            explainer_shap_svmLineal =
shap.LinearExplainer(model,X_train_df)

            joblib.dump(explainer_shap_svmLineal,'explainer_shap_svmLineal.pkl')
            shap_values =
            explainer_shap_svmLineal.shap_values(X_test_df)
            joblib.dump(shap_values,'shap_values_svmLineal.pkl')
            shap.summary_plot(shap_values, X_test, feature_names =
feature_names)

    elif modelName == 'randomForest':
        try:
            shap_values=joblib.load('shap_values_randomForest.pkl')
            shap_values_class0 = shap_values[:, :,clase]
            shap.summary_plot(shap_values_class0, X_test,
feature_names = feature_names)
        except FileNotFoundError:
            explainer_shap_randomForest =
shap.TreeExplainer(model,X_train_df)

            joblib.dump(explainer_shap_randomForest,'explainer_shap_randomForest.p
kl')

            shap_values =
            explainer_shap_randomForest.shap_values(X_test_df)
            joblib.dump(shap_values,'shap_values_randomForest.pkl')
            shap_values_class0 = shap_values[:, :,clase]
            shap.summary_plot(shap_values_class0, X_test,
feature_names = feature_names)

    elif modelName == 'lgb':
        try:
            shap_values = joblib.load('shap_values_lgb.pkl')
            shap.summary_plot(shap_values, X_test, feature_names =
feature_names)
        except FileNotFoundError:
            explainer_shap_lgb = shap.TreeExplainer(model,X_train_df)
            joblib.dump(explainer_shap_lgb,'explainer_shap_lgb.pkl')
            shap_values = explainer_shap_lgb.shap_values(X_test_df)
            joblib.dump(shap_values,'shap_values_lgb.pkl')
            shap.summary_plot(shap_values, X_test, feature_names =
feature_names)

def shap_explicacion_local(explainerModelName, X_test, sample_index,
feature_names):
    explainerModel =
joblib.load(f'explainer_shap_{explainerModelName}.pkl')
    shap_values = joblib.load(f'shap_values_{explainerModelName}.pkl')
```

```

shap.initjs()
if explainerModelName == 'randomForest':
    shap.force_plot(explainerModel.expected_value[0],
shap_values[sample_index,:,1], X_test[sample_index,:],
feature_names=feature_names,matplotlib=True,figsize=(100, 5))
    elif explainerModelName == 'svmLineal' or explainerModelName ==
'lgb' or explainerModelName == 'xgBoost' or explainerModelName ==
'logisticRegression' or explainerModelName == 'mlp' or
explainerModelName == 'poly2':
    expl = explainerModel.shap_values(X_test[sample_index])
    shap.force_plot(explainerModel.expected_value, expl,
X_test[sample_index], feature_names =
top_20_feature_names,matplotlib=True, figsize=(50, 2.5))
    else:
        print("No se ha introducido un modelo válido")

```

```

def shap_valores_vs_reales(explainerModelName, X_test, feature_names,
feature_index, clase=0):
    if explainerModelName == 'randomForest':
        # Choose an interaction index that is different from the
feature index
        shap_values =
joblib.load(f'shap_values_{explainerModelName}.pkl')
        shap.dependence_plot(feature_index, shap_values[:, :, clase],
X_test,
feature_names=feature_names,interaction_index=feature_names[feature_in
dex])
    elif explainerModelName == 'svmLineal' or explainerModelName
=='lgb' or explainerModelName == 'xgBoost' or explainerModelName ==
'logisticRegression' or explainerModelName == 'mlp' or
explainerModelName == 'poly2':
        # Choose an interaction index that is different from the
feature index
        shap_values =
joblib.load(f'shap_values_{explainerModelName}.pkl')
        shap.dependence_plot(feature_index, shap_values, X_test,
feature_names=feature_names,interaction_index=feature_names[feature_in
dex])
    else:
        print("No se ha introducido un modelo válido")

```

Este código utiliza la librería SHAP para interpretar modelos de machine learning entrenados. Se incluyen tres funciones principales: shap\_valores\_globales, shap\_explicacion\_local y shap\_valores\_vs\_reales.

A continuación, se describe brevemente cada una:

#### 1. Función shap\_valores\_globales:

- Genera e interpreta los valores SHAP globales para diferentes modelos (svmLineal, randomForest, lgb, también XGBoost y Logistic Regression que son idénticas a svmLineal y lgb).

- Si los valores SHAP preexistentes se encuentran guardados, los carga y genera un gráfico de resumen. Si no, calcula los valores SHAP, los guarda, y luego genera el gráfico de resumen.
- Los gráficos de resumen muestran la importancia de las características y cómo influyen en las predicciones del modelo.

## 2. Función **shap\_explicacion\_local**:

- Proporciona una interpretación local de una muestra específica del conjunto de prueba.
- Carga el modelo explicador SHAP correspondiente y los valores SHAP, y genera un gráfico de fuerza que muestra cómo cada característica afecta la predicción de la muestra seleccionada.

## 3. Función **shap\_valores\_vs\_reales**:

- Genera gráficos de dependencia para comparar los valores SHAP de una característica específica con sus valores reales.
- Carga los valores SHAP preexistentes para el modelo especificado y genera un gráfico de dependencia que muestra la relación entre una característica y su impacto en la predicción.
- Admite los mismos modelos mencionados anteriormente.

Este conjunto de funciones permite una interpretación detallada y visual de los modelos entrenados, ayudando a entender cómo cada característica contribuye a las predicciones, tanto a nivel global como local.

## 5.3 LIME

*Función análisis de instancias con LIME:*

```
def lime_analisis_instancias(modelName, X_train, X_test, sample_index,
feature_names):
    if modelName == 'svmLineal' or modelName == 'svmPoly2' or
modelName == 'svmPoly3' or modelName == 'svmRbf':
        calibrated = joblib.load('calibrated_{}'.format(modelName))
    elif modelName == 'randomForest':
        calibrated = randomForest_modelDDoS
    elif modelName == 'lgb':
        calibrated = lgb_model_DDOS
    elif modelName == 'xgBoost':
        calibrated = xgBoost_model_DDOS
    elif modelName == 'logisticRegression':
        calibrated = logisticRegression_model_DDOS
    elif modelName == 'mlp':
        calibrated = mlp_model_DDOS
    else:
        print("No se ha introducido un modelo válido")

    explainer_lime = lime.lime_tabular.LimeTabularExplainer(X_train,
feature_names=feature_names, class_names=['BENIGN', 'DDoS'],
discretize_continuous=True)
    exp = explainer_lime.explain_instance(X_test[sample_index],
calibrated.predict_proba)
    exp.show_in_notebook(show_table=True)
```

Esta función también está replicada, pero mostrando los pesos en gráficos de dispersión y de barras. A continuación, se detalla su funcionamiento:

- **Carga de modelos calibrados:**
  - Para los modelos SVM (svmLineal, svmPoly2, svmPoly3, svmRbf), se carga un modelo previamente calibrado. La calibración se realiza con el método CalibratedClassifierCV para mejorar la precisión de las probabilidades predichas (esto se hace en otra parte de la carga de datos y modelos).
  - Esto es esencial para el método predict\_proba en el explicador de LIME porque proporciona la probabilidad de que una instancia pertenezca a cada clase, permitiendo a LIME descomponer la predicción en contribuciones de características individuales.
- **Selección del modelo:**
  - Dependiendo del nombre del modelo (modelName), se selecciona el modelo adecuado para el análisis. Esto incluye modelos como randomForest, lgb, xgBoost, logisticRegression, y mlp.

- **Generación de explicaciones con LIME:**
  - Se crea un explicador LIME utilizando `LimeTabularExplainer`, que toma el conjunto de entrenamiento (`X_train`), los nombres de las características (`feature_names`), y los nombres de las clases (`class_names`).
  - Se genera una explicación para una instancia específica del conjunto de prueba (`X_test[sample_index]`) mediante la función `explain_instance`.
  - La explicación se muestra en un formato interactivo dentro del notebook con `exp.show_in_notebook(show_table=True)`.

Al conocer las probabilidades (de ahí la importancia de `predict_proba`), LIME puede resaltar mejor qué características influyen más en aumentar o disminuir la probabilidad de que una instancia pertenezca a una clase particular. Esto proporciona una visión más rica y detallada de la toma de decisiones del modelo.

En resumen, el uso de LIME proporcionar explicaciones detalladas y precisas sobre las decisiones del modelo, facilitando la comprensión de cómo y por qué se hicieron las predicciones.

## 5.4 Técnicas y evaluaciones adicionales

En esta sección, se proporciona una visión general de tres funciones clave utilizadas para evaluar el rendimiento de los modelos de manera más técnica. Debido a la extensión del código y a la familiaridad de los métodos empleados, no se mostrará el código. A continuación, se describen brevemente cada una de estas funciones y su propósito:

1. **Análisis de residuos:** esta función, `analisis_residuos`, tiene como objetivo evaluar los residuos generados por el modelo al comparar las probabilidades predichas con las etiquetas verdaderas. Primero, se codifican las etiquetas utilizando `LabelEncoder` para garantizar la compatibilidad con los cálculos. Luego, se calculan y visualizan los



residuos mediante un diagrama de dispersión, un histograma y un gráfico de densidad de kernel. Estos gráficos permiten observar la distribución de los residuos y detectar cualquier patrón sistemático en las predicciones del modelo.

2. **Análisis de sensibilidad:** la función `analisis_sensibilidad` examina cómo cambia la predicción del modelo al modificar los valores de una característica específica en el conjunto de pruebas. Para cada valor posible de la característica, se crea un nuevo conjunto de datos en el que esa característica tiene un valor fijo, mientras que las demás características permanecen constantes. Luego, se realiza una predicción y se calcula la media de las predicciones para cada valor de la característica.
3. **Análisis de influencia:** la función `analisis_influencia` evalúa cómo la exclusión de un subconjunto de instancias (las primeras 20,000 en este caso) afecta el rendimiento del modelo. Primero, se calculan las métricas de rendimiento del modelo con el conjunto de pruebas original. Luego, se excluye el subconjunto de instancias y se recalculan las métricas de rendimiento con el conjunto de datos reducido. Se comparan las métricas antes y después de la exclusión mediante gráficos de barras, lo que permite evaluar si la eliminación de instancias afecta significativamente el rendimiento del modelo.

#### 5.4.1 Permutación de características

*Función permutación de características:*

```
def permutacion_de_caracteristicas(model, X_test, y_test, feature):  
    precision_original = accuracy_score(y_test, model.predict(X_test))  
    print("Precisión original:", precision_original)  
  
    # Función para calcular la precisión después de permutar una  
    # característica específica  
    def precision_despues_permutacion(X, y, modelo,  
    caracteristica_permutada):  
        X_permutado = X.copy()  
        np.random.shuffle(X_permutado[:, caracteristica_permutada]) #  
        Permutar los valores de la característica  
        return accuracy_score(y, modelo.predict(X_permutado))
```

```

# Calcular la precisión después de permutar la característica
seleccionada
precision_permutacion = precision_despues_permutacion(X_test,
y_test, model, feature)
print("Precisión después de la permutación de la característica:",
precision_permutacion)

# Calcular el cambio en la precisión debido a la permutación de
características
cambio_precision = precision_permutacion - precision_original
print("Cambio en la precisión:", cambio_precision)

```

A continuación, se detalla su funcionamiento:

1. **Evaluación inicial:** la función comienza calculando la precisión del modelo en el conjunto de prueba original utilizando `accuracy_score`, que mide la proporción de predicciones correctas. Este valor se imprime como la "Precisión original".
2. **Permutación de característica:** a continuación, define una subfunción, *precision\_despues\_permutacion*, que calcula la precisión del modelo después de permutar los valores de una característica específica en el conjunto de prueba. La permutación se realiza al desordenar aleatoriamente los valores de la característica seleccionada, mientras que las demás características permanecen intactas. Esto simula un escenario en el que la característica seleccionada pierde su significado original.
3. **Cálculo de precisión permutada:** se calcula la precisión del modelo en el conjunto de prueba con la característica permutada, y este valor se imprime como "Precisión después de la permutación de la característica".
4. **Análisis del impacto:** finalmente, la función evalúa el impacto de la permutación en el rendimiento del modelo al calcular la diferencia entre la precisión original y la precisión después de la permutación. Esta diferencia, conocida como el "Cambio en la precisión", se imprime para determinar cuánto afecta la característica específica al desempeño del modelo.

# 6

## Análisis de resultados

Dentro de esta sección aplicaremos los conocimientos adquiridos a lo largo de este trabajo para analizar los resultados de los ataques DDoS y PortScan. Primero se proporcionará una evaluación de los modelos entrenados, y posteriormente se dará un análisis de su explicabilidad (consulte la Sección 3.5 para algoritmos y técnicas). Estos análisis tendrán en cuenta el impacto tanto a nivel general como en casos concretos.

Si bien la parte de explicabilidad, que es el objetivo de este trabajo, es la más importante, es primordial no ignorar la evaluación del modelo implementado en aspectos como precisión, recuerdo (sensibilidad), puntaje F1, resultados de validación cruzada, matriz de confusión.

La validación cruzada implica dividir los datos de entrenamiento en múltiples subconjuntos o "divisiones". El modelo se entrena en algunos de estos subconjuntos y se valida en los subconjuntos restantes. Este proceso se repite hasta que cada subconjunto se utiliza una vez como conjunto de validación. Esto reduce el sesgo asociado con una única partición de datos, permite una evaluación más sólida del rendimiento del modelo y proporciona una medida más confiable de la capacidad de un modelo para generalizar a nuevos datos. En este caso, la técnica de validación cruzada se realiza con k pliegues (valor predeterminado de 5 pliegues en este caso). Luego obtendrá los resultados para cada múltiplo, valor medio y desviación estándar.

Una matriz de confusión es una tabla que le permite visualizar el rendimiento de su modelo. Esto muestra no sólo el número de predicciones correctas e incorrectas, sino también cómo se distribuyen estos errores en las diferentes clases. La matriz de confusión consta de cuatro componentes principales: verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Los verdaderos positivos y negativos representan predicciones correctas para las clases positivas y negativas, respectivamente, mientras que los falsos positivos y negativos representan predicciones incorrectas para la misma clase. Una curva de característica operativa del receptor (ROC) es una representación gráfica que se utiliza para evaluar el rendimiento de un modelo de clasificación binaria. Mostramos la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) en diferentes umbrales de clasificación.

Cómo calcular la curva ROC:

- Generar predicciones probabilísticas: el modelo debe generar probabilidades de pertenecer a la clase positiva.
- Cambiar el umbral: varíe el umbral de decisión de 0 a 1 para determinar diferentes puntos en la curva ROC.
- Calcule TPR y FPR: calcule el TPR y FPR para cada umbral. Gráfico: para cada umbral, trace TPR en el eje Y y FPR en el eje X.

Área bajo la curva (AUC):

AUC (área bajo la curva): esta es una métrica que resume el rendimiento de la curva ROC en un solo valor.

AUC = 1: modelo completo.

AUC = 0,5: el modelo solo coincide.

AUC < 0,5: modelo peor que el azar (inversión de rango)

## 6.1 DDoS

### 6.1.1 Evaluación de modelos

En este apartado se presenta una tabla que resume las métricas de evaluación de varios modelos de clasificación aplicados al dataset CICIDS-2017, específicamente para la detección de ataques DDoS. La tabla incluye la precisión, puntuaciones de validación cruzada, media y desviación estándar de las puntuaciones de la validación cruzada se puede ver en la tabla 1, así como las matrices de confusión tabla 2. Estos resultados permiten comparar el rendimiento de diferentes algoritmos y entender su efectividad en la identificación de ataques DDoS.

Modelo	Precisión	Validación Cruzada	Media	Desviación Estándar
Random Forest	0.9998	[0.998,0.999,0.999, 0.999,0.997]	0.999	0.0009
Logistic Regression	0.9856	[0.951,0.977,0.985, 0.970,0.952]	0.967	0.0135
MLP	0.9994	[0.998,0.999,0.999, 0.999,0.998]	0.999	0.0003
LightGBM	0.9998	[0.999,0.999,0.999, 0.999,0.999]	0.999	0.0002
XGBoost	0.9998	[0.999,0.999,0.999, 0.999,0.999]	0.999	0.0002
SVM Linear	0.9985	[0.997,0.999,0.998, 0.998,0.997]	0.998	0.0006
SVM Poly Degree 2	0.9953	[0.993,0.997,0.997, 0.995,0.992]	0.995	0.0022
SVM Poly Degree 3	0.9957	[0.976,0.997,0.997, 0.996,0.993]	0.992	0.0081
SVM RBF	0.9982	[0.997,0.998,0.999, 0.998,0.996]	0.997	0.0009

Tabla 1: Resultados de Evaluación de Modelos para Detección de Ataques DDoS en el Dataset CICIDS-2017

Como se puede ver en la tabla, los modelos evaluados muestran un muy buen rendimiento en términos de precisión, con la mayoría alcanzando valores muy cercanos al 100%. Esto indica una capacidad notable para clasificar correctamente los datos de prueba. Las medias de validación cruzada son igualmente altas, reflejando la efectividad de los modelos en diferentes subconjuntos del conjunto de datos. Las bajas desviaciones estándar observadas en la mayoría de los modelos sugieren una alta consistencia y estabilidad en sus predicciones.

En general, los resultados demuestran que los modelos evaluados son altamente efectivos para la detección de ataques DDoS en el dataset CICIDS-2017. La alta precisión y baja variabilidad de las predicciones indican que estos modelos están bien ajustados para esta tarea específica. Sin embargo, se observa que algunos modelos, aunque muy precisos, presentan cierta variabilidad en sus resultados de validación cruzada, lo que sugiere que podrían beneficiarse de una optimización adicional.

Aunque el objetivo principal del proyecto no era optimizar los modelos, es importante reconocer que existen oportunidades para mejorar aún más su desempeño. Técnicas adicionales de ajuste de hiperparámetros y optimización podrían reducir la variabilidad y aumentar la precisión.

De hecho, el análisis de los casos en los que los modelos fallan puede proporcionar información valiosa sobre los patrones subyacentes en los datos, lo que es crucial para la explicabilidad y la mejora continua de los sistemas de detección de intrusiones.

Modelo	Matriz de Confusión			
	TN	FP	FN	TP
Random Forest	19537	1	8	25597
Logistic Regression	18932	606	41	25564
MLP	19532	6	20	25585
LightGBM	19537	1	8	25597
XGBoost	19537	1	8	25597
SVM Linear	19507	31	36	25569
SVM Poly Degree 2	19360	178	30	25575
SVM Poly Degree 3	19366	172	21	25584
SVM RBF	19485	53	28	25577

Tabla 2: Resumen de Resultados de la Matriz de Confusión para la Detección de Ataques DDoS

Los resultados que se presentan en la tabla de matrices de confusión en términos generales, la mayoría de los modelos muestran un rendimiento excepcionalmente alto en la clasificación de ataques DDoS. Modelos como Random Forest, LightGBM y XGBoost han demostrado ser particularmente efectivos, con muy pocos falsos positivos (FP) y falsos negativos (FN). Estos modelos muestran una capacidad notable para identificar correctamente tanto los ataques (True Positives, TP) como los eventos no maliciosos (True Negatives, TN).

El modelo de Regresión Logística, aunque efectivo, presenta un mayor número de falsos positivos en comparación con los modelos más avanzados como LightGBM y XGBoost. El MLP también ha mostrado un alto rendimiento, aunque con un número ligeramente mayor de falsos negativos en comparación con los modelos de árboles de decisión. Los modelos SVM presentan resultados variados, con el kernel lineal y los kernels polinomiales mostrando un mayor número de errores de clasificación en comparación con los modelos basados en árboles y redes neuronales.

En resumen, todos los modelos evaluados muestran un alto rendimiento en la detección de ataques DDoS.

Además, el código desarrollado incluye informes de clasificación para cada clase y cada modelo, en los que se presentan métricas como el Recall, F1-Score y precisión, todas aproximadas a dos decimales. La mayoría de estos valores se aproximan a 1.

Asimismo, al ejecutar el gráfico con las curvas ROC para todos los modelos, se puede observar y confirmar que todos los modelos presentan un rendimiento excelente, con puntuaciones AUC de 1.00 en la curva ROC, lo que indica que los modelos son casi perfectos, tal como se definió anteriormente.

### **6.1.2 Evaluación de residuos e influencia**

En este apartado, se presentan las gráficas de las funciones implementadas y explicadas en el apartado 3.5, así como las conclusiones derivadas de estas. Se utiliza un modelo específico para cada gráfica, aunque estas funciones son aplicables a cualquier modelo implementado. Un ejemplo será suficiente para comprender el comportamiento general, evitando así una sobrecarga de gráficas y explicaciones para cada modelo, función y tipo de ataque.

Comenzaremos viendo un ejemplo de la interpretación de los residuos (las diferencias entre los valores predichos y los valores reales) esta es fundamental para entender la precisión y el comportamiento del modelo. La gráfica que se muestra a continuación (figura 4) representa la densidad del kernel de los residuos para el modelo de regresión logística.

Este tipo de análisis es aplicable a cualquier modelo, simplemente pasando el nombre del modelo como parámetro a la función correspondiente.



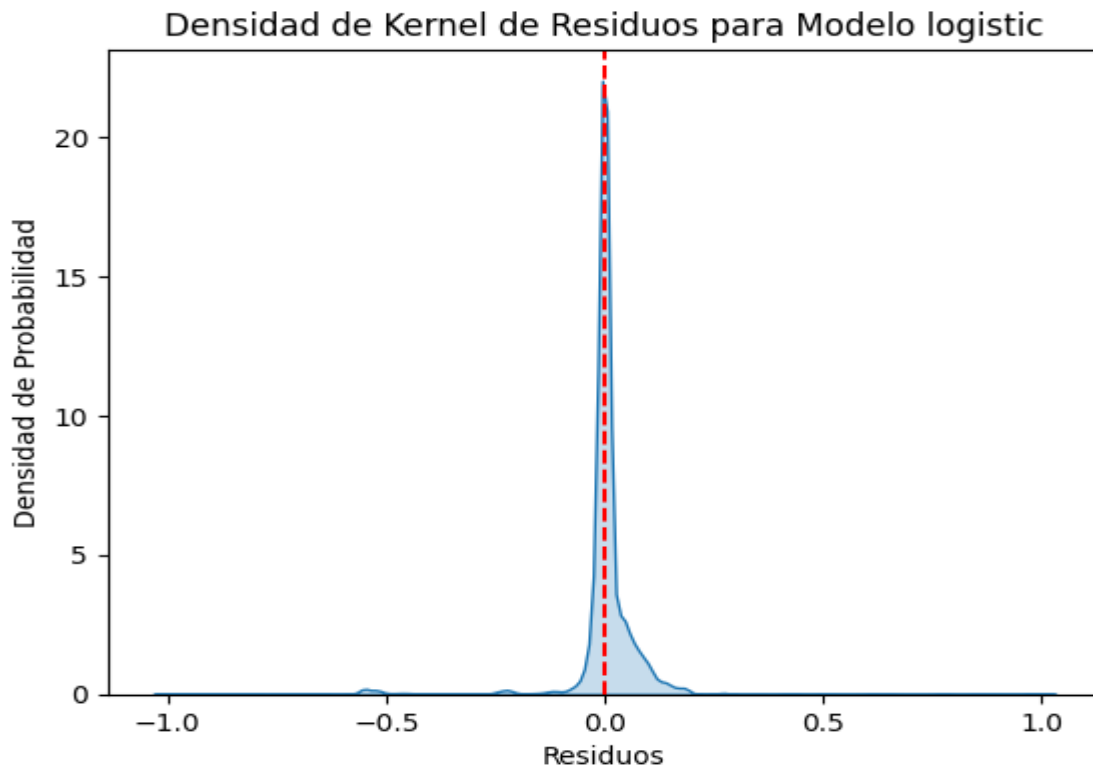


Figura 4: Gráfica de análisis de residuos para el modelo Logistic Regression para el ataque DDoS

La gráfica muestra que la mayoría de los residuos están muy cerca de cero, lo que indica que el modelo de regresión logística tiene un buen ajuste a los datos. La línea roja punteada en el cero resalta esta concentración. La distribución es altamente concentrada alrededor del cero con una curva muy pronunciada, lo que sugiere que los errores del modelo son pequeños y están bien distribuidos alrededor de la media.

La distribución de los residuos es aproximadamente simétrica respecto al cero. Esto indica que no hay un sesgo sistemático en las predicciones del modelo: no hay una tendencia clara a sobreestimar o subestimar los valores reales. Hay algunos residuos en los extremos de la distribución, pero son relativamente pocos. Estos pueden corresponder a outliers o casos donde el modelo no se ajusta tan bien, lo cual es esperable en cualquier modelo predictivo.

La simetría de la distribución de residuos alrededor del cero sugiere que no hay un sesgo sistemático en las predicciones del modelo.

Se puede sacar de conclusión al ver la simetría y la concentración de residuos alrededor del cero que el modelo no tiene un sesgo sistemático y que la mayoría de las predicciones son cercanas a los valores reales.

A continuación, se evalúa la influencia de subconjuntos específicos de datos en la gráfica (figura 5) representa una comparación de las métricas de rendimiento del modelo antes y después de excluir una parte significativa del conjunto de prueba. En este ejemplo, se ha utilizado un modelo SVM con kernel polinomial de grado 2 para ilustrar este análisis y el rango excluido de instancias ha sido **18000-35500** se puede aplicar la función a cualquier rango (que esté dentro del conjunto de test) y modelo.

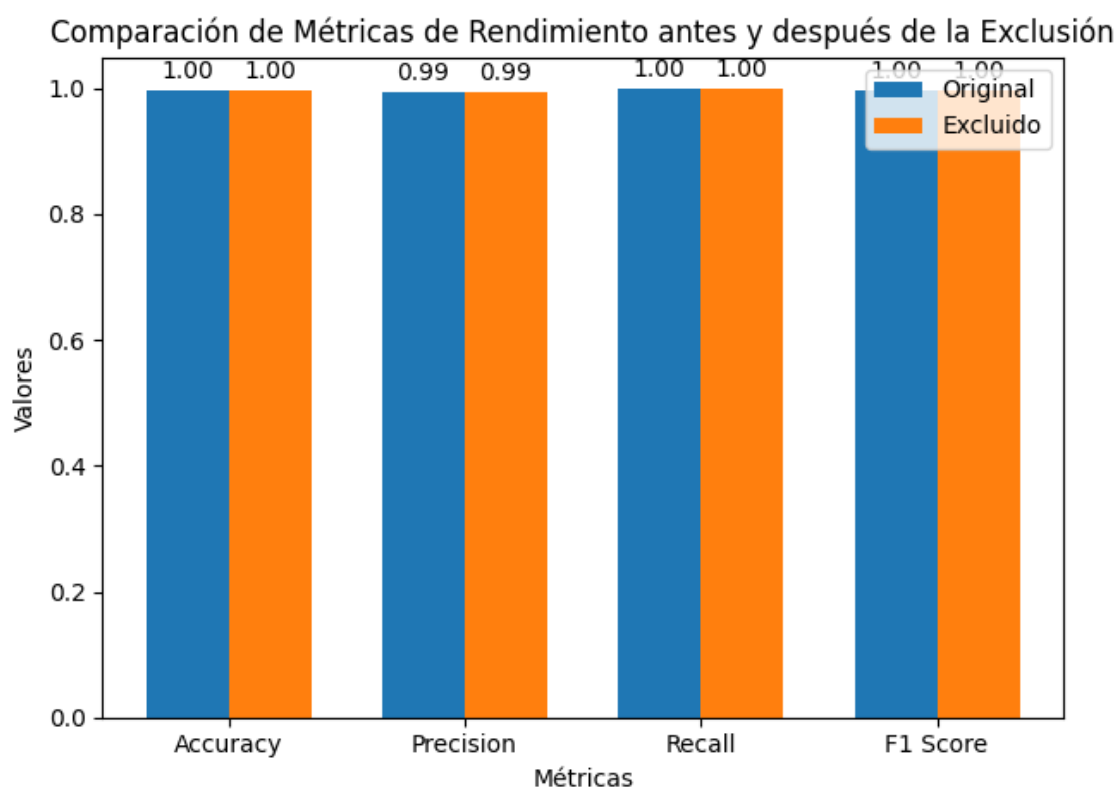


Figura 5: Gráfica de análisis de influencia para el modelo SVM Poly 2 para el ataque DDoS

La exclusión de las instancias no ha afectado ni a la exactitud, ni a la precisión, ni al Recall ni al F1 Score del modelo, lo que indica que el modelo es capaz de mantener un gran rendimiento incluso con un subconjunto reducido de datos

(se ha probado para varios subconjuntos y de datos y modelos y el comportamiento general es el mismo).

La constancia en las métricas de rendimiento sugiere que el modelo SVM con kernel polinomial de grado 2 es robusto, fiable y consistente. La exclusión de una gran porción de los datos de prueba no afecta significativamente el rendimiento del modelo, esto indica una buena capacidad de generalización también.

Este análisis permite a los usuarios entender cómo se comporta el modelo cuando se excluye un subconjunto de datos y ayuda a explicar por qué el modelo puede ser confiable en diferentes escenarios.

### 6.1.3 SHAP

Para el análisis SHAP, se empezará con un análisis detallado de los valores SHAP de cada característica. Los valores SHAP permiten una explicación consistente y local de las predicciones del modelo, lo cual es crucial para entender cómo cada característica influye en el resultado del modelo.

Estos análisis se realizan mediante una gráfica de valores SHAP. En ella, cada punto representa un valor SHAP para una instancia específica del dataset, y los puntos están coloreados de acuerdo con los valores de las características (con el color rojo indicando valores altos y el color azul indicando valores bajos). A su vez están distribuidos en el eje horizontal según su impacto en el modelo positivo o negativo.

**Análisis valores SHAP para XGBoost.** Inicialmente se realizará el análisis de los valores SHAP correspondientes al modelo XGBoost (figura 6). Esto es así ya que presenta características y valores que parecen están bien distribuidos alrededor del eje de impacto, lo cual es esencial para comprender el análisis más detallado de la interpretabilidad del modelo.

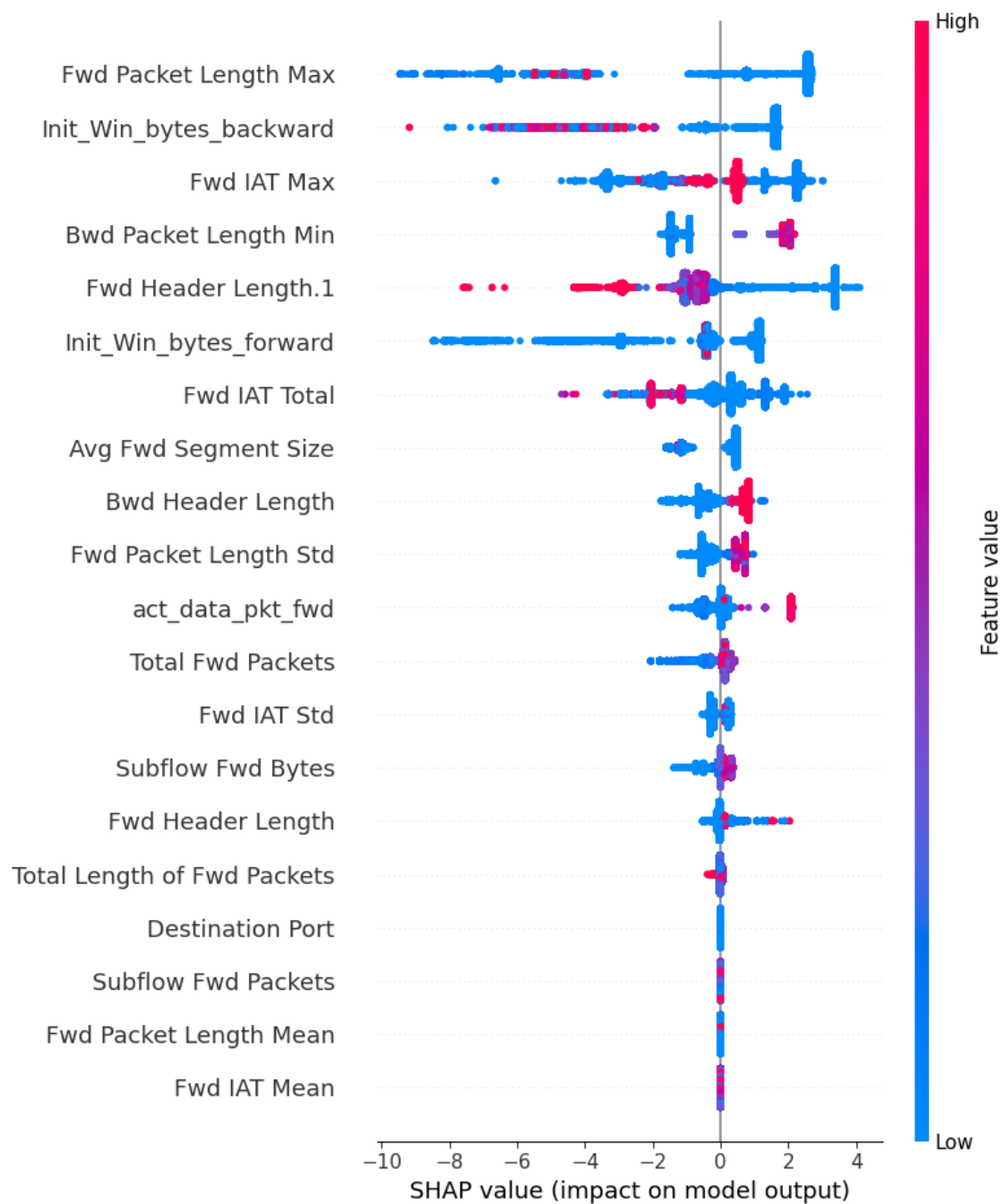


Figura 6: Valores SHAP para el modelo XGBoost para el ataque DDoS

Los valores SHAP en el eje horizontal indican el impacto de cada característica en la predicción del modelo. Un valor SHAP positivo indica que la característica aumenta la probabilidad de la clase positiva, mientras que un valor negativo indica que disminuye dicha probabilidad.

En esta gráfica, los valores SHAP varían entre aproximadamente -8 y 4. Esto sugiere que las características tienen un impacto relativamente significativo en las predicciones del modelo.

Respecto a los valores en el eje vertical, éstos representan las características, siendo las características principales las siguientes:

- `Init_Win_bytes_backward`: es la segunda característica más influyente. Valores altos (en rojo) generalmente disminuyen la predicción de la clase positiva este caso DDoS, mientras que valores bajos (en azul) tienden a aumentarla.
- `Bwd Packet Length Min`: al contrario que la característica anterior, con valores altos, aumenta la predicción, y con valores bajos tiende a disminuirla.
- `Fwd IAT Max`: vemos como cuando tiene valores altos no contribuye demasiado ni a aumentar ni a disminuir de hecho algunas veces contribuye a aumentar y otras a disminuir, sin embargo, con los valores bajos, sí que tiene una contribución alta tanto a aumentar como a disminuir.
- `Fwd Packet Length Max`: es la característica más influyente cuando tiene valores altos tienden a disminuir la predicción de la clase positiva.
- `Fwd Header Length.1`: con valores altos disminuye la predicción y con valores bajos aumenta.

Analizando la gráfica, podemos ver la influencia de características particulares. Por ejemplo, las características como `Init_Win_bytes_backward` (Tamaño inicial de la ventana de bytes en la dirección inversa.) y `Fwd Packet Length Max` (Longitud máxima de los paquetes enviados hacia adelante en una conexión.) son las más influyentes en las predicciones del modelo XGBoost. Esto sugiere que estos atributos son cruciales para la toma de decisiones del modelo.

Adicionalmente, podemos observar dos aspectos relacionados con los valores SHAP:

- Valores SHAP negativos y positivos: la presencia de valores SHAP tanto negativos como positivos muestra cómo diferentes valores de una característica pueden influir de manera diversa en la predicción. Esta dualidad es esencial para entender la dinámica interna del modelo.
- Varianza en el impacto: la varianza de los valores SHAP para características específicas, como se muestra en la dispersión de los puntos, indica la variabilidad en cómo estas características afectan las predicciones en diferentes instancias del dataset.

### **Análisis valores SHAP para otros modelos.**

Finalmente, al comparar los valores SHAP de diferentes modelos para este conjunto de datos (figura 7 para el modelo Random Forest y figura 8 para el modelo LightGBM), se puede observar cómo las características que influyen en las predicciones varían entre los modelos. Estas variaciones se deben a las diferencias en sus mecanismos de aprendizaje y estructuras internas.

Además, se han implementado los valores SHAP para los modelos SVM Lineal y Logistic Regression. Sin embargo, no se han calculado para otros modelos debido a problemas de rendimiento que se discutirán en el apartado de limitaciones. Por lo tanto, no disponemos de valores SHAP ni análisis de instancias específicas para esos modelos. En su lugar, utilizaremos el análisis LIME, que no presentó problemas de rendimiento.

Las figuras 7 y 8 de los modelos Random Forest y LightGBM, tienen el objetivo similar al del modelo XGBoost: identificar las características con mayor impacto en el modelo, su rango, y cómo afectan a la predicción cuando la característica tiene un valor alto o bajo. Es importante ver como las características más influyentes para cada modelo varían. En el caso de Random Forest, las características más influyentes son `Init_Win_bytes_backward` y `Avg Fwd Segment Size`. Para el modelo LightBM, las más influyentes son las mismas, aunque en distinto orden.

Estas características tienden a ser influyentes en la mayoría de los modelos de manera general en los modelos basados en árboles. Además, otras

características como "Fwd IAT Max" y "Init\_Win\_bytes\_backward" suelen figurar entre las más importantes, aunque su posición en el ranking de influencia puede variar entre los modelos.

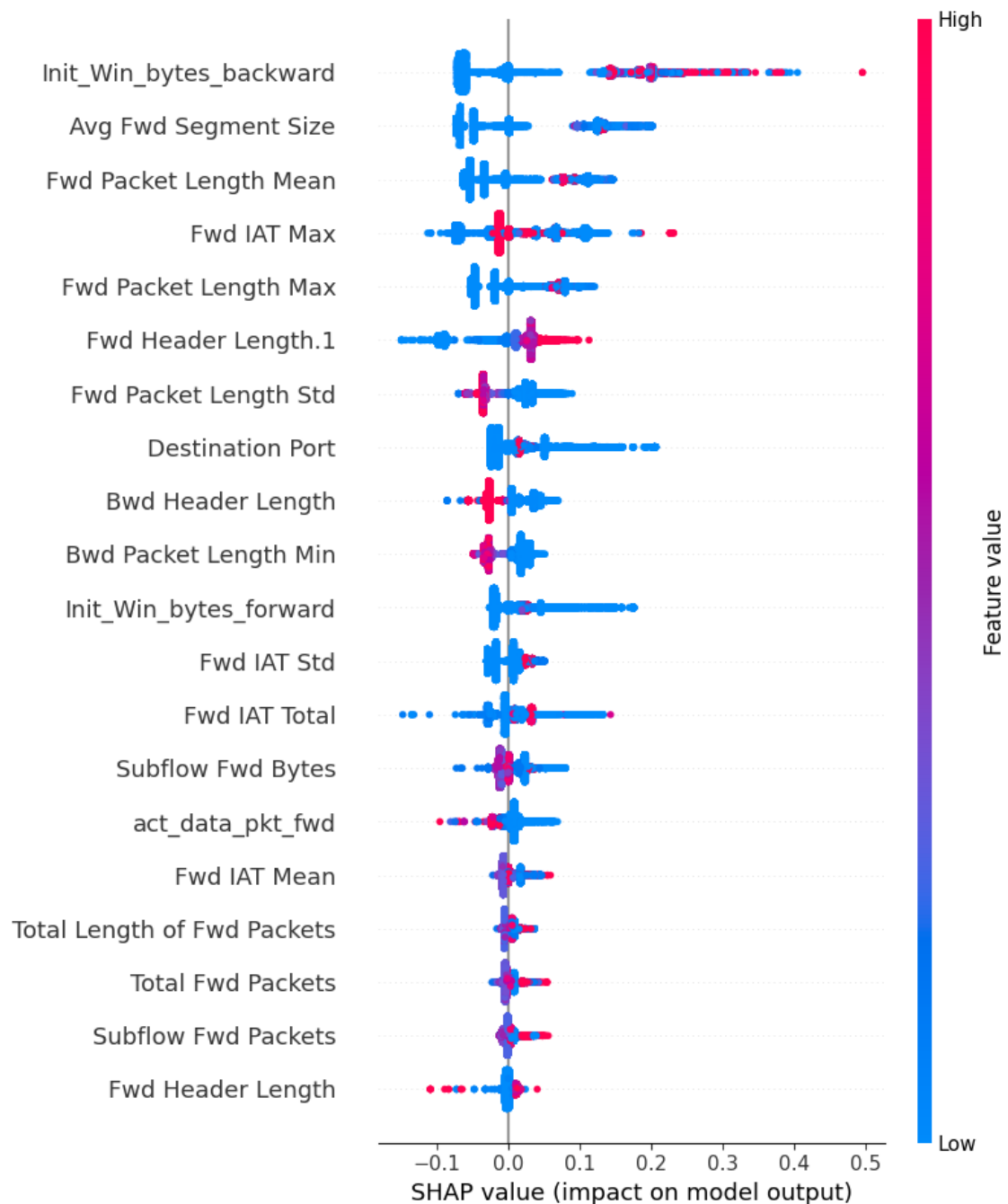


Figura 7: Valores SHAP para el modelo Random Forest (Clase 0- Benign) para el ataque DDoS

El caso de los valores SHAP para el modelo Random Forest es algo más complejo que para el resto, a pesar de ser un modelo de clasificación binaria, Random Forest trata la predicción de probabilidad de cada clase (clase positiva

y clase negativa) por separado por lo que los valores SHAP reflejan cómo cada característica afecta la predicción de la probabilidad para cada clase en específico.

Cuando se obtiene los valores SHAP para el RandomForest se obtiene una matriz tridimensional donde el primer índice corresponde a las instancias del conjunto de pruebas, el segundo índice corresponde a las características, y el tercer índice corresponde a las clases.

Aunque lo hace más complejo, ya que tenemos valores SHAP para cada clase, y esto implica después en las explicaciones tener que seleccionar la clase relevante para la visualización e interpretación. Normalmente, se selecciona la clase positiva para la interpretación, pero dependiendo del contexto, podría ser útil visualizar ambas. Este comportamiento asegura que incluso en problemas binarios, las explicaciones SHAP proporcionen una visión clara de las contribuciones de cada característica a la predicción del modelo, manteniendo la consistencia con su implementación general para problemas multiclase.

También para la relación entre valores SHAP y valores reales, habría que tener en cuenta la clase lo que implica que la interpretación de los valores SHAP y su relación con el valor real es siempre en el contexto de esa clase. Como se puede apreciar en la figura 7 cuyos valores SHAP están en el contexto de la clase 0 (Benign). Esto puede complicar la visualización cuando se desea una interpretación global, pero proporciona mayor precisión y detalle por clase.

Esta especificidad permite obtener interpretaciones detalladas y precisas para cada clase, aunque a costa de una mayor complejidad en el cálculo y visualización de los valores SHAP.

En los otros modelos de árboles utilizados como XGBoost, LightBM (figura 8) usan técnicas de boosting y suelen tener un manejo más directo de las salidas multiclase, donde los valores SHAP pueden ser calculados directamente para todas las clases en una sola llamada.



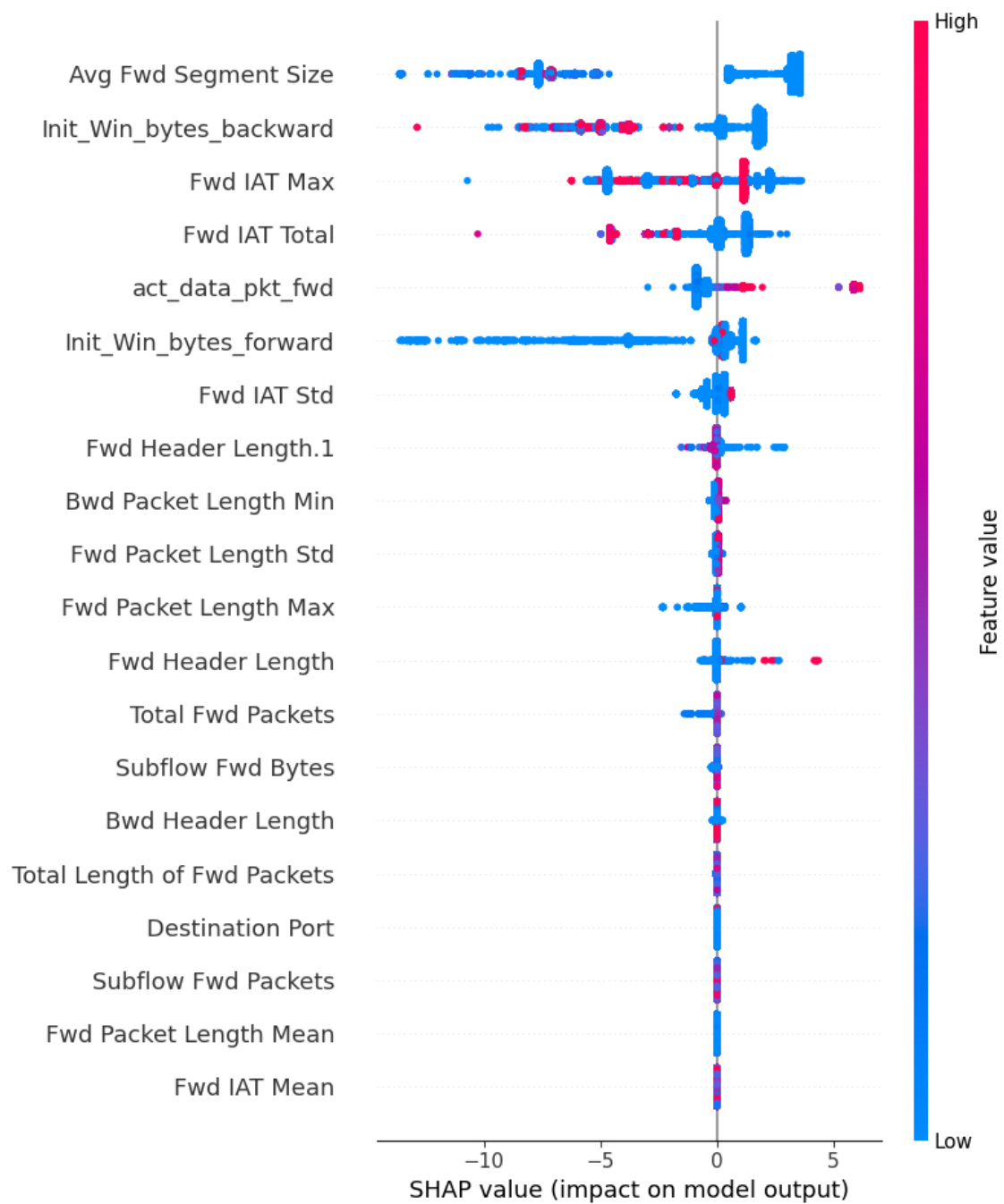


Figura 8: Valores SHAP para el modelo LightBM para el ataque DDoS

## Valores SHAP para analizar instancias específicas.

Después de analizar los valores SHAP con una visión global, procederemos a explicar el análisis de instancias específicas.

Los valores SHAP interpretan el impacto de un valor específico de una característica en comparación con la predicción que se haría si esa característica tomara un valor base. La predicción base es la estimación que el modelo haría si todas las características tomaran sus valores promedio o de referencia, sin considerar las características específicas de la instancia analizada. Este valor sirve como un punto de partida neutral.

Para profundizar en el análisis, examinaremos cómo los valores SHAP pueden ser utilizados para desglosar y entender predicciones específicas. Este enfoque nos permite identificar y cuantificar el impacto de cada característica en la predicción de una instancia particular, proporcionando una comprensión detallada del comportamiento del modelo.

A continuación, nos centraremos en el modelo XGBoost, del cual ya hemos examinado sus valores SHAP. Se presentarán muestras Benign en las figuras 9, 10 y 11, así como una muestra de tipo DDoS en las figuras 12, 13, y 14 divididas en varias imágenes para facilitar una observación detallada de cada una de ellas.

### Caso instancia Benign

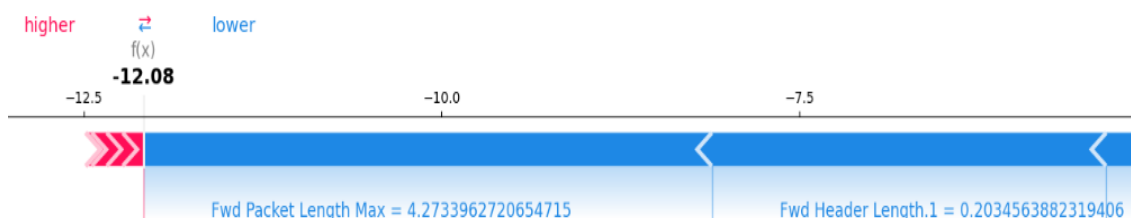


Figura 9: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque DDoS- Parte1

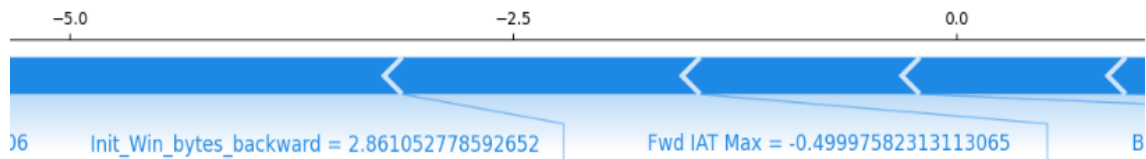


Figura 10: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque DDoS- Parte2



Figura 11: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque DDoS- Parte3

En este caso, a partir de las 3 figuras se puede ver como el valor base de la predicción es de aproximadamente 3 (figura 11), este es el punto de partida, y posteriormente se puede apreciar como las características más representativas para esta instancia provocan la desviación de la predicción respecto del valor base.

Al sumar todos los valores SHAP para las características, se obtiene la diferencia total entre la predicción del modelo y la predicción base. Cada valor SHAP de cada característica muestra cuanto contribuyó a disminuir la predicción de la clase positiva DDoS desde el 3 (valor base) a -12.08. Este último valor se obtiene sumando el valor base y la suma de los valores SHAP de todas las características para esa instancia.

La magnitud de cada flecha indica la fuerza de la contribución de esa característica a la predicción.

1. Fwd Packet Length Max (4.273396270654715 valor normalizado de la característica)

Esta característica tiene la mayor contribución negativa, reduciendo el valor de la predicción.

Valor SHAP: aproximadamente -4 (Figura 9)

2. Fwd Header Length.1 (0.2034563882319406 valor normalizado de la característica para esta instancia)

Esta característica también tiene una contribución negativa significativa.

Valor SHAP: aproximadamente -3 (Figura 9)

3. Init\_Win\_bytes\_backward (2.861052778592652 valor normalizado de la característica para esta instancia)

Contribuye de manera negativa, aunque en menor medida que las dos anteriores.

Valor SHAP: aproximadamente -2 (Figura 10)

4. Fwd IAT Max (-0.49997582313113065 valor normalizado de la característica para esta instancia)

Esta característica también tiene una contribución negativa.

Valor SHAP: aproximadamente -1.5 (Figura 10)

Si retrocedemos un poco hasta la figura 6, y recordamos cuales eran las características más influyentes según XGBoost, observamos que, las 3 principales eran precisamente las que se muestran en los gráficos: Init\_Win\_bytes\_backward, Fwd IAT Max y Fwd Packet Length Max. Además, en la gráfica de valores SHAP también podíamos ver como afectaban con un impacto negativo al modelo con ciertos valores. Estas características son críticas para la predicción del modelo en esta instancia específica.

La consistencia de estas observaciones específicas está alineada con las tendencias generales observadas en el gráfico global de valores SHAP. Esto sugiere que el modelo es coherente en la forma en que utiliza estas características para realizar predicciones.

En cuanto a las contribuciones, la mayoría de estas características principales tienen un impacto negativo en la predicción, lo que resulta en una disminución del valor final predicho.

**Caso instancia DDoS**

En las siguientes figuras (12, 13, 14 y 15), se analizará el caso contrario para una instancia positiva (DDoS), sin entrar en tanto detalle, ya que el concepto subyacente es el mismo.

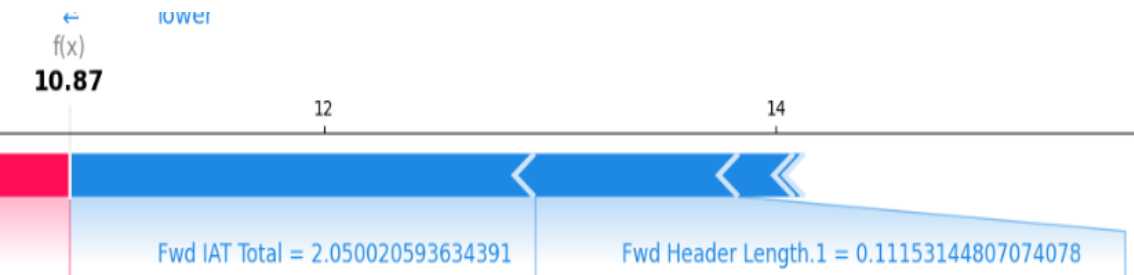


Figura 12: Análisis SHAP para instancia DDoS para modelo XGBoost en el ataque DDoS- Parte1.ra modelo XGBoost en el ataque DDoS- Parte1

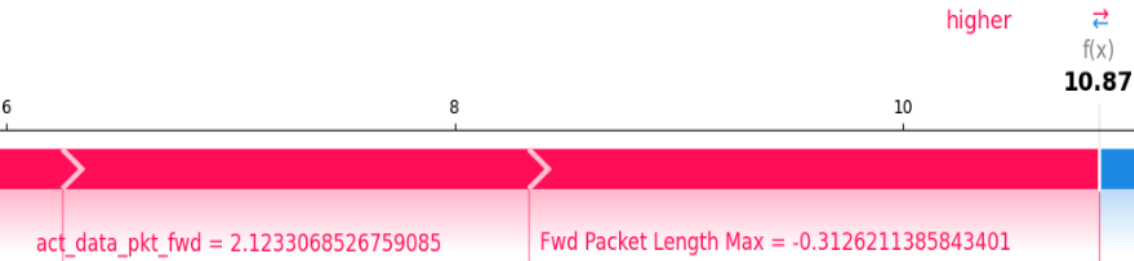


Figura 13: Análisis SHAP para instancia DDoS para modelo XGBoost en el ataque DDoS- Parte2

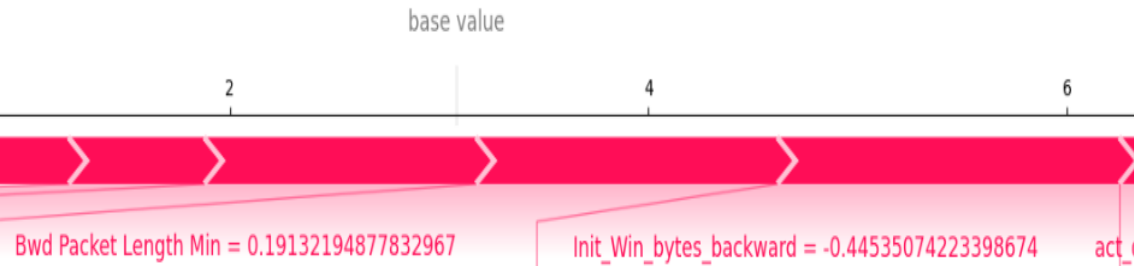


Figura 14: Análisis SHAP para instancia DDoS para modelo XGBoost en el ataque DDoS- Parte3

Como se puede observar en la última imagen, lógicamente, el valor base permanece inalterado en aproximadamente 3. Sin embargo, el efecto de las características en esta instancia ha ocasionado que cada valor SHAP de cada una de estas contribuya a incrementar la predicción de la clase positiva DDoS desde el 3 (valor base) a 10.87 (figura 12 valor predicho real).

Es evidente que varias características están aumentando la predicción, algunas en mayor medida que otras. Sin embargo, también son relevantes las características Fwd IAT Total y Fwd Header Length.1, que disminuyen la predicción con valores SHAP aproximados de -1 y -1.5, respectivamente. Lo que nos podría llevar a la conclusión errónea de que no se trata de una instancia maliciosa si se consideraran exclusivamente los valores de estas características.

Si nos fijamos en el valor SHAP de la característica Fwd Packet Length Max, vemos que es la que en mayor medida aumenta la predicción a la clase positiva. Asimismo, también era la que disminuía en mayor medida la predicción en el caso Benign, ¿Realmente tienen coherencia estos resultados?

Pues si nos referimos nuevamente a la figura 6 de los valores SHAP esta coherencia se hace evidente. En la instancia actual el valor de Fwd Packet Length Max es considerablemente menor que en la instancia previa. Esta característica puede impactar en el modelo tanto de manera positiva como negativa, debido a su amplio rango de valores SHAP en la figura 6. Cuando Fwd Packet Length Max tiene un valor bajo, como en este caso, puede influir positivamente en la predicción, alcanzando un valor SHAP de casi 2.5, lo cual concuerda con la observación actual.

Por lo contrario, en la instancia anterior Fwd Packet Length Max tenía un valor significativamente más alto, correspondiente a los puntos rojos en la figura 6, que se sitúan en el rango de valores SHAP de entre -3 y -6. En ese caso anterior, esta característica contribuía con un valor SHAP de -4. Estos resultados son coherentes y consistentes con las observaciones previas,

mostrando cómo las variaciones en las características afectan las predicciones del modelo de manera predecible según sus valores SHAP correspondientes.

En conclusión, los valores SHAP para cada característica representan la contribución específica de esa característica a la desviación de la predicción respecto al valor base. Sumando todos los valores SHAP para las características, se obtiene la diferencia total entre la predicción del modelo y la predicción base (al restar la longitud de las barras azules de las barras rojas, se obtiene la diferencia entre el valor base y el resultado de la predicción). Esta metodología permite descomponer una predicción y mostrar el impacto de cada característica.

Aunque la técnica SHAP presenta cierta complejidad, especialmente para asegurar que el valor base más la suma de los efectos individuales se iguale a la predicción, esta complejidad no es crítica para la utilización práctica de la técnica. Siendo especialmente útiles en varios contextos, como en un modelo que detecta anomalías en un dataset de DDoS, donde es crucial entender qué características específicas están contribuyendo a la detección de una posible amenaza. También son valiosos en sistemas de ciberseguridad que requieren explicaciones claras sobre los factores que impulsan la clasificación de tráfico como legítimo o malicioso, permitiendo así tomar medidas correctivas específicas.

Por ejemplo, podríamos preguntarnos en qué medida una predicción fue influenciada por el hecho de que el número total de paquetes enviados superó un umbral específico. Reformulando esta pregunta de manera más clara y cuantificable: ¿en qué medida fue una predicción influenciada por el hecho de que el número total de paquetes enviados fue 1000, en lugar de algún valor base de paquetes?

Dado que cada registro del dataset posee múltiples características, este proceso podría repetirse para todas las demás características. Los valores SHAP facilitan esta descomposición de manera coherente, lo que permite

visualizar el impacto individual de cada característica en un gráfico, como se observa en las figuras 9, 10 y 11.

### Valores SHAP vs Valores reales

Finalmente, en la sección de SHAP, considerando la característica Fwd Packet Length Max en el contexto del modelo XGBoost, se presenta una gráfica específica (figura 15). Esta representación ilustra la relación entre los valores reales de esta característica y sus respectivos valores SHAP. Es importante recordar que los valores reales fueron normalizados desde el inicio del entrenamiento de los modelos, medida que mejora la convergencia de los algoritmos y la precisión de las predicciones, como se explicó en el apartado 3.3.3.

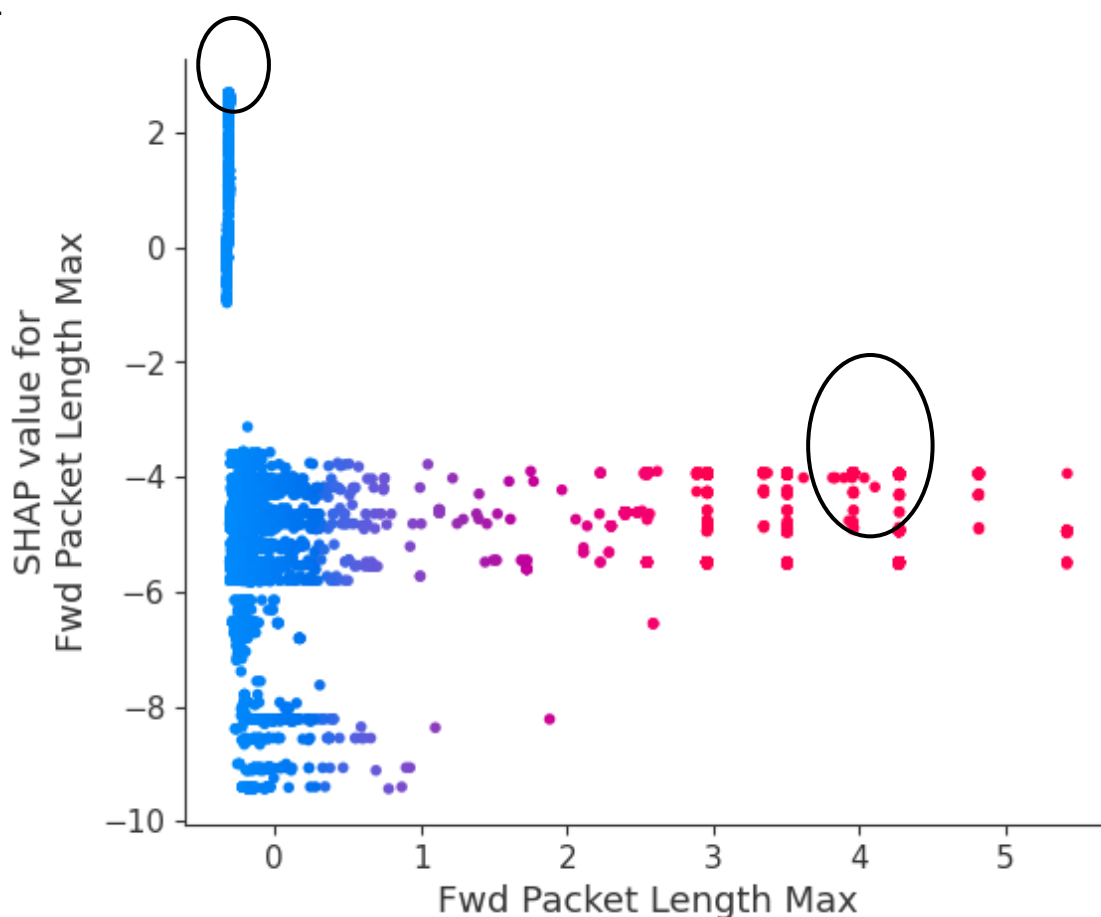


Figura 15: Relación de valores SHAP con valores reales (normalizados) para característica Fwd Packet Length Max con el modelo XGBoost para el ataque DDoS



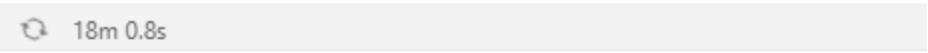
Se observa cómo varían los valores SHAP a medida que aumenta el valor real de la característica. Sin embargo, con los valores bajos de la característica, las conclusiones son menos claras. En contraste, los valores altos muestran consistentemente una contribución SHAP en un rango general entre -4 y -6. Al comparar con casos específicos que hemos analizado previamente, como la instancia Benign en la figura 9 donde la característica tiene un valor de 4.2733 y un valor SHAP cercano a -4, podemos corroborar esta relación utilizando la gráfica correspondiente (marcada). Similarmente, para la segunda instancia DDoS en la figura 13, donde la característica tiene un valor de -0.3126 y un valor SHAP aproximado de 2.5, también podemos reafirmar esta relación utilizando la gráfica 16 (marcada).

#### **6.1.3.1 Fracasos relacionados con el rendimiento**

Como ya se ha comentado antes, no todo han sido éxitos en el contexto de este análisis. A pesar de los buenos resultados anteriores, se han presentado algunos problemas de rendimiento para ciertos modelos con la librería SHAP. En específico con los explicadores DeepExplainer, utilizado para redes neuronales, perceptrón multicapa en nuestro caso, y también con KernelExplainer para las explicaciones de aquellos modelos que no tienen un explicador específico. Afectando así a los modelos SVM Poly2, SVM Poly3 y SVM Rbf, ya que en estos no se pueden usar los explicadores lineales ni de árboles, con los cuáles sí hemos tenido éxito en el resto de los modelos, TreeExplainer para XGBoost, Random Forest y LightBM y LinealExplainer para SVM Lineal y Logistic Regression.

Se ha intentado optimizar estos explicadores reduciendo el tamaño de datos de fondo que utiliza `shap.sample` a 100 muestras, también se ha reducido el tamaño del conjunto de entrenamiento en muchísimo porcentaje, pero aun así sigue siendo muy lento. Además, con tan pocos datos no aporta resultados consistentes en las explicaciones debido a los pocos valores de SHAP.

En la siguiente imagen (figura 16) se puede ver una prueba de cuando tarda el cálculo de valores SHAP para todas las características de cada instancia con el explicador KernelExplainer, en este caso para el modelo SVM Poly2, pero con cualquiera de los otros modelos mencionados es igual de ineficiente.



0% | 18m 0.8s

| 10/45143 [17:04<1328:51:05, 105.99s/it]

Figura 16: Tiempo de ejecución para cálculo de valores SHAP para el modelo SVM Poly 2 en el ataque DDoS

Se puede deducir que es inviable esa ejecución al menos con los recursos actuales, ya que para cada valor SHAP tarda unos 105 segundos, y al ser 45143 iteraciones, la ejecución se iría a 1316 días para poder terminar.

#### 6.1.4 LIME

El análisis de LIME no proporciona unos resultados tan extensos o ilustrativos como el análisis de SHAP, aunque presenta ciertas ventajas que hacen importante su aplicación. Una de ellas es la posibilidad de aplicarla al modelo de perceptrón multicapa, tal y como se observa en las figuras 17 y 18. En ellas, se muestran cómo afecta cada característica a la predicción según el algoritmo LIME, que genera una explicación local para cada predicción.

Los valores numéricos observados en cada figura son los coeficientes de este modelo lineal ajustado, que indican la contribución de cada característica a la predicción. Un valor positivo significa que esa característica contribuye a la predicción de 'DDoS', mientras que un valor negativo indica una influencia hacia la categoría 'BENIGN'. La magnitud del valor indica la importancia de la característica: cuanto mayor es el valor absoluto, más importante es la característica para la decisión de predicción del modelo.

Es importante recordar que estas explicaciones son locales a una decisión individual, y pueden no representar el comportamiento global del modelo. Sin embargo, proporcionan una valiosa intuición sobre cómo el modelo está tomando decisiones.

Ya en más detalle, la figura 17 proporciona los siguientes resultados:

- Probabilidades de predicción: la imagen indica que el modelo predice la categoría 'Benign' con una certeza del 100%, ya que muestra 'BENIGN' en 1.00 y 'DDoS' en 0.00.
- Contribución de características: las barras azules y naranjas representan el peso o la contribución de cada característica a la decisión de predicción del modelo. Por ejemplo, 'Fwd Header Length.1' y 'Fwd Packet Length Max' son algunas de las características que más influyen listadas.
- Interpretación de resultados: las características con barras positivas (naranjas) son aquellas que contribuyen a la predicción de 'DDoS', mientras que las negativas (azules) indican una influencia hacia la categoría 'Benign'.
- Explicabilidad del modelo: este tipo de visualización es crucial para entender el comportamiento del modelo en tareas de clasificación, mostrando qué características son más influyentes en la decisión del modelo.

### Caso instancia Benign

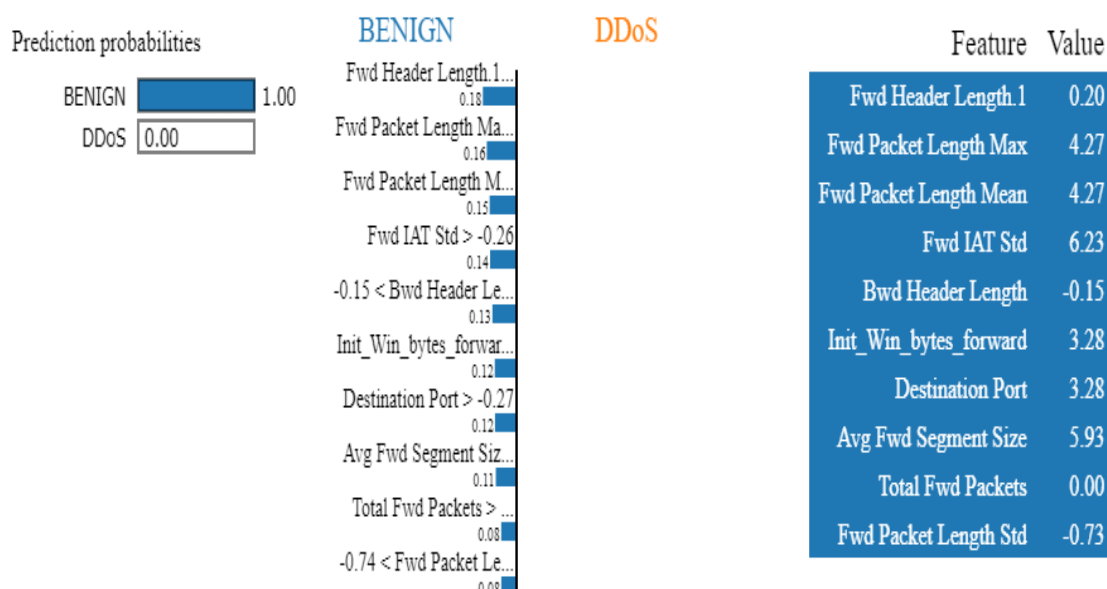


Figura 17: Análisis LIME para instancia Benign para modelo Perceptrón multicapa en el ataque DDoS

Respecto a la figura 18, esta ofrece otro ejemplo de una explicación LIME para el modelo perceptrón multicapa, en este caso con una instancia DDoS, es decir, una instancia clasificada como ataque.

Aunque algunas características tienen contribuciones que favorecen la clase BENIGN como son Fwd IAT Mean, Total Fwd Packets y Bwd Header Length, puede observarse que hay bastantes características que favorecen la clase DDoS (el resto). Esto resalta la capacidad del modelo para capturar las señales sutiles de un ataque DDoS, a pesar de la presencia de ciertas características que por sí solas podrían sugerir benignidad.

Así se obtiene la explicación de la predicción final de DDoS con una probabilidad del 100%. Las características como la desviación estándar de la longitud de los paquetes enviados (Fwd Packet Length Std), el número de bytes en subflujos enviados (Subflow Fwd Bytes) y la cantidad de paquetes de datos activos enviados (act\_data\_pkt\_fwd) son indicadores fuertes de tráfico malicioso.

### Caso instancia DDoS

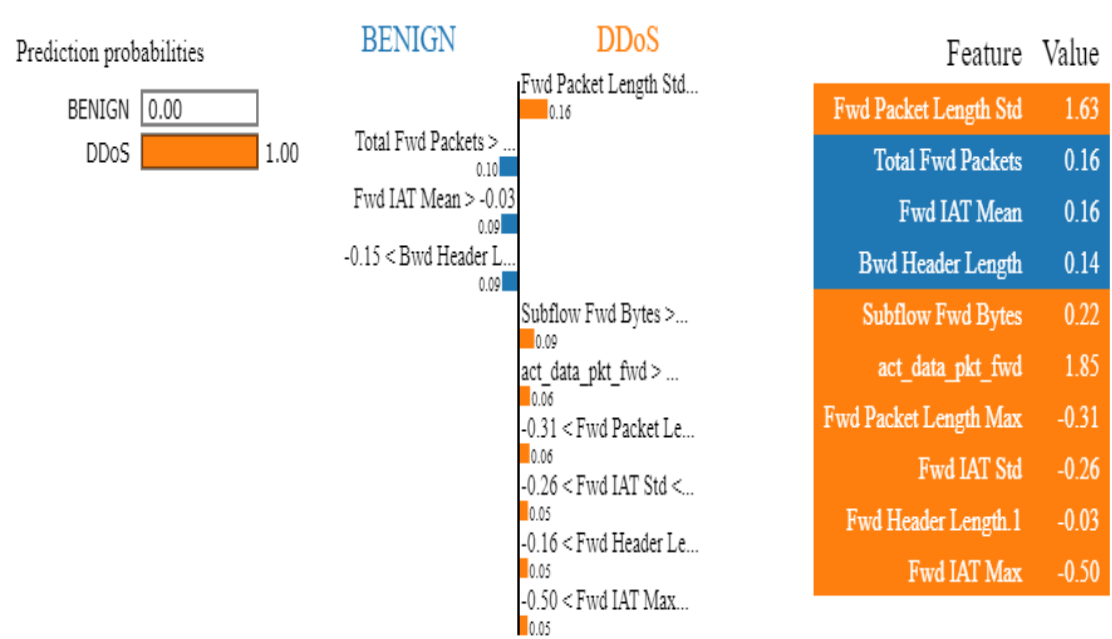


Figura 18: Análisis LIME para instancia DDoS para modelo Perceptrón multicapa en el ataque DDoS

Como conclusiones del análisis LIME, podemos resaltar las siguientes características:

- Contribuciones no lineales: la suma de las contribuciones no se traduce directamente en la probabilidad predicha, porque LIME es una aproximación lineal local a un modelo potencialmente no lineal y complejo.
- Predicciones robustas: el modelo puede predecir con alta certeza debido a la alta influencia de características específicas que son indicativas de DDoS, incluso si hay características que tengan menor influencia que podrían sugerir benignidad.
- Transparencia: LIME ofrece una explicación sobre qué características son más importantes para la predicción, permitiendo a los usuarios entender y confiar en las decisiones del modelo.
- Comparación con SHAP. SHAP proporciona bastante más información sobre cada instancia, incluyendo valores SHAP para cada característica, relación de esos valores con los valores reales de estas, y otros. No obstante, LIME proporciona explicaciones detalladas sobre cómo afectan las características a una predicción, y presenta menos problemas de rendimiento.

#### **6.1.5 Evaluación permutación y sensibilidad de características**

Llevando adelante el análisis de características realizado con el modelo XGBoost mediante SHAP, se identificó que la característica más influyente fue Fwd Packet Length Max. Se ha realizado una permutación específica para esta característica y este modelo a fin de observar cómo varía la precisión (figura 19). Aunque el modelo continúa mostrando una precisión generalmente "buena" debido a la cantidad de características e instancias, es importante notar cómo puede ajustarse a estos cambios y cómo algunas características específicas pueden afectarlo significativamente.

Para este caso particular:

```
Precisión original: 0.9998006335422989  
Precisión después de la permutación de la característica: 0.8251999202534169  
Cambio en la precisión: -0.17460071328888194
```

Figura 19: Permutación en la característica Fwd Packet Length Max para el modelo XGBoost en el ataque DDoS

Se puede observar que la precisión inicial del modelo es del 99.98%. Sin embargo, al realizar una permutación aleatoria de los valores de la característica mencionada, esta precisión disminuye significativamente en un 17.46%, alcanzando un 82.51% de precisión. Aunque sigue siendo una precisión aceptable, ya no alcanza el nivel excelente que tenía antes de la permutación.

Es importante destacar que, con las demás características, el modelo muestra una capacidad notable para ajustarse bien a cambios, con variaciones mínimas en la precisión. Además, la sensibilidad a las permutaciones puede variar dependiendo de cómo se alteren aleatoriamente los valores de cada característica.

Este análisis puede generalizarse a cualquier modelo y cualquier característica, permitiendo comparar e identificar cuáles características son más sensibles a las permutaciones de sus valores.

También es interesante realizar una evaluación mediante el análisis de sensibilidad centrado en una característica específica. En este caso, se ha tomado otra de las características más influyentes según SHAP, Bwd Packet Length Min (Longitud mínima de los paquetes recibidos desde atrás.) para el modelo XGBoost. En la figura 20, podemos observar cómo varía la predicción media del modelo a medida que se modifican los valores de esta característica seleccionada.

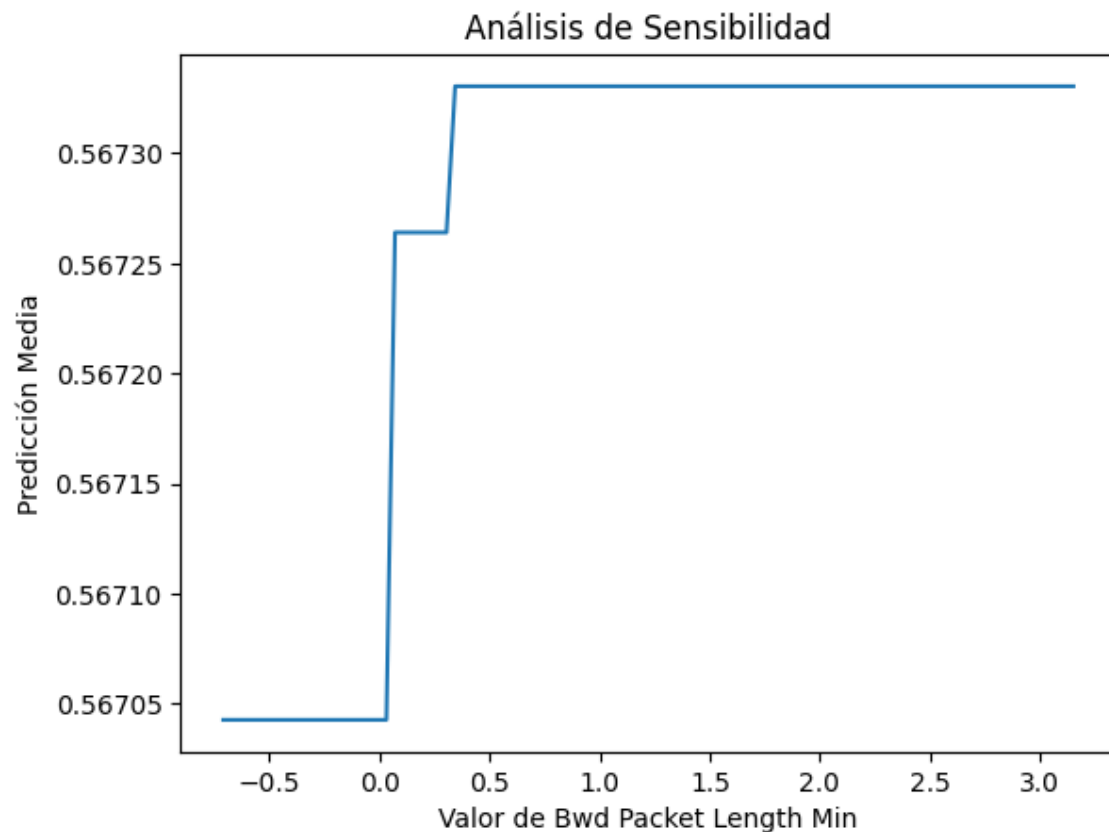


Figura 20: Análisis de sensibilidad para la característica Bwd Packet Length Min para el modelo XGBoost con el ataque DDoS

Se observa que el rango de la precisión media del modelo apenas varía, fluctuando entre 0.5670 y 0.5673. Sin embargo, se nota un aumento cuando el valor de la característica alcanza 0.5, a partir de donde se mantiene constante.

Para las demás características, ya sea en este modelo u otros, la interpretación de la gráfica sería la siguiente: una pendiente pronunciada indica que la característica tiene un gran impacto en las predicciones del modelo. Por el contrario, una curva relativamente plana sugiere que la característica tiene poco impacto en las predicciones del modelo. Es crucial observar siempre el rango de las predicciones medias y cómo varía a medida que aumenta el valor de la característica.

## 6.2 PortScan

### 6.2.1 Evaluación de modelos

En este apartado se presenta una tabla que resume las métricas de evaluación de varios modelos de clasificación aplicados al dataset CICIDS-2017, específicamente para la detección de ataques PortScan.

La tabla incluye la precisión, puntuaciones de validación cruzada, media y desviación estándar de las puntuaciones de la validación cruzada se puede ver en la tabla 3, así como las matrices de confusión tabla 4.

Modelo	Precisión	Validación Cruzada	Media	Desviación Estándar
Random Forest	0.9998	[0.999,0.999,0.999,0.999,0.998]	0.999	0.0003
Logistic Regression	0.9559	[0.946,0.962,0.951,0.957,0.955]	0.954	0.0050
MLP	0.9995	[0.999,0.998,0.999,0.999,0.998]	0.999	0.0005
LightGBM	0.9997	[0.999,0.999,0.999,0.999,0.998]	0.999	0.0003
XGBoost	0.9997	[0.999,0.999,0.999,0.999,0.998]	0.999	0.0003
SVM Linear	0.9754	[0.977,0.983,0.973,0.974,0.973]	0.976	0.0038
SVM Poly Degree 2	0.9505	[0.935,0.943,0.946,0.947,0.940]	0.942	0.0044
SVM Poly Degree 3	0.9354	[0.924,0.929,0.931,0.933,0.926]	0.929	0.0032
SVM RBF	0.9930	[0.993,0.993,0.991,0.993,0.990]	0.992	0.0012

Tabla 3: Resultados de Evaluación de Modelos para Detección de Ataques PortScan en el Dataset CICIDS-2017



Según lo presentado en la tabla 3, los modelos evaluados muestran, en general, un rendimiento sobresaliente en términos de precisión, con la mayoría alcanzando valores cercanos al 100%. Sin embargo, algunos modelos presentan una precisión ligeramente inferior respecto al ataque DDoS.

En concreto estos modelos son Logistic Regression cuya precisión es del 95.59%, SVM Lineal con una precisión del 97.54%, y los modelos SVM con Kernel polinómico, SVM Poly2 y SVM Poly3, con precisiones del 95.05% y 93.55% (la más baja) respectivamente.

Estos resultados permiten identificar que, aunque la mayoría de los modelos son altamente efectivos, algunos, como Logistic Regression y los modelos SVM con kernel polinómico, muestran áreas donde la optimización podría mejorar su rendimiento y consistencia en la detección de ataques PortScan.

Como ya se comentó en el ataque DDoS, el objetivo principal del proyecto no era optimizar los modelos, si no el de la explicabilidad tanto a nivel general como en casos concretos. Además, el análisis de los casos en los que los modelos fallan podría proporcionar información valiosa.

Modelo	Matriz de Confusión			
	TN	FP	FN	TP
Random Forest	25455	4	3	31758
Logistic Regression	23263	2196	326	31435
MLP	25438	21	3	31758
LightGBM	25451	8	4	31757
XGBoost	25450	9	3	31758
SVM Linear	24306	1153	253	31508
SVM Poly Degree 2	22701	2758	74	31687
SVM Poly Degree 3	21917	3542	151	31610
SVM RBF	25122	337	62	31699

Tabla 4: Resumen de Resultados de la Matriz de Confusión para la Detección de Ataques PortScan

Los resultados que se observan en la tabla de matrices de confusión indican, en líneas generales, que la mayoría de los modelos exhiben un desempeño excelente en la clasificación de ataques PortScan. Modelos como Random Forest, LightGBM y XGBoost se destacan por su eficacia, presentando muy pocos falsos positivos (FP) y falsos negativos (FN). Estos modelos demuestran una capacidad sobresaliente para identificar correctamente tanto los ataques (True Positives, TP) como los eventos no maliciosos (True Negatives, TN).

Sin embargo, se observa que las tasas de falsos positivos y falsos negativos son mayores en comparación con la detección de ataques DDoS, especialmente en los modelos que presentaban una menor precisión. El modelo de Regresión Logística, aunque efectivo, presenta un mayor número de falsos positivos en comparación con los modelos más avanzados como LightGBM y XGBoost. El MLP también ha mostrado un alto rendimiento, aunque con un número ligeramente mayor de falsos negativos en comparación con los modelos de árboles de decisión.

Los modelos SVM, excluyendo el kernel RBF, presentan tasas de error más altas, lo que es coherente con su menor precisión. El kernel RBF, tiene mejores tasas de clasificación, con una precisión del 99.30%, superando significativamente a los modelos SVM con kernel lineal y polinomiales. Estos últimos muestran un mayor número de errores de clasificación en comparación con los modelos basados en árboles y redes neuronales, debido a sus porcentajes de precisión más bajos.

### **6.2.2 Evaluación de residuos e influencia**

En esta sección se muestra un ejemplo de la interpretación de los residuos (las diferencias entre los valores predichos y los valores reales). La gráfica que se presenta a continuación (figura 21) representa la densidad del kernel de los residuos para el modelo de regresión logística.

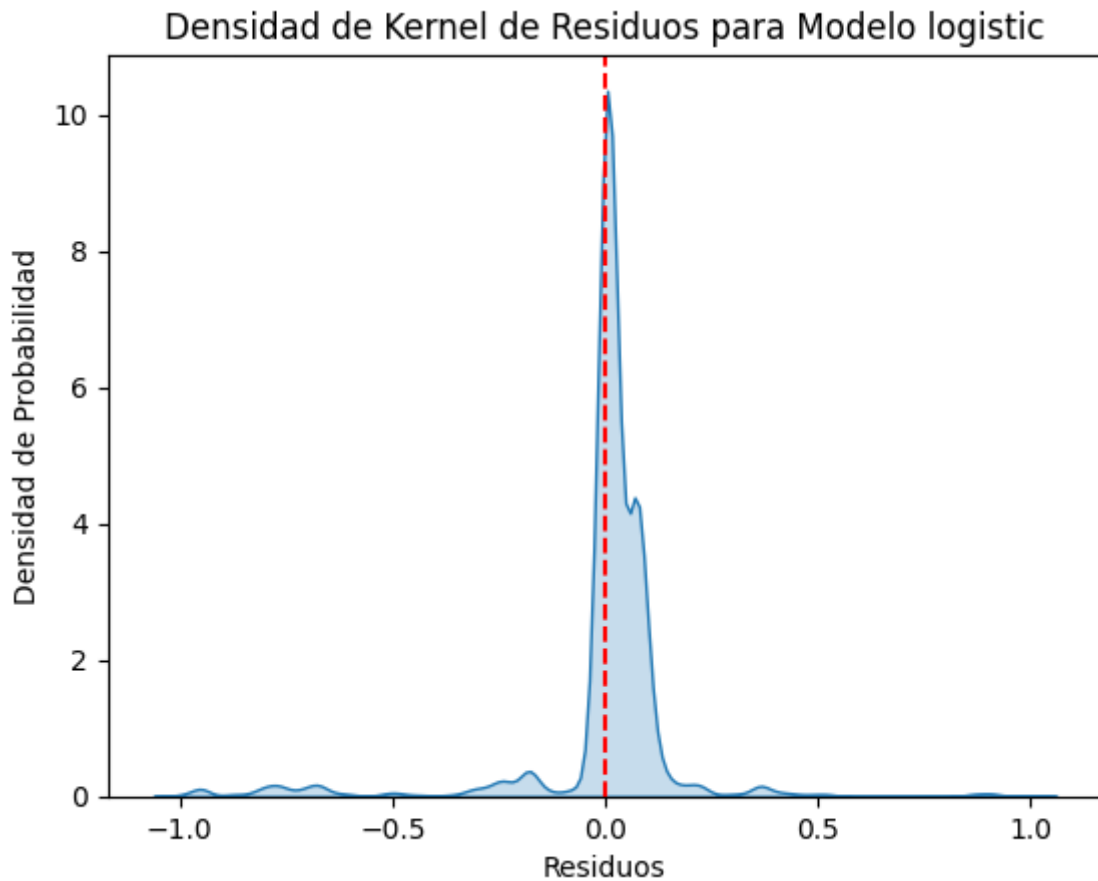


Figura 21: Gráfica de análisis de residuos para el modelo Logistic Regression para el ataque PortScan

La gráfica de la figura 4 representa la densidad de los residuos del modelo logístico, donde se observa que la mayoría de los residuos se encuentran cerca de cero. Esto sugiere que el modelo de regresión logística está bien ajustado a los datos. La línea roja punteada en el cero enfatiza esta concentración central.

La simetría de la distribución de los residuos en torno al cero indica la ausencia de un sesgo sistemático en las predicciones del modelo, es decir, no hay una tendencia significativa a sobrestimar o subestimar los valores reales.

Se observa la presencia de algunos residuos en los extremos de la distribución, más destacados que en el caso del ataque DDoS. Estos residuos son relativamente escasos y pueden corresponder a outliers o a casos donde el modelo no se ajusta tan bien. Cabe recordar que este modelo alcanzó una

precisión del 95.58%, lo que explica la diferencia de residuos entre este ataque y el anterior. La gráfica es coherente con esta precisión y refleja una situación común en cualquier modelo predictivo.

En la siguiente gráfica (figura 22), se evalúa la influencia de subconjuntos específicos de datos, obteniendo una comparación de las métricas de rendimiento del modelo antes y después de excluir una parte significativa del conjunto de prueba.

En este ejemplo, al igual que en el ataque DDoS, se ha utilizado un modelo SVM con kernel polinomial de grado 2 para ilustrar este análisis y el rango excluido de instancias ha sido **18000-35500** se puede aplicar la función a cualquier rango (que esté dentro del conjunto de test, este conjunto de datos es mayor que el del ataque DDoS, teniendo hasta 57220 instancias) y modelo.

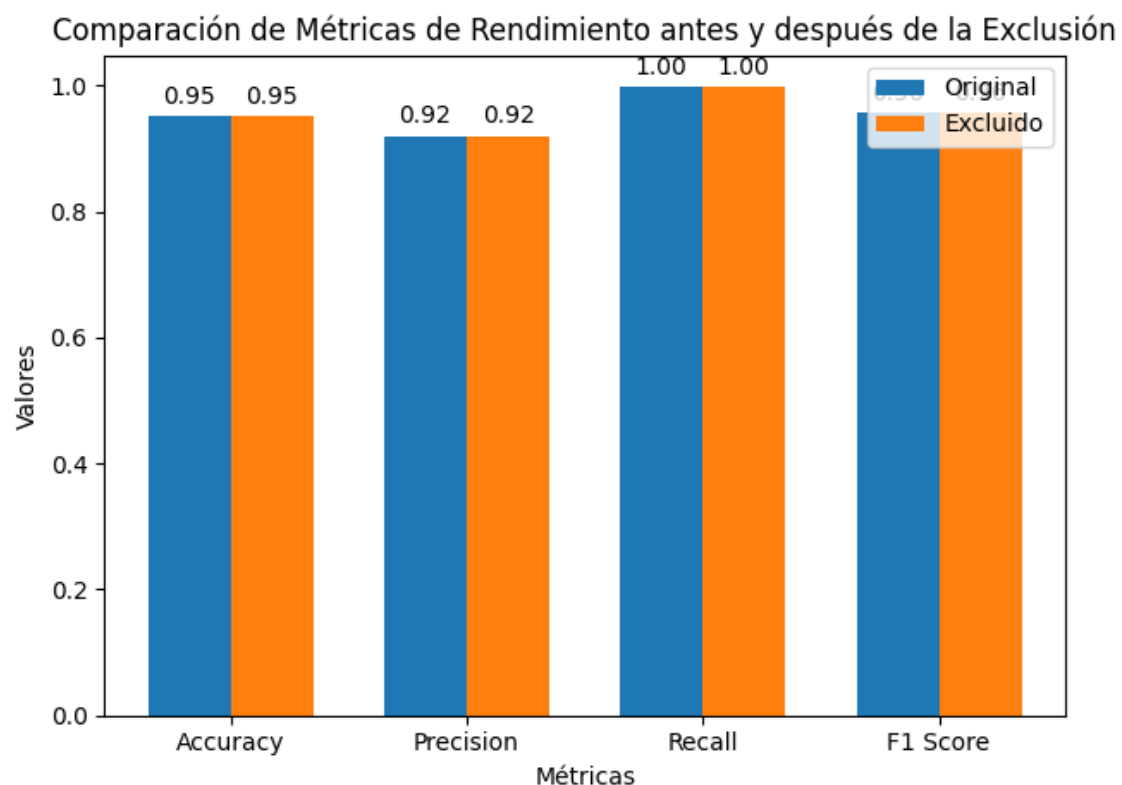


Figura 22: Gráfica de análisis de influencia para el modelo SVM Poly 2 para el ataque PortScan

La gráfica adjunta muestra que la exclusión de las instancias no ha afectado la exactitud, precisión, recall ni el F1 Score del modelo, lo que sugiere que el modelo mantiene un alto rendimiento incluso con un subconjunto reducido de datos. Este comportamiento ha sido consistente al probar varios subconjuntos y modelos.

La estabilidad en las métricas de rendimiento indica que el modelo SVM con kernel polinomial de grado 2 es robusto, fiable y consistente. La exclusión de una gran porción de los datos de prueba no afecta significativamente el rendimiento del modelo, lo que también demuestra una buena capacidad de generalización.

Este análisis permite a los usuarios comprender cómo se comporta el modelo cuando se excluye un subconjunto de datos y explica por qué el modelo puede ser confiable en diferentes escenarios.

### **6.2.3 SHAP**

En esta sección para poder contrastar mejor entre un ataque y otro, se va a realizar el análisis de SHAP, tanto de sus valores SHAP, como el análisis de instancias específicas con el mismo modelo.

Análisis valores SHAP para XGBoost. Se expone en la figura 23, la gráfica de valores SHAP correspondientes al modelo XGBoost.

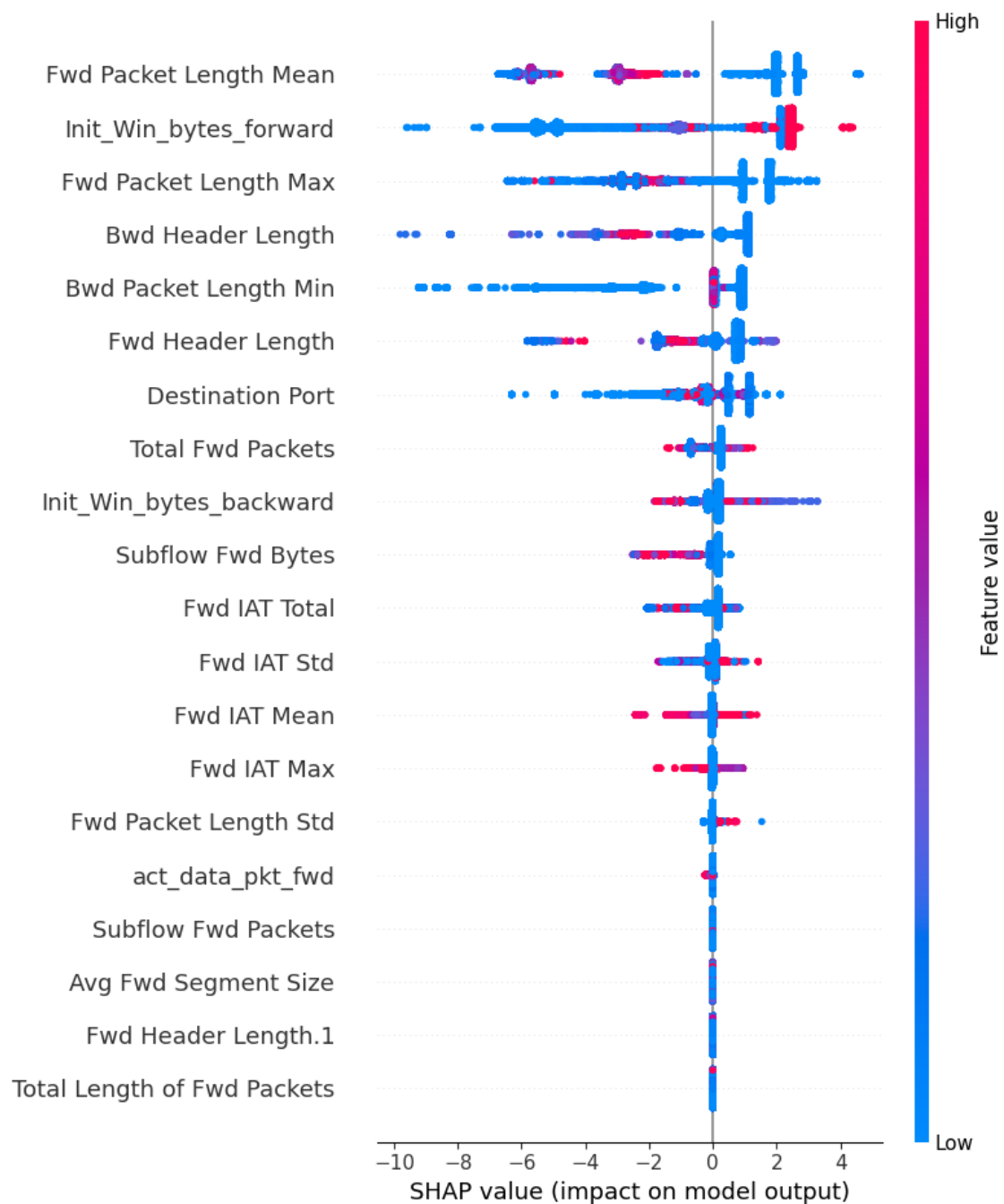


Figura 23: Valores SHAP para el modelo XGBoost para el ataque PortScan

En esta gráfica, se puede observar que los valores SHAP varían entre aproximadamente -10 y 4.

### Características Principales

- Fwd Packet Length Mean: esta característica tiene una influencia significativa en el modelo, con valores SHAP que varían ampliamente en

el rango comentado. Con valores altos disminuyendo la predicción de la clase positiva (PortScan).

- Init\_Win\_bytes\_forward: similar a la característica anterior, los valores SHAP para Init\_Win\_bytes\_forward muestran una variabilidad considerable, con un impacto notable en las predicciones del modelo. Los valores altos de esta característica están asociados con un aumento en la predicción del modelo, mientras los valores bajos la disminuyen.
- Fwd Packet Length Max: esta característica también muestra una variabilidad significativa en los valores SHAP, indicando su importancia en las predicciones del modelo. Los valores bajos de Fwd Packet Length Max tienden a aumentar la predicción, mientras que los valores altos la disminuyen.

La característica Fwd Packet Length Max se destaca como una de las más influyentes en distintos tipos de ataques, según el análisis de valores SHAP para los modelos basados en árboles. Su variabilidad considerable en los valores SHAP, tanto positiva como negativa, indica su importancia crítica en la predicción del modelo. En el caso de ataques específicos como DDoS y PortScan, los valores altos de Fwd Packet Length Max tienden a disminuir la probabilidad de una predicción de la clase positiva para la detección de ataques, mientras que los valores bajos pueden aumentar dicha probabilidad.

Este comportamiento consistente sugiere que Fwd Packet Length Max es un indicador robusto y confiable para la identificación de tráfico malicioso.

La notable influencia de esta característica permite una identificación temprana y precisa de posibles amenazas, mejorando la capacidad del sistema de ciberseguridad para tomar medidas preventivas y correctivas de manera eficiente.

### Análisis valores SHAP para otros modelos.

Al comparar con otros modelos como LightBM figura 24, esta gráfica ilustra como el mismo conjunto de características tienden a ser las más influyentes.

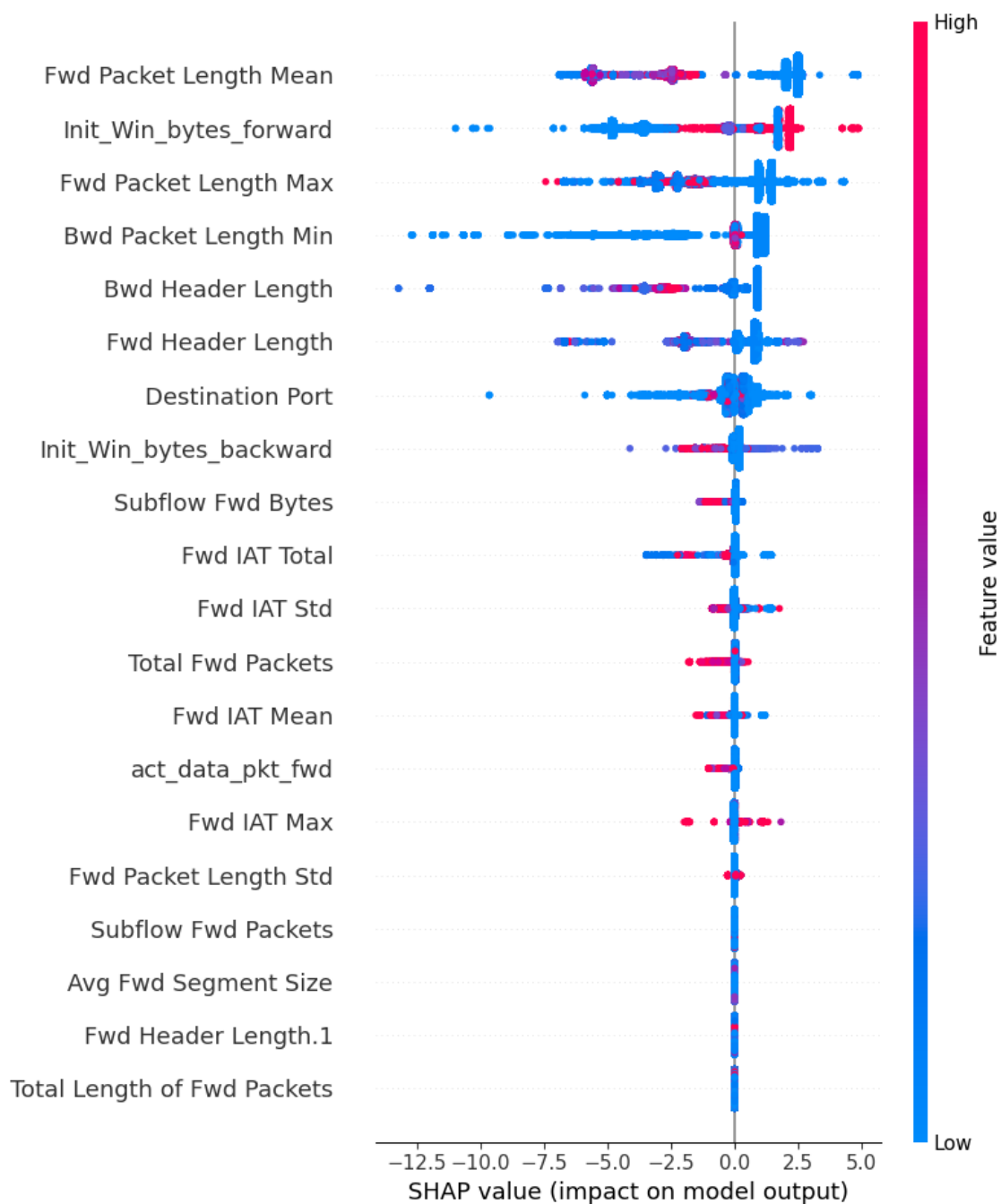


Figura 24: Valores SHAP para el modelo LightBM para el ataque PortScan

En el caso de LightBM tiene un rango incluso mayor de los valores SHAP que impactan en el modelo, pero la influencia de características es muy similar a la



del modelo XGBoost tanto en la influencia de las características como el comportamiento específico de cada una de forma general.

Con los modelos lineales, cuyo explicador es lineal o se puede ajustar de manera lineal, en este caso SVM Lineal (figura 25) y Logistic Regression, si tienen unos valores SHAP significativamente distintos cambiando así la influencia de las características en comparación con los modelos basados en árboles.

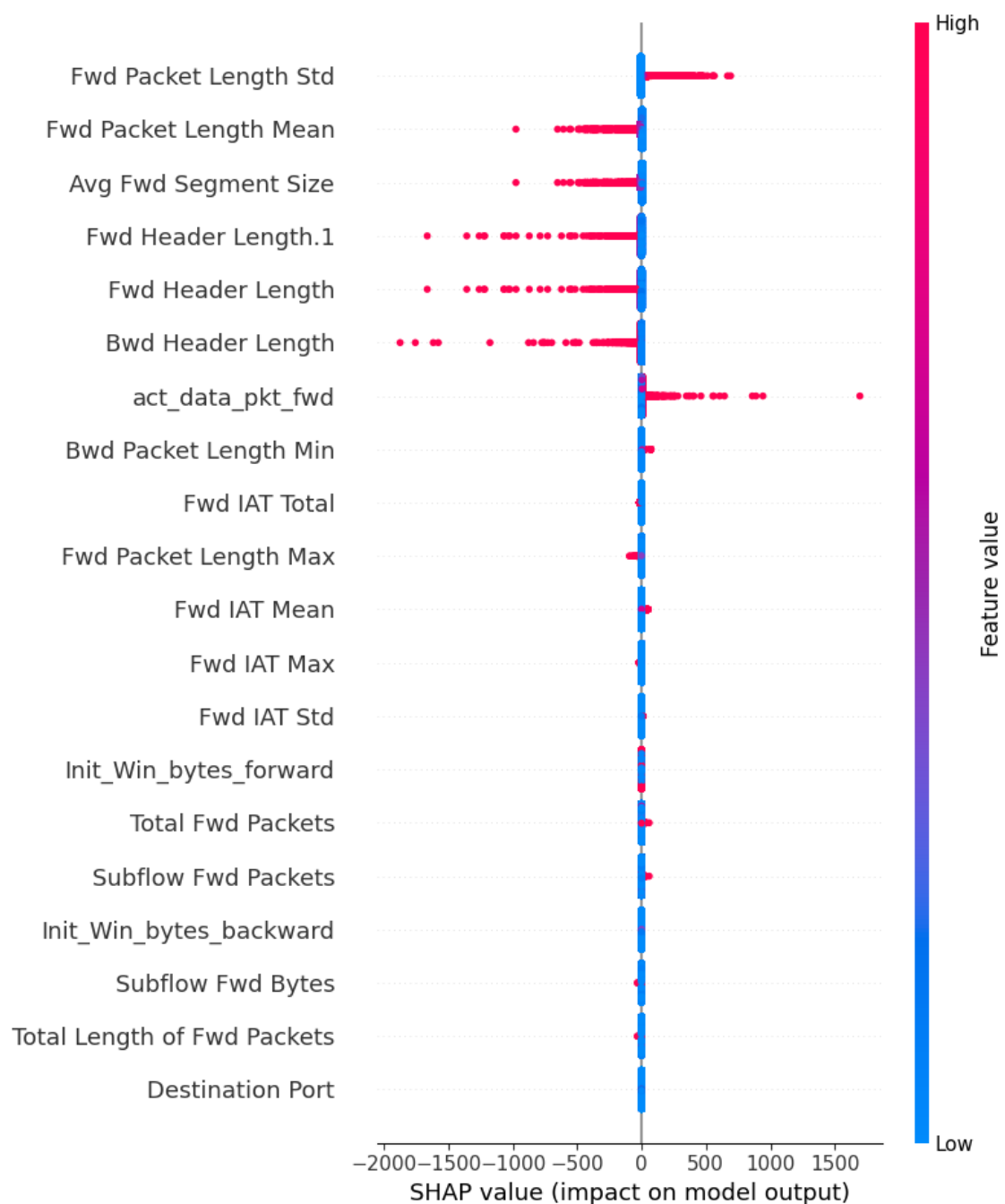


Figura 25: Valores SHAP para el modelo SVM Lineal para el ataque PortScan

En esta visualización, se observa que la mayoría de los valores SHAP están centrados alrededor de cero, con un rango de impacto mucho mayor en comparación con el modelo XGBoost. Esto sugiere diferencias significativas en cómo ambos modelos manejan y ponderan las características para realizar sus predicciones.

### **Diferencias Notables con el Modelo XGBoost**

**Centrado de valores SHAP:** a diferencia del modelo XGBoost, los valores SHAP en el modelo SVM lineal están más concentrados alrededor del cero. Este centrado indica que, en promedio, las características tienen un impacto menos extremo en la predicción del modelo SVM en comparación con XGBoost, donde los valores SHAP mostraban una mayor dispersión.

**Rango de impacto mayor:** el rango de valores SHAP en el modelo SVM lineal es considerablemente más amplio, extendiéndose desde aproximadamente -2000 hasta 1500. Este rango más amplio sugiere que el SVM lineal puede asignar impactos mucho más extremos a ciertas características en comparación con XGBoost. Esto puede deberse a la naturaleza de los modelos: mientras que XGBoost, un modelo de ensamblaje basado en árboles puede manejar interacciones no lineales y complejas entre características, el SVM lineal se enfoca en encontrar una separación óptima en un espacio de características linealmente transformado.

**Distribución de características:** en el modelo SVM lineal, características como Fwd Packet Length Std, Fwd Packet Length Mean y Avg Fwd Segment Size tienen valores SHAP que se extienden ampliamente, mostrando que estas características pueden tener un impacto significativo tanto positivo como negativo en las predicciones. En comparación, el modelo XGBoost mostraba un impacto más modulado de las características.

La diferencia en la centralización y el rango de los valores SHAP entre SVM lineal y XGBoost puede atribuirse a la forma en que estos modelos procesan y ponderan las características:

- SVM Lineal: este modelo busca una frontera de decisión lineal en el espacio de características. Por lo tanto, cualquier característica que ayude a maximizar el margen entre las clases tendrá un impacto considerable, lo que explica los valores SHAP más extremos.
- XGBoost: al ser un modelo basado en árboles, XGBoost puede capturar interacciones no lineales complejas y realizar ajustes finos en las predicciones, resultando en valores SHAP más dispersos, pero menos extremos en general.

Con el resto de los modelos no podemos sacar conclusiones por los problemas relacionados con el rendimiento explicados en el apartado 5.1.3.1.

### **Valores SHAP para analizar instancias específicas.**

Para un análisis más profundo y detallado, se va a examinar instancias específicas y sus correspondientes valores SHAP con el modelo XGBoost, del cual ya hemos examinado y analizado sus valores SHAP. Se presentarán muestras Benign en las figuras 26, 27 y 28, así como una muestra de tipo PortScan en las figuras 29, 30, y 31.

## Caso instancia Benign

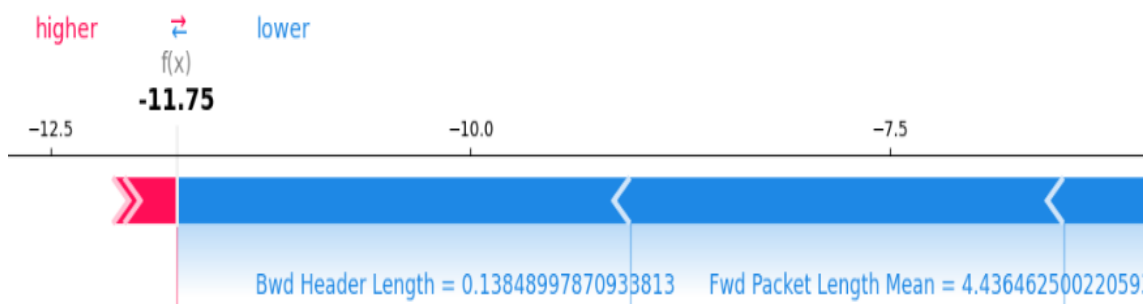


Figura 26: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque PortScan- Parte1

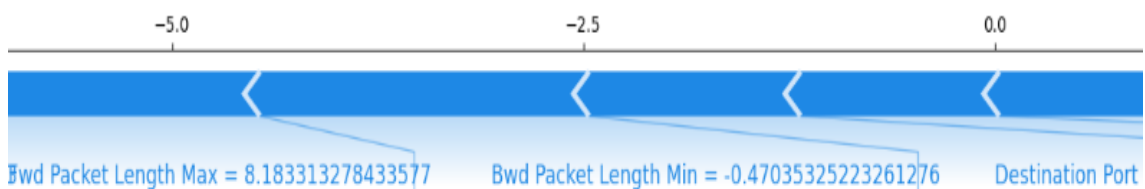


Figura 27: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque PortScan- Parte1

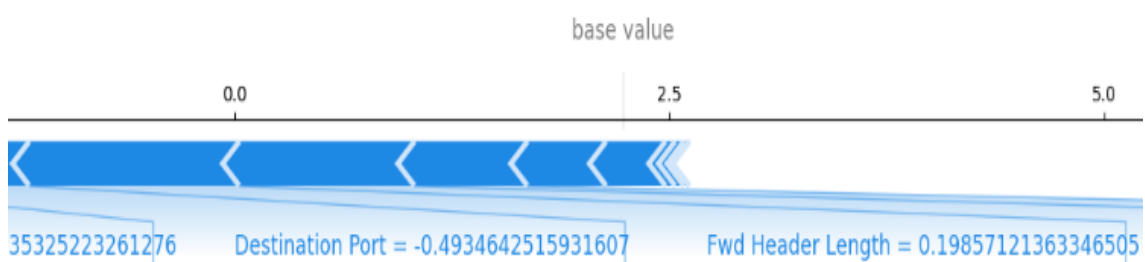


Figura 28: Análisis SHAP para instancia Benign para modelo XGBoost en el ataque PortScan- Parte3

A partir de las figuras presentadas, se puede observar que el valor base de la predicción es aproximadamente 2.25 (Figura 28). Este valor actúa como punto de partida, y las características más representativas para esta instancia desvían la predicción respecto a este valor base.

Cada valor SHAP indica cuánto ha contribuido una característica a disminuir la predicción de la clase positiva PortScan, desde el valor base de 3 hasta -11.75 (Figura 26). Este último valor se obtiene al sumar el valor base y los valores SHAP de todas las características para esa instancia.

La magnitud de cada flecha en las figuras indica la fuerza de la contribución de esa característica a la predicción.

1. Bwd Header Length (0.13848997870933813, valor normalizado de la característica)
  - Esta característica tiene la mayor contribución negativa, reduciendo significativamente el valor de la predicción.
  - Valor SHAP: aproximadamente -2.75 (Figura 26)
2. Fwd Packet Length Mean (4.43646250022059, valor normalizado de la característica)
  - También contribuye negativamente de manera significativa.
  - Valor SHAP: aproximadamente -2.5 (Figura 26)
3. Bwd Packet Length Max (8.183313278433577, valor normalizado de la característica)
  - Contribuye negativamente, aunque en menor medida que las dos anteriores.
  - Valor SHAP: aproximadamente -2 (Figura 27)
4. Bwd Packet Length Max (-0.47035325223261276, valor normalizado de la característica)
  - Esta característica también tiene una contribución negativa.
  - Valor SHAP: aproximadamente -1.5 (Figura 27)

Al revisar la figura 23, se destaca que Fwd Packet Length Mean es una de las características más influyentes según XGBoost. En este caso particular, como se mencionó anteriormente, esta característica presenta un valor SHAP de aproximadamente -2.5.

En el gráfico de valores SHAP, se observa que este rango del Valor SHAP entorno al -2.5 para dicha característica está mayoritariamente representada por valores medios y altos (puntos rojos y morados).

En el caso específico del ataque PortScan, esta característica puede tener valores cercanos a 0. Sin embargo, en esta instancia particular, con un valor de 4.4364, se ubica claramente en el rango de valores altos. Esta observación confirma la influencia significativa de Fwd Packet Length Mean en la predicción del modelo y refuerza la coherencia del modelo en el uso de esta característica para realizar predicciones precisas.

Si observamos otras características, como Bwd Header Length, que es la más influyente en esta instancia específica, encontramos que tiene un valor bajo, correspondiente a uno de los pocos puntos azules en el rango de valores SHAP alrededor de -2.75.

Estas mismas observaciones se aplican a las demás características presentes en esta instancia específica.

### Caso instancia PortScan

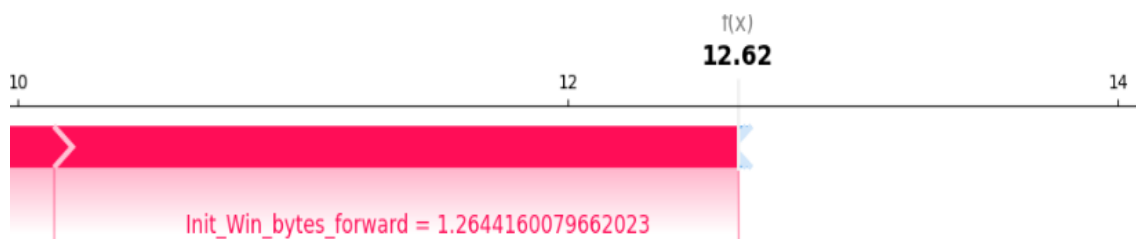


Figura 29: Análisis SHAP para instancia PortScan para modelo XGBoost en el ataque PortScan- Parte1

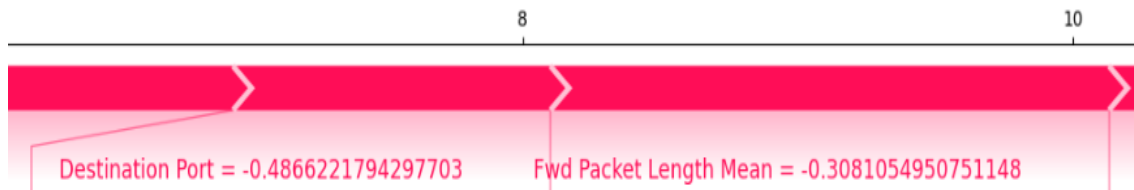


Figura 30: Análisis SHAP para instancia PortScan para modelo XGBoost en el ataque PortScan- Parte2

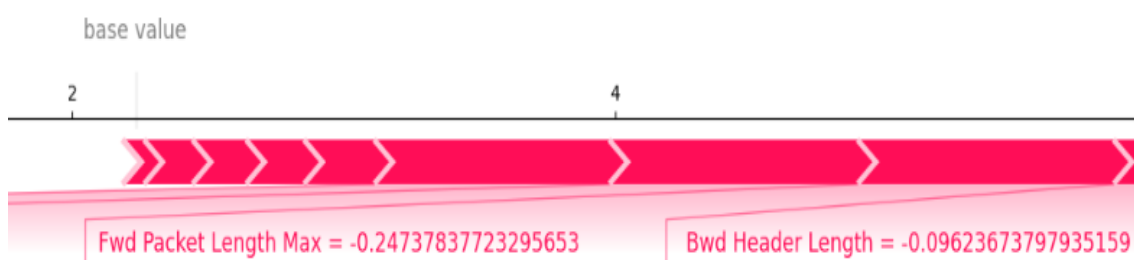


Figura 31: Análisis SHAP para instancia PortScan para modelo XGBoost en el ataque PortScan- Parte3

Como se observa en la última imagen, el valor base se mantiene inalterado en aproximadamente 2.25. No obstante, el efecto de las características en esta instancia ha provocado que cada valor SHAP contribuya a incrementar la predicción de la clase positiva PortScan, desde 2.25 (valor base) hasta 12.62 (valor predicho real, figura 12).

Es evidente que varias características aumentan la predicción, destacando entre ellas las tres más influyentes según el gráfico de valores SHAP: Init\_Win\_bytes\_forward, Fwd Packet Length Mean y Fwd Packet Length Max.

Al observar el valor SHAP de la característica Fwd Packet Length Mean, se aprecia que incrementa la predicción de la clase positiva. Con un valor de -0.30810, impacta con un valor SHAP de aproximadamente 2. En el caso Benign, esta misma característica disminuía la predicción con un valor mucho

más alto de 4.4364, lo que provocaba un valor SHAP de aproximadamente -2.5.

Referenciando nuevamente la figura 23 de los valores SHAP, en la instancia actual el valor de Fwd Packet Length Mean es considerablemente menor que en la instancia previa, lo que se corresponde con uno de los puntos azules en la gráfica de valores SHAP, alrededor de los valores positivos entre 2 y 3, concordando con la observación actual y con la anterior.

El resto de las características, como Init\_Win\_bytes\_forward —la más influyente en esta instancia específica— con un valor de 1.2644, impacta con un valor SHAP de aproximadamente 2.5. Esto se verifica nuevamente en la gráfica de valores SHAP, observando que pertenece a uno de los valores altos de la característica en el rango de 2 a 4.

### **Valores SHAP vs Valores reales**

Finalmente, en la sección de SHAP, se evaluará la relación entre los valores reales de la característica Fwd Packet Length Mean y sus correspondientes valores SHAP, en el contexto del modelo XGBoost. Esta característica ha mostrado contribuir tanto positiva como negativamente en las dos instancias específicas analizadas: Benign y PortScan. Para ilustrar esta relación, se presenta una gráfica específica (figura 32). Es importante recordar que los valores reales de las características fueron normalizados desde el inicio del entrenamiento de los modelos, una medida que mejora la convergencia de los algoritmos y la precisión de las predicciones, como se explicó en el apartado 3.3.3.



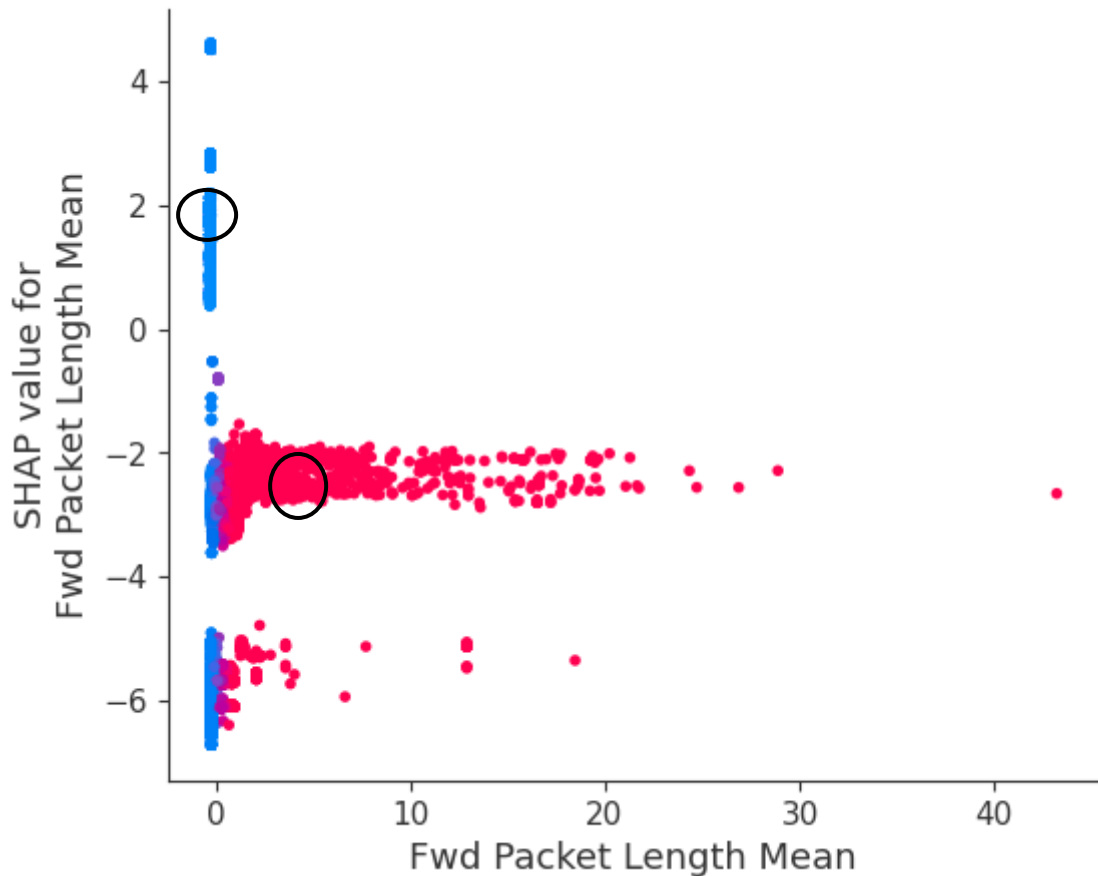


Figura 32: Relación de valores SHAP con valores reales (normalizados) para característica Fwd Packet Length Mean con el modelo XGBoost para el ataque PortScan

Al analizar el gráfico, se puede observar que la mayoría de los puntos de datos se concentran en valores bajos de "Fwd Packet Length Mean" (entre -0.5 y 10).

La densidad de los puntos es mayor en la región donde "Fwd Packet Length Mean" es bajo, lo que sugiere que esta característica tiene una mayor variabilidad de impacto en esa zona.

El examen de casos específicos dentro del gráfico refuerza la coherencia y consistencia de los resultados obtenidos. Cuando el valor de la característica real es de -0.30810, este valor impacta con un valor SHAP de aproximadamente 2 (marcado).

Y cuando el valor real de la característica es de 4.4364 este valor impacta con un valor SHAP de aproximadamente -2.5 (marcado).

La observación de los valores específicos mencionados es coherente con la distribución general de los datos en el gráfico. Un valor bajo de "Fwd Packet Length Mean" puede tener un impacto positivo significativo en la predicción del modelo, mientras que un valor más alto puede tener un impacto negativo igualmente significativo. Esta variabilidad sugiere que el modelo está utilizando de manera efectiva esta característica para ajustar sus predicciones, dependiendo del contexto específico de cada observación.

6.2.4 LIME

Caso instancia Benign

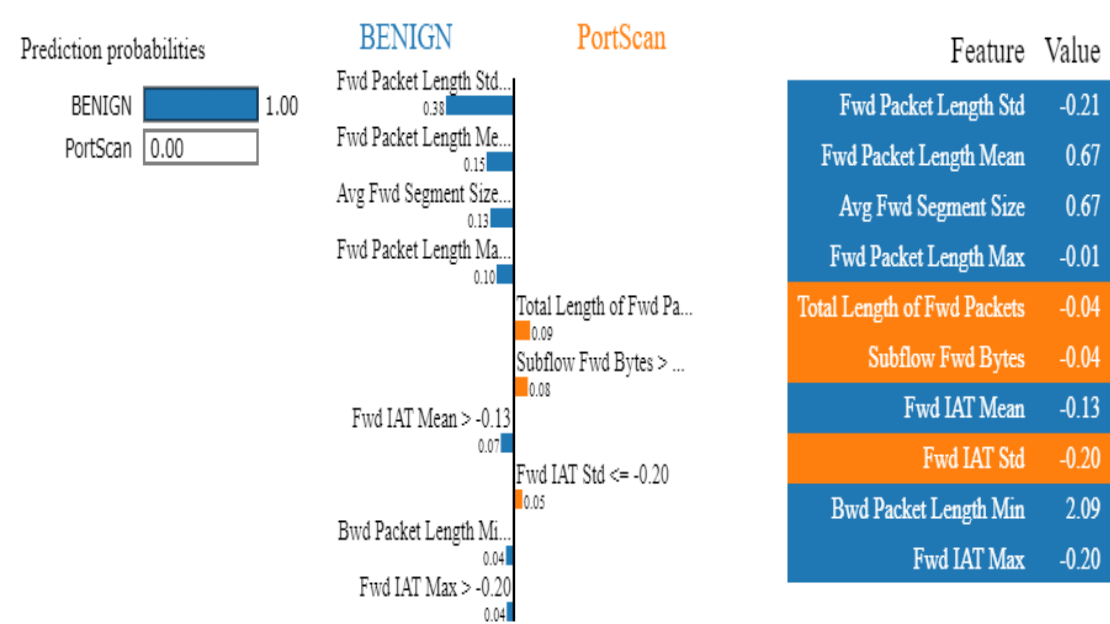


Figura 33: Análisis LIME para instancia Benign para modelo Perceptrón multicapa en el ataque PortScan

La figura 33 ofrece una visión clara de cómo diferentes características del tráfico de red influyen en la predicción del modelo. A continuación, se detallan las contribuciones específicas de estas características y su impacto en la clasificación final:

1. Características que favorecen la clase PortScan:

- Fwd IAT Std (Desviación Estándar del Intervalo de Llegada de Paquetes Forward): esta característica sugiere variabilidad en los intervalos de llegada de paquetes, lo que puede ser un indicativo de actividad sospechosa.
- Total Length of Fwd Packets (Longitud Total de los Paquetes Forward): puede ser asociado con escaneos de puertos o actividad anómala.
- Subflow Fwd Bytes (Bytes Forward de Subflujos): puede indicar un posible escaneo de puertos.

2. Características que favorecen la clase Benign:

- Fwd Packet Length Std (Desviación Estándar de la Longitud de los Paquetes Forward): esta característica muestra una alta consistencia en la longitud de los paquetes enviados, lo que es típico de tráfico benigno.
- Fwd Packet Length Mean (Media de la Longitud de los Paquetes Forward): una media alta de la longitud de los paquetes forward indica una transferencia de datos regular y continua.
- Avg Fwd Segment Size (Tamaño Promedio de los Segmentos Forward): consistencia en el tamaño de los segmentos forward es otro indicativo de tráfico benigno.

A pesar de la presencia de ciertas características que podrían sugerir actividad maliciosa (como Fwd IAT Std, Total Length of Fwd Packets y Subflow Fwd Bytes), el modelo ha determinado que la mayoría de las características analizadas favorecen una clasificación Benign. Esto resalta la capacidad del modelo para capturar y priorizar las señales sutiles que indican benignidad, incluso cuando algunas características por sí solas podrían sugerir un ataque PortScan.

Caso instancia PortScan

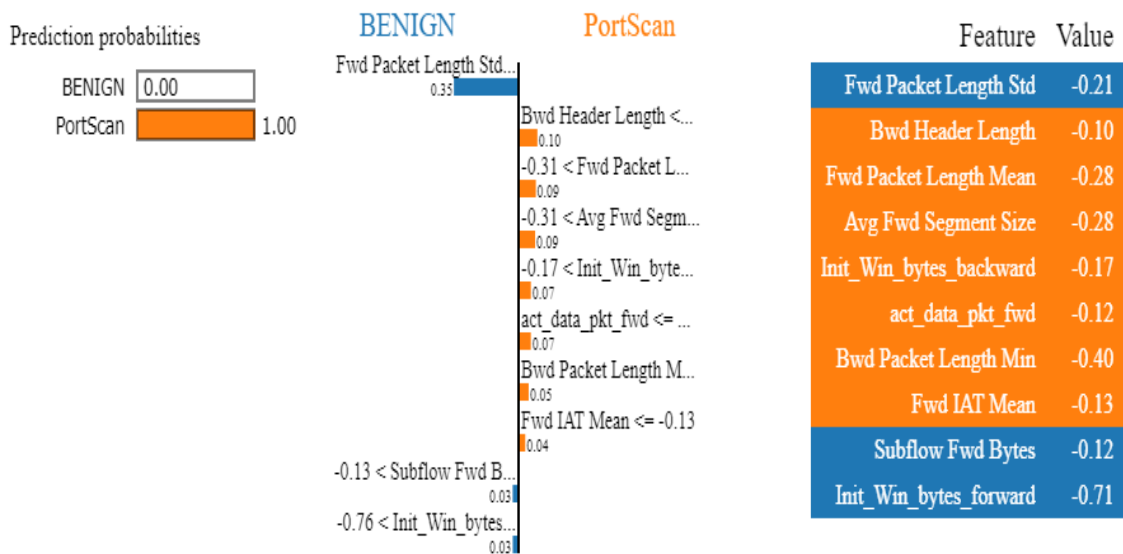


Figura 34: Análisis LIME para instancia PortScan para modelo Perceptrón multicapa en el ataque PortScan

A continuación, se presentan las características con sus respectivos valores y contribuciones de la figura 34:

1. Fwd Packet Length Std (Estándar de longitud de paquete enviado)
  - Valor: -0.21
  - Contribución significativa hacia "Benign".
2. Bwd Header Length (Longitud de cabecera recibida)
  - Valor: -0.10
  - Contribución hacia "PortScan".
3. Fwd Packet Length Mean (Media de longitud de paquete enviado)
  - Valor: -0.28
  - Contribución notable hacia "PortScan".
4. Avg Fwd Segment Size (Tamaño promedio de segmento enviado)
  - Valor: -0.28
  - Contribución hacia "PortScan".
5. Init\_Win\_bytes\_backward (Bytes iniciales de ventana hacia atrás)
  - Valor: -0.12
  - Contribución hacia "PortScan".

El modelo clasifica la muestra como un ataque de tipo PortScan con una certeza del 100%. Esto indica que las características de la muestra son altamente indicativas de un ataque de este tipo.

Las características más influyentes que apoyan la clasificación como PortScan incluyen Fwd Packet Length Mean, Avg Fwd Segment Size, y Fwd IAT Mean. Por otro lado, Fwd Packet Length Std la característica con mayor impacto que sugiere una naturaleza benigna, pero su impacto no es suficiente para cambiar la clasificación global.

#### **6.2.5 Evaluación permutación y sensibilidad de características**

En el análisis realizado utilizando el modelo XGBoost, con el apoyo de SHAP, se determinó que la característica con mayor influencia es Fwd Packet Length Mean. Para evaluar el impacto de esta característica en la precisión del modelo, se llevó a cabo una permutación específica, cuyos resultados se presentan en la figura 35.

```
Precisión original: 0.999790283117791  
Precisión después de la permutación de la característica: 0.8896364907375044  
Cambio en la precisión: -0.11015379238028655
```

Figura 35: Permutación en la característica Fwd Packet Length Mean para el modelo XGBoost en el ataque PortScan

Inicialmente, el modelo presenta una precisión excepcional del 99.97%. Sin embargo, al permutar aleatoriamente los valores de una característica particular, la precisión experimenta una disminución significativa del 11.01%, resultando en una nueva precisión de 88.96%.

Aunque una precisión del 88.96% sigue siendo aceptable, es evidente que la precisión ya no alcanza el nivel excelente que se observaba antes de la permutación.

Este resultado subraya la sensibilidad del modelo a los cambios en características específicas. A pesar de mantener una precisión globalmente "buena" debido a la cantidad de características e instancias consideradas, es

crucial destacar que la alteración en una sola característica puede tener un impacto considerable en el rendimiento del modelo.

Es fundamental llevar a cabo una evaluación a través de un análisis de sensibilidad enfocado en una característica específica. En este caso, continuamos con la característica identificada como la más influyente por SHAP: la longitud media de los paquetes enviados hacia adelante (Fwd Packet Length Mean) en el modelo XGBoost. La figura 36 ilustra cómo varía la predicción promedio del modelo cuando se alteran los valores de esta característica seleccionada.

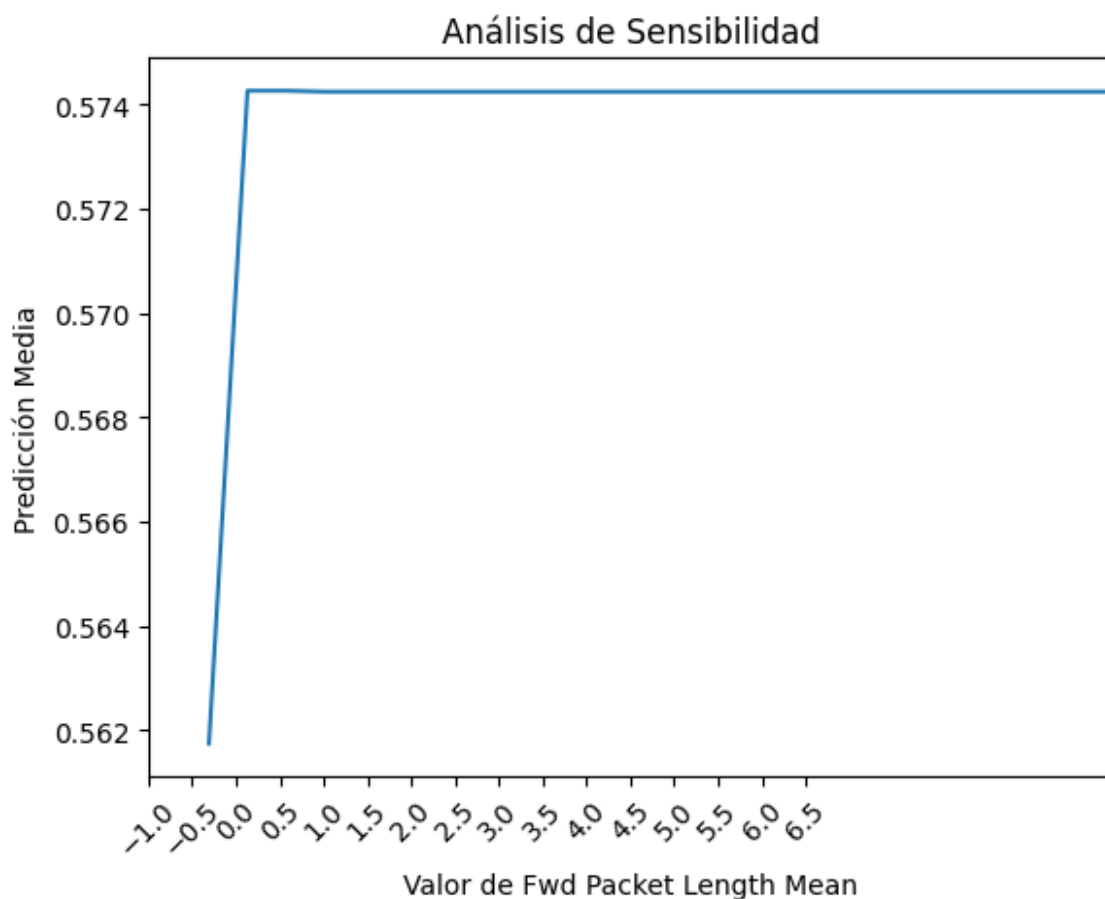


Figura 36: Análisis de sensibilidad para la característica Fwd Packet Length Mean para el modelo XGBoost con el ataque PortScan

Se puede observar en el eje y, la predicción media del modelo, con valores que oscilan entre 0.562 y 0.574. La gráfica revela varios puntos clave sobre la sensibilidad del modelo respecto a la característica "Fwd Packet Length Mean":

La predicción media del modelo se mantiene constante en aproximadamente 0.574 para valores de "Fwd Packet Length Mean" desde 0.0 en adelante.

Hay un incremento notable en la predicción media del modelo cuando el valor de "Fwd Packet Length Mean" pasa de -1.0 a 0.0. Esta rápida transición sugiere una alta sensibilidad del modelo a esta característica en el rango negativo. A partir de un valor de aproximadamente 0.0 en adelante, la predicción media se estabiliza, indicando que el modelo es menos sensible a los cambios en "Fwd Packet Length Mean" en este rango.

Para valores positivos de "Fwd Packet Length Mean", el modelo mantiene una predicción media estable, indicando que otros factores o características podrían tener mayor relevancia en este rango.

# Conclusiones y Líneas futuras

## 7.1 Conclusiones

La explicabilidad de los modelos de aprendizaje automático es una herramienta crucial para entender el comportamiento y las decisiones de los modelos, especialmente en contextos críticos como la detección de ataques cibernéticos. A lo largo de este análisis, hemos explorado diversas técnicas de explicabilidad, enfocándonos principalmente en SHAP y LIME.

### Utilidad de SHAP y LIME

- SHAP: SHAP ha demostrado ser una herramienta más robusta y precisa para la explicabilidad global y local de los modelos. Su capacidad para proporcionar valores aditivos de Shapley que explican la contribución de cada característica en una predicción específica lo hace particularmente útil para entender el impacto de las características a nivel granular.
- LIME: aunque LIME es útil para obtener explicaciones locales de manera rápida, su precisión y consistencia pueden ser inferiores a las de SHAP. LIME es más adecuado para una comprensión rápida y menos detallada del modelo.



En este análisis, se ha dado especial atención a los modelos basados en árboles, particularmente XGBoost, que ha demostrado ser altamente efectivo para la detección de ataques cibernéticos. XGBoost, con su capacidad para manejar grandes cantidades de datos y capturar interacciones complejas entre características, ha sido el modelo más analizado y ha proporcionado resultados significativos en la identificación de patrones de tráfico malicioso.

### **Características claves en la detección de ataques**

El análisis ha identificado varias características críticas para la detección de ataques específicos:

- Características comunes: algunas características son relevantes para la detección de múltiples tipos de ataques. Por ejemplo, la **máxima longitud de paquetes forward (Fwd Packet Length Max)** es una característica clave tanto en ataques DDoS como en PortScan.
- Ataque DDoS: las características específicas más influyentes incluyen:
  - **Init\_Win\_bytes\_backward**: esta característica refleja el tamaño inicial de la ventana de recepción en la dirección backward y es crucial para identificar comportamientos anómalos en ataques DDoS.
  - **Fwd IAT Max**: el intervalo máximo de llegada de paquetes forward también es significativo para distinguir patrones de tráfico en ataques DDoS.
- Ataque PortScan: para los ataques PortScan, las características más relevantes son:
  - **Init\_Win\_bytes\_forward**: esta característica es crucial para detectar actividades de escaneo de puertos, ya que refleja el tamaño inicial de la ventana de recepción en la dirección forward.
  - **Fwd Packet Length Mean**: la media de la longitud de los paquetes forward ayuda a identificar patrones específicos de tráfico asociados con los escaneos de puertos.

Las explicaciones de instancias específicas han sido cruciales para entender cómo las características individuales afectan las predicciones del modelo en casos concretos. Estas explicaciones permiten una visión detallada y personalizada, facilitando la identificación de características críticas y su impacto en la decisión del modelo.

Además, técnicas como la permutación de características han demostrado ser útiles para evaluar la importancia de cada característica en el rendimiento del modelo. Al permutar aleatoriamente los valores de una característica específica, se puede observar cómo la precisión del modelo se ve afectada, destacando la sensibilidad del modelo a cambios en características clave.

## **7.2 Líneas futuras**

El desarrollo y perfeccionamiento de modelos de aprendizaje automático para la detección de ciberamenazas requiere un enfoque integral que abarque tanto mejoras generales como específicas. A nivel general, es fundamental centrar esfuerzos en mejorar el rendimiento de métodos clave de la librería SHAP, como DeepExplainer y KernelExplainer, que permiten interpretaciones precisas de modelos complejos, incluyendo redes neuronales profundas. Estas mejoras deben ir acompañadas de un esfuerzo paralelo en la creación de una mayor cantidad de documentación y la promoción de código libre para estas librerías. Proveer recursos accesibles y detallados no solo facilitará la adopción de estas tecnologías por parte de la comunidad, sino que también fomentará la colaboración y la innovación en el campo del aprendizaje automático aplicado a la ciberseguridad.

Otra área crucial por explorar es la desnormalización de datos en casos específicos para visualizar valores reales. Esto permitirá a los analistas entender mejor las características y su impacto en los modelos, proporcionando una base sólida para la interpretación y toma de decisiones. Además, es esencial realizar un análisis exhaustivo de los casos en los que el modelo falle. Identificar y comprender estos fallos ofrecerá insights valiosos sobre las

limitaciones y áreas de mejora de los modelos actuales, posibilitando ajustes y refinamientos que aumenten su precisión y efectividad.

La generación de más explicaciones a nivel de instancia utilizando explicadores lineales en otros modelos también representa una línea prometedora de investigación. Este enfoque puede proporcionar una comprensión más detallada y específica de cómo diferentes modelos toman decisiones, lo cual es crucial para la interpretación y confianza en los sistemas de detección de amenazas. Comparar los resultados de diferentes modelos y sus explicaciones puede revelar patrones e insights que pueden ser utilizados para mejorar las estrategias de detección.

Es fundamental ampliar el análisis para incluir una mayor variedad de ataques y modelos de aprendizaje automático. Evaluar diferentes tipos de ciberamenazas y aplicar diversos algoritmos permitirá obtener una comprensión más completa de las fortalezas y debilidades de cada enfoque. Este análisis exhaustivo no solo mejorará la capacidad de los modelos para detectar una gama más amplia de amenazas, sino que también contribuirá al desarrollo de soluciones más robustas y adaptables en el ámbito de la ciberseguridad.

La XAI ya se está utilizando en empresas líderes como Ericsson. Según un artículo reciente [70] (IT Trends, 2024), Ericsson ha destacado las ventajas que aporta XAI, como permitir que los Proveedores de Servicios de Comunicación (CSP) dispongan de una explicación completa sobre las acciones recomendadas por esta solución de IA. XAI puede identificar la raíz de los eventos que afectan al rendimiento de la red y a la experiencia del usuario final. Gracias a esto, los equipos de optimización dispondrán de una mejor visibilidad sobre los factores que generan los problemas, el impacto en el rendimiento de la red y las acciones más adecuadas.

Pronto, la XAI comenzará a desempeñar un papel fundamental en el ámbito de la ciberseguridad. La capacidad de explicar las decisiones de los modelos de IA en tiempo real permitirá a los profesionales de la seguridad informática

identificar rápidamente la raíz de los incidentes y tomar medidas proactivas para mitigarlos. Esto no solo mejorará la eficacia de las estrategias de defensa, sino que también aumentará la confianza en el uso de IA para la protección de infraestructuras críticas.

Los esfuerzos comentados, combinados con la puesta en práctica en empresas como el caso de Ericsson, contribuirán significativamente a desarrollar sistemas más precisos, transparentes y confiables, fortaleciendo así la capacidad de las organizaciones para protegerse contra amenazas cibernéticas.



# Referencias

1. Djenouri, Y., Belhadi, A., Srivastava, G. and Lin, J.C.W., 2023. When explainable AI meets IoT applications for supervised learning. *Cluster Computing*, 26(4), pp.2313-2323.
2. Kamil, S., Norul, H. S. A. S., Firdaus, A., & Usman, O. L. (2022). The rise of ransomware: A review of attacks, detection techniques, and future challenges. In *2022 International Conference on Business Analytics for Technology and Security (ICBATS)* (pp. 1-7). IEEE.
3. Ozkan-Okay, M., Samet, R., Aslan, Ö., & Gupta, D. (2021). A comprehensive systematic literature review on intrusion detection systems. *IEEE Access*, 9, 157727-157760.
4. Mishra, A., & Yadav, P. (2020). Anomaly-based IDS to detect attack using various artificial intelligence & machine learning algorithms: a review. In *2nd International Conference on Data, Engineering and Applications (IDEA)* (pp. 1-7). IEEE.
5. Saleh, H.M., Marouane, H. and Fakhfakh, A., 2024. Stochastic Gradient Descent Intrusions Detection for Wireless Sensor Network Attack Detection System Using Machine Learning. *IEEE Access*.
6. Karthik, M.G., Sivaji, U., Manohar, M., Jayaram, D., Gopalachari, M.V. and Vatambeti, R., 2024. An Intrusion Detection Model Based on Hybridization of S-ROA in Deep Learning Model for MANET. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, pp.1-12.
7. Liao, H.J., Lin, C.H.R., Lin, Y.C. and Tung, K.Y. (2013) 'Intrusion detection system: A comprehensive review', *Journal of Network and Computer Applications*, 36(1), pp. 16-24.
8. Abdulganiyu, O.H., Ait Tchakoucht, T. and Saheed, Y.K., 2023. A systematic literature review for network intrusion detection system (IDS). *International Journal of Information Security*, 22(5), pp.1125-1162.

9. Angra, S., & Ahuja, S. (2017). Machine learning and its applications: A review. In 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC) (pp. 57-60). IEEE.
10. DataScientest. Machine Learning: definición, funcionamiento y usos. Available at <https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos>.
11. Singh, A., Thakur, N. y Sharma, A. (2016). 'A review of supervised machine learning algorithms', en 2016 3rd international conference on computing for sustainable global development (INDIACom), IEEE, pp. 1310-1315.
12. Chitrlekha, G., & Roogi, J. M. (2021). A quick review of ML algorithms. In 2021 6th International Conference on Communication and Electronics Systems (ICCES) (pp. 1-5). IEEE.
13. Becker, D. and Cook, A., 2024. Use Cases for Model Insights. [online] Kaggle. Available at: <https://www.kaggle.com/code/dansbecker/use-cases-for-model-insights>
14. HRC (s.f.) Análisis de residuos Available at: [http://www.hrc.es/bioest/Reglin\\_16.html](http://www.hrc.es/bioest/Reglin_16.html)
15. Fedefliguer (s.f.) Observaciones influyentes, disponible en: <https://fedefliguer.github.io/AAI/influyente.html>
16. Wikipedia. (n.d.). Análisis de sensibilidad. Available at: [https://es.wikipedia.org/wiki/An%C3%A1lisis\\_de\\_sensibilidad](https://es.wikipedia.org/wiki/An%C3%A1lisis_de_sensibilidad)
17. Becker, D. (n.d.). Permutation Importance. Kaggle. Available at: <https://www.kaggle.com/code/dansbecker/permutation-importance>.
18. Edoardo Mosca, Ferenc Szigeti, Stella Tragianni, Daniel Gallagher, Georg Groh. SHAP-Based Explanation Methods: A Review for NLP Interpretability. Proceedings of the 29th International Conference on Computational Linguistics, 2022, pp. 4593-4603.
19. Visani, G., Bagli, E., Chesani, F., Poluzzi, A., & Capuzzo, D. (2020). Statistical stability indices for LIME: Obtaining reliable explanations for machine learning models.
20. IIC. Explicabilidad algorítmica: el método SHAP. Available at: <https://www.iic.uam.es/innovacion/explicabilidad-algoritmica-metodo-shap/>.
21. Openlayer Understanding LIME in 5 steps. Available at <https://www.openlayer.com/blog/post/understanding-lime-in-5-steps>
22. Ribeiro, M.T., Singh, S. and Guestrin, C. (2016) 'Why should I trust you? Explaining the predictions of any classifier', Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data

Mining, pp. 1135-1144.

23. Wang, M., Zheng, K., Yang, Y. and Wang, X., 2020. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8, pp.73127-73141.
24. Dias, T., Oliveira, N., Sousa, N., Praça, I. and Sousa, O., 2021, December. A hybrid approach for an interpretable and explainable intrusion detection system. In *International Conference on Intelligent Systems Design and Applications* (pp. 1035-1045). Cham: Springer International Publishing.
25. Jiang, R., Xue, Y. and Zou, D., 2023. Interpretability-aware industrial anomaly detection using autoencoders. *IEEE Access*.
26. Maricar, S.B.A., Anoop, A., Samuel, B.E., Appukuttan, A. and Alsinjlawi, K.H., 2024. An Improved Explainable Artificial Intelligence for Intrusion Detection System. *International Journal of Intelligent Systems and Applications in Engineering*, 12(14s), pp.108-115.
27. ShadyCyberSecutiry. 'CICIDS 2017 data and classifiers analysis', Kaggle. Available at: <https://www.kaggle.com/code/shadycybersecutiry/cicids-2017-data-and-classifiers-analysis-dc1ca5>
28. Huang, S., 2018. *KDD Cup 1999 Data: Computer network intrusion detection*. [online] Kaggle. Available at: <https://www.kaggle.com/datasets/galaxyh/kdd-cup-1999-data?select=kddcup.names>
29. Naveed, K., 2020. *N-BaloT Dataset to Detect IoT Botnet Attacks*. [online] Kaggle. Available at: <https://www.kaggle.com/datasets/mkashifn/nbaiot-dataset/data>
30. Solarmainframe, 2020. *IDS 2018 Intrusion CSVs (CSE-CIC-IDS2018)*. [online] Kaggle. Available at: <https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv/code>
31. Communications Security Establishment (CSE) & Canadian Institute for Cybersecurity (CIC), 2018. *CSE-CIC-IDS2018 on AWS*. [online] Available at: <https://www.unb.ca/cic/datasets/ids-2018.html>
32. Canadian Institute for Cybersecurity (CIC) & Canadian Internet Registration Authority (CIRA), 2020. *CIRA-CIC-DoHBrw-2020*. [online] Available at: <https://www.unb.ca/cic/datasets/dohbrw-2020.html>
33. Zaib, M.H., 2019. *NSL-KDD: Network Security, Information Security, Cyber Security*. [online] Kaggle. Available at: <https://www.kaggle.com/datasets/hassan06/nslkdd?select=KDDTrain%2B.txt>
34. Shuaib, J. (2023) 'Project IT302', Kaggle. Available at:



<https://www.kaggle.com/code/shuaibjawed/project-it302/notebook>

35. Scikit-learn. StandardScaler. Available at <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>
36. StudySmarter. Regresión Logística. Available at: <https://www.studysmarter.es/resumenes/matematicas/estadisticas/regresion-logistica/>.
37. JACAR. (s.f.). La función Sigmoide: una herramienta clave en redes neuronales. Available at <https://jacar.es/la-funcion-sigmoide-una-herramienta-clave-en-redes-neuronales/>
38. Zou, X., Hu, Y., Tian, Z., & Shen, K. (2019). Logistic regression model optimization and case analysis. En 2019 IEEE 7th international conference on computer science and network technology (ICCSNT) (pp. 135-139). IEEE.
39. Accelerated Analyst. How to use logistic regression for practical business problems. Medium. Available at: <https://medium.com/accelerated-analyst/how-to-use-logistic-regression-for-practical-business-problems-e4582826a802>.
40. Fernández Navarro, F., 2024. Modelos de la Computación (Aprendizaje supervisado), Redes Neuronales Multicapa I. Apuntes de clase, Universidad de Málaga. p. 9.
41. Wikipedia. Perceptrón. Available at: <https://es.wikipedia.org/wiki/Perceptr%C3%B3n>.
42. Popescu, M. C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8(7), 579-588.
43. Al-Mahasneh, A.J., Anavatti, S.G., and Garratt, M.A. (2017) 'The development of neural networks applications from perceptron to deep learning', 2017 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA), IEEE, pp. 1-6.
44. Wang, H. and Hu, D., 2005, October. Comparison of SVM and LS-SVM for regression. In 2005 International conference on neural networks and brain (Vol. 1, pp. 279-283). IEEE.
45. Abdullah, D.M. and Abdulazeez, A.M. (2021) 'Machine learning applications based on SVM classification: a review', Qubahan Academic Journal, 1(2), pp. 81-90.
46. Rigatti, S. J. (2017). Random forest. Journal of Insurance Medicine, 47(1), 31-39.

47. GeeksforGeeks. (2023). XGBoost Available at: <https://www.geeksforgeeks.org/xgboost/>.
48. Chen, T. & Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. August, pp. 785-794.
49. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... and Liu, T.Y. (2017) 'LightGBM: A highly efficient gradient boosting decision tree', Advances in Neural Information Processing Systems, 30
50. Ma, X., Sha, J., Wang, D., Yu, Y., Yang, Q. and Niu, X. (2018) 'Study on A Prediction of P2P Network Loan Default Based on the Machine Learning LightGBM and XGBoost Algorithms according to Different High Dimensional Data Cleaning', Electronic Commerce Research and Applications, 31, 24 – 39.
51. Min, W., Tang, Z., Zhu, M., Dai, Y., Wei, Y. and Zhang, R. (2018) 'Behavior Language Processing with Graph based Feature Generation for Fraud Detection in Online Lending', Proceedings of WSDM workshop on Misinformation and Misbehavior Mining on the Web, [Online], [http://snap.stanford.edu/mis2/files/MIS2\\_paper\\_26.pdf](http://snap.stanford.edu/mis2/files/MIS2_paper_26.pdf)
52. Minastireanu, E.A. & Mesnita, G., 2019. Light gbm machine learning algorithm to online click fraud detection. Journal of Information Assurance & Cybersecurity, vol. 2019, p. 263928.
53. Máxima Formación. (2019). ¿Cómo validar tu modelo de regresión?. Available at: <https://www.maximaformacion.es/blog-dat/como-validar-tu-modelo-de-regresion/>
54. LabelF. (n.d.). What is Accuracy, Precision, Recall and F1 Score? Available at: <https://www.labelf.ai/blog/what-is-accuracy-precision-recall-and-f1-score>.
55. Martínez Pérez, J.A. and Pérez Martín, P.S., 2023. La curva ROC. *SEMERGEN: revista española de medicina de familia*, e101821-e101821. Available at: <https://www.elsevier.es/index.php?p=revista&pRevista=pdf-simple&pii=S1138359322002817&r=40>
56. Razavi, S., Jakeman, A., Saltelli, A., Prieur, C., Iooss, B., Borgonovo, E., ... and Maier, H.R. (2021) 'The future of sensitivity analysis: an essential discipline for systems modeling and policy support', Environmental Modelling & Software, 137, p. 104954.
57. Python Software Foundation, 2023. Referencia del Lenguaje Python. Available at: <https://docs.python.org/es/3/reference/index.html>
58. Hunter, J. et al., 2023. Matplotlib: Visualization with Python. Matplotlib. Available at: <https://matplotlib.org/>
59. NumPy, 2023. NumPy: the fundamental package for scientific computing in

- Python. Available at : <https://numpy.org/>
60. Pandas, 2024. Pandas: powerful data analysis and manipulation tool for Python. Available at: <https://pandas.pydata.org/>
  61. Lundberg, S. M. and Lee, S. I., 2024. LIME: Local Interpretable Model-Agnostic Explanations. Available at: <https://pypi.org/project/lime/>
  62. Scikit-learn, 2024. Scikit-learn: Machine Learning in Python. Available at: <https://scikit-learn.org/stable/>
  63. Joblib, 2024. Joblib: running Python functions as pipeline jobs. Available at: <https://joblib.readthedocs.io/en/stable/>
  64. Seaborn, 2024. Seaborn: Statistical Data Visualization. Available at: <https://seaborn.pydata.org/>
  65. Chen, T. and Guestrin, C., 2024. XGBoost Python Package Introduction. Available at: [https://xgboost.readthedocs.io/en/stable/python/python\\_intro.html](https://xgboost.readthedocs.io/en/stable/python/python_intro.html)
  66. LightGBM, 2024. LightGBM Python Package Introduction. Available at: <https://lightgbm.readthedocs.io/en/latest/Python-Intro.html>
  67. Lundberg, S. M. and Lee, S. I., 2024. SHAP: SHapley Additive exPlanations. Available at: <https://shap.readthedocs.io/en/latest/>
  68. Microsoft, 2024. Visual Studio Code: Code Editing. Redefined. Available at: <https://code.visualstudio.com/>
  69. Microsoft, 2024. Working with Jupyter Notebooks in Visual Studio Code. Available at: <https://code.visualstudio.com/docs/datascience/jupyter-notebooks>
  70. IT Trends. (2024). La IA explicable (XAI) puede ayudar a las empresas a optimizar sus servicios de comunicaciones. Available at: <https://www.ittrends.es/software-y-apps/2024/02/la-ia-explicable-xai-puede-ayudar-a-las-empresas-a-optimizar-sus-servicios-de-comunicaciones>

# Apéndice A

# Manual de Instalación

## Requerimientos:

Este apéndice proporciona instrucciones detalladas para configurar y utilizar el entorno de desarrollo necesario para trabajar con el código del proyecto, asegurando así la replicación precisa de los resultados obtenidos.

### 1. Configuración del entorno de desarrollo

Instalación de Python: Descarga e instala Python 3.9 o una versión superior desde la página oficial de Python. Asegúrate de añadir Python al PATH durante la instalación.

Instalación de Visual Studio Code: Descarga e instala Visual Studio Code desde su sitio oficial.

### 2. Descarga del repositorio del proyecto

Obtén y descomprime la carpeta y sus archivos .zip del repositorio del proyecto proporcionado.

### 3. Configuración de Visual Studio Code

Instalación de extensiones en Visual Studio Code: Abre Visual Studio Code. Ve a la pestaña de extensiones (en la barra lateral)). En la barra de búsqueda, ingresa el nombre de la extensión: Python, Python Debugger, Pylance, Jupyter, Jupyter Cell Tags y Jupyter Notebook Renderers.

Haz clic en “Instalar” junto a cada extensión. Una vez instaladas todas las extensiones, reinicia Visual Studio Code para aplicar los cambios.

Configuración del entorno Python: Selecciona el intérprete de Python de tu entorno virtual en Visual Studio Code.

Ve a “View -> Command Palette”.

Escribe “Python: Select Interpreter”.

Selecciona la ruta a tu versión de Python.

#### 4. Instalación de bibliotecas necesarias

- Abrir el terminal o línea de comandos.
- Ejecutar el siguiente comando para instalar las bibliotecas necesarias utilizando pip:

```
pip install matplotlib numpy pandas lime joblib imbalanced-learn  
shap seaborn lightgbm
```

```
pip install scikit-learn==1.4.2
```

```
pip install xgboost==2.0.3
```

Una vez instalado jupyter notebook: *pip install --upgrade jupyter ipywidgets*

#### 5. Estructura del Directorio y Guía de Ejecución

La carpeta general del repositorio del proyecto se divide en dos carpetas, una para cada ataque:

- DDoS
- PortScan

Cada ataque a su vez se descompone en una carpeta dataset (donde estará el csv del dataset correspondiente a este ataque, debe estar en la siguiente ruta:

“dataset\MachineLearningCSV\_reducido\MachineLearningCVE”).

Y el archivo con el código de ese ataque, con extensión “ipynb”, el cuál será el que se ejecute para replicar los resultados, para ello se abre la

pestaña “File” -> “Open Folder” y abre una de las dos carpetas mencionadas DDOS o PortScan.

Además, si se quieren usar los mismos modelos, valores SHAP y explicadores, todos los archivos descomprimidos tienen que estar a la misma altura que el archivo con el código de ese ataque, y que no estén en subcarpetas, la única carpeta dentro de la carpeta del ataque es la de dataset, el resto de los archivos, en esta misma ruta.

## 6. Guía de Ejecución

Para ejecutar el código de manera efectiva y obtener resultados precisos, sigue estos pasos:

### Ejecución celda por celda

- Ejecuta las celdas en el orden en el que aparecen, especialmente las primeras, para asegurar que las librerías estén importadas y los modelos y datos cargados correctamente.
- La primera ejecución llevará más tiempo debido a la carga de modelos. Una vez finalizada, se generarán archivos con los modelos guardados en la misma carpeta. Esto permitirá que las ejecuciones posteriores carguen los modelos más rápidamente.

Cada celda contiene una introducción que explica qué se está haciendo y ejemplos de cómo llamar a cada función con cada uno de los modelos. Utiliza estos ejemplos como guía para entender el funcionamiento del código y adaptar los parámetros según tus necesidades.

La última celda es la celda de ejecución, que también incluye algunos ejemplos. Si deseas ejecutar cualquier función con cualquier modelo, en la celda donde se definió la función encontrarás un ejemplo de cómo hacerlo. Solo debes ajustar los parámetros variables a los valores deseados, siempre dentro de los rangos permitidos, como las instancias o características específicas.

Para obtener más información sobre los modelos utilizados en el análisis de resultados del proyecto o si tiene algún problema de rendimiento, visita la siguiente dirección de GitHub: <https://github.com/javierluquerueda/XAI-para-la-Deteccion-Confiable-de-Intrusiones-Explicabilidad-e-Interpretabilidad>. En esta dirección, encontrarás los modelos, datos, explicadores y valores SHAP cargados que se utilizaron en el análisis. Si tuviera alguna duda o necesitara información adicional, no dude en ponerse en contacto con el autor. Los datos de contacto se encuentran disponibles en el archivo README de esta misma dirección.