

UNIVERSIDAD TECNOLOGICA ECOTEC

FACULTAD:

INGENIERÍAS, ARQUITECTURA Y CIENCIAS DE LA NATURALEZA

TITULO:

DESARROLLO DE UN MODELO MACHINE LEARNING PARA LA DETECCIÓN DE ATAQUES DDOS EN ENTORNOS DE INTERNET DE LAS COSAS (IOT)

LINEA DE INVESTIGACIÓN:

TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN

MODALIDAD DE TITULACIÓN:

TRABAJO DE INTEGRACIÓN CURRICULAR

CARRERA:

INGENIERIA EN SOFTWARE

TITULO A OBTENER:

INGENIERO DE SOFTWARE

AUTORES:

FRANCESCA ALESSANDRA CALLE MERA

IVÁN ISAAC PAREDES CANO

TUTOR:

ING. MARCOS ANTONIO ESPINOZA MINA, PHD.

GUAYAQUIL 2023

ANEXO No. 9

PROCESO DE TITULACIÓN
CERTIFICADO DE APROBACIÓN DEL TUTOR

Samborondón, 22 de agosto de 2024

Magíster
Erika Ascencio
Unidad Académica: Ingenierías, Arquitectura y Ciencias de la Naturaleza
Universidad Tecnológica ECOTEC

De mis consideraciones:

Por medio de la presente comunico a usted que el trabajo de titulación TITULADO: DESARROLLO DE UN MODELO MACHINE LEARNING PARA LA DETECCIÓN DE ATAQUES DDOS EN ENTORNOS DE INTERNET DE LAS COSAS (IOT), fue revisado, siendo su contenido original en su totalidad, así como el cumplimiento de los requerimientos establecidos en la guía para su elaboración, por lo que se autoriza a los estudiantes: CALLE MERA FRANCESCA ALESSANDRA y PAREDES CANO IVÁN ISAAC, para que proceda con la presentación oral del mismo.

ATENTAMENTE,

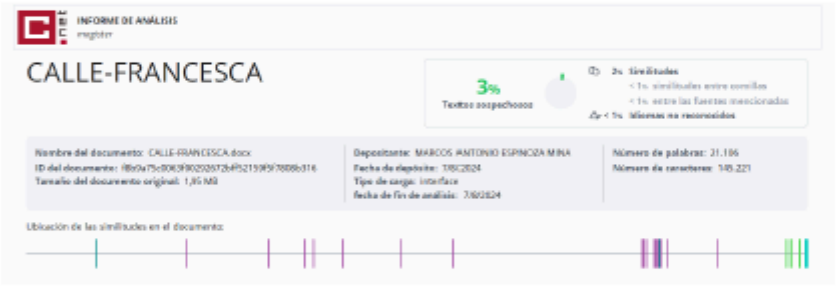


Firma
ING. MARCOS ANTONIO ESPINOZA MINA, PHD.
Tutor

ANEXO No. 10

PROCESO DE TITULACIÓN
CERTIFICADO DEL PORCENTAJE DE COINCIDENCIAS
DEL TRABAJO DE TITULACIÓN

Habiendo sido revisado el trabajo de titulación TITULADO: DESARROLLO DE UN MODELO MACHINE LEARNING PARA LA DETECCIÓN DE ATAQUES DDOS EN ENTORNOS DE INTERNET DE LAS COSAS (IOT) elaborado por CALLE MERA FRANCESCA ALESSANDRA y PAREDES CANO IVÁN ISAAC fue remitido al sistema de coincidencias en todo su contenido el mismo que presentó un porcentaje del 3% mismo que cumple con el valor aceptado para su presentación que es inferior o igual al 10% sobre el total de hojas del documento. Adicional se adjunta print de pantalla de dicho resultado.



ATENTAMENTE,

Firma
ING. MARCOS ANTONIO ESPINOZA MINA, PHD.
Tutor

Dedicatoria

Francesca Calle.

Dedico este trabajo de titulación con profunda gratitud a mis padres, Miguel Calle y Rocío Mera, quienes han sido los pilares fundamentales en mi vida y en cada paso de mi carrera universitaria. A mis hermanas Giomayra e Irene, por su apoyo incondicional y estar siempre a mi lado en cada situación.

También dedico este logro a mis amigos Iván, Axel, Marcelo y César, con quienes he compartido este proceso, por su disposición para ayudarme en todo momento y por el gran apoyo que me han brindado a lo largo de este camino.

Iván Paredes.

Quiero dedicar este trabajo y todos estos años de carrera a mi familia, quienes incondicionalmente siempre me apoyaron de distinta manera. Tanya, Iván y Paula, este es el inicio de muchas demostraciones de mi parte por todo el amor que les tengo.

También dedico este logro a mis amigos Francesca, Axel, Marcelo y César, con quienes he compartido este proceso, por su disposición para ayudarme en todo momento y por el gran apoyo que me han brindado a lo largo de este camino.

Agradecimientos

Francesca Calle.

Agradezco a Dios quien con su bendición llena siempre mi vida. A mi familia y amigos, quienes me dieron la fuerza para cumplir mi objetivo de culminar mi carrera, y estar siempre presentes animándome en cada fase de esta. Gracias a estas personas por extender su mano en mis momentos difíciles, impulsándome a cumplir uno de mis anhelos más deseados como la obtención de mi título universitario.

Iván Paredes.

Agradezco a mi hermana por su guía, a mis padres por compartirme su sabiduría y darme la oportunidad de demostrar el fruto de todo su esfuerzo y sacrificio. A todos mis amigos quienes me impartieron consejos. A cada uno los docentes y maestros que vieron potencial en mí y brindaron perspectivas de la carrea que ayudaron a que sea el profesional que soy ahora. Y, finalmente, a Rocco, mi mascota. Quien estuvo en cada mala noche y dispuesto a recibirme cada noche que regresaba de la universidad. Todos me ayudaron a cumplir este objetivo.

Índice

Resumen *10*

1. Introducción *12*

1.1. Contexto histórico del objeto de estudio **12**

1.2. Antecedentes **14**

1.3. Planteamiento del problema **16**

1.4. Objetivos **19**

1.4.1. Objetivo General 19

1.4.2. Objetivos Específicos 19

1.5. Justificación **19**

2. Revisión de la Literatura *21*

2.1. Marco Teórico Fundamental **21**

2.1.1. Definición de ataques DDoS 21

2.1.2. Impacto de los ataques DDoS 21

2.1.3. Ventajas de la detección de ataques DDoS 21

2.1.4. Definición de Machine Learning 22

2.1.5. Machine Learning y su Aplicación en la Seguridad IoT 22

2.1.6. Internet de las Cosas (IoT) 23

2.1.7. Evolución de IoT 23

2.2. Marco Teórico Conceptual **23**

2.2.1. Tipos de ataques DDOS y características 23

2.2.2. Protocolos de redes IoT 24

2.2.3. Machine Learning (ML) 25

2.2.3.1. Random Forest 25

2.2.3.2. K-Nearest Neighbors (KNN) 26

2.2.3.3. XGboost (Extreme Gradient Boosting) 26

2.2.4. Hiperparámetros de ML 26

2.2.5. Validación cruzada 26

2.2.6. Machine Learning como método de herramienta de seguridad 27

2.2.7. Dataset 27

2.2.8. Lenguaje R 28

2.3. Marco Teórico Situacional 28

2.3.1. Estado actual de la seguridad en IoT 28

2.3.2. Actores involucrados 28

2.3.3. Recursos y limitaciones 29

2.3.4. Desafíos en la detección de ataques DDoS en IoT 29

2.3.5. Diversidad y volumen de datos 29

2.3.6. Protección de la privacidad del usuario 29

2.3.7. Adaptabilidad a patrones de ataque en evolución 29

2.4. Marco Teórico Contextual 30

2.4.1. Impacto de los ataques DDoS en la sociedad y la economía 30

2.4.2. Interrupción de servicios esenciales 30

2.4.3. Pérdidas económicas 30

2.4.4. Daño a la reputación 30

2.4.5. Oportunidades para la mejora de la seguridad en IoT 30

2.4.6. Desarrollo de nuevas técnicas y enfoques 31

2.4.7. Mejora de la conciencia de seguridad 31

3. Metodología *32*

3.1. Enfoque de la investigación 32

3.2. Alcance de la investigación 33

3.3. Delimitación de la investigación 33

3.4. Métodos empleados 34

3.5. Procesamiento y análisis de la información 36

3.5.1. De la fuente de datos 36

3.5.2. Herramientas utilizadas para la analítica de datos 36

3.5.3. Importación y consolidación de los datos 38

3.5.4. Preprocesamiento de datos 40

3.5.5. Análisis exploratorio de datos 43

3.5.6. Ajuste de los modelos predictivos 46

3.6. Elementos metodológicos específicos para TI 51

4. Resultados *53*

4.1. De la consolidación de los datos 53

4.2. De la inspección general del dataset 53

4.3. Del preprocesamiento de datos 55

4.3.1. Del tratamiento de los valores ausentes 55

4.3.2. Del tratamiento de variables dicotómicas desbalanceadas 59

4.3.3. Del tratamiento a variables con 0 como único valor 59

4.3.4. Del tratamiento a variables con varianza cercana a 0 60

4.3.5. De la transformación de variables categóricas 61

4.3.6. Del análisis de correlación entre variables continuas 62

4.4. Del análisis exploratorio de datos **63**

4.4.1. Análisis descriptivo de la variable de respuesta 63

4.4.2. Análisis descriptivo de las variables predictoras 64

4.5. De la preparación del conjunto de datos para el ajuste de los modelos predictivos **68**

4.5.1. De la separación del conjunto de datos en datos de entrenamiento y prueba (Data splitting) 68

4.5.2. Del balanceo de la clase de respuesta 68

4.5.3. De la estandarización de variables continuas 69

4.6. Del ajuste de los modelos predictivos **70**

4.6.1. Del ajuste del modelo KNN 70

4.6.2. Del ajuste del modelo Random Forest 71

4.6.3. Del ajuste del modelo XGBoost 73

4.7. Evaluación de los modelos de machine learning **74**

4.7.1. Resultados de la evaluación del modelo Random Forest 75

4.7.2. Resultados de la evaluación del modelo XGBoost 77

4.8. Selección del modelo **78**

4.9. Evaluación del desempeño del producto/servicio de TI **81**

4.9.1. Evaluación con experto estadístico. 81

4.9.2. Calificación del modelo por el experto estadístico. 82

4.9.3. Evaluación con experto en redes y seguridades. 85

4.9.4. Calificación del modelo por el experto en redes y seguridades. 86

5. Conclusiones ***89***

6. Recomendaciones ***90***

7. Bibliografía ***92***

8. ANEXOS ***97***

Resumen

La expansión del Internet de las Cosas (IoT) ha incrementado la vulnerabilidad de las redes a ataques de denegación de servicio distribuido (DDoS), particularmente a través de botnets IoT, lo que representa una amenaza considerable para las redes empresariales. Estos ataques pueden causar sobrecarga de tráfico y exponer datos confidenciales. Esta investigación se enfoca en desarrollar un modelo de aprendizaje automático para detectar ataques DDoS en entornos IoT, ofreciendo una solución efectiva ante esta creciente amenaza. El estudio tiene como objetivo principal diseñar un modelo de clasificación utilizando algoritmos de Machine Learning para identificar estos ataques. Para ello, se realizaron revisiones de la literatura, análisis de datos y selección de algoritmos, seguidos de un entrenamiento y evaluación exhaustivos del modelo propuesto. La metodología empleada combina datos cuantitativos con entrevistas cualitativas a expertos, lo que enriquece la investigación y asegura la validez de los resultados. Los hallazgos revelan que el modelo Random Forest, correctamente ajustado, ofrece alta precisión en la detección de ataques, con un AUC de 0.866, superando a modelos como XGBoost. Este estudio no solo mejora la detección de DDoS en IoT, sino que también avanza la ciberseguridad al adaptar técnicas de Machine Learning a las particularidades de los dispositivos IoT, contribuyendo significativamente al campo de la seguridad informática.

Palabras claves: Internet de las Cosas, Ataques DDoS, Machine Learning, Random Forest.

Abstract

The expansion of the Internet of Things (IoT) has increased network vulnerability to distributed denial-of-service (DDoS) attacks, particularly through IoT botnets, representing a significant threat to enterprise networks. These attacks can lead to traffic overload and expose confidential data. This research focuses on developing a machine learning model to detect DDoS attacks in IoT environments, providing an effective solution to this growing threat. The main objective of the study is to design a classification model using machine learning algorithms to identify these attacks. The research involved literature reviews, data analysis, and algorithm selection, followed by extensive training and evaluation of the proposed model. The methodology combines quantitative data with qualitative expert interviews, enriching the research and ensuring result validity. Findings reveal that the finely tuned Random Forest model offers high accuracy in attack detection, with an AUC of 0.866, surpassing models like XGBoost. This study not only enhances DDoS detection in IoT but also advances cybersecurity by adapting machine learning techniques to the specific characteristics of IoT devices, significantly contributing to the field of information security.

Keywords: Internet of Things (IoT), DDoS Attacks, Machine Learning, Random Forest.

1. Introducción

1.1. Contexto histórico del objeto de estudio

El Internet de las cosas (IoT), se trata de una arquitectura emergente basada en la red global que agiliza el intercambio de productos y servicios entre las distintas cadenas de suministros, generando un impacto considerable en la seguridad y privacidad de los involucrados (Salazar & Silvestre, 2016). En la última década, este sistema ha desencadenado una transformación sin precedentes en la manera de interactuar con el mundo; fusionando el ámbito digital con el físico.

Esta revolución tecnológica ha propiciado la interconexión de dispositivos, sensores y sistemas en una amplia variedad de aplicaciones, que abarcan desde el monitoreo de la salud y la automatización del hogar hasta la gestión de ciudades inteligentes y la optimización de la cadena de suministros.

Con el avance imparable del ecosistema de la IoT, la seguridad emerge como una preocupación crítica en este entorno híbrido, puesto que requiere de una recopilación de datos personales que pueden ser usados incorrectamente. Uno de los retos más apremiantes en este contexto es la amenaza de los ataques de denegación de servicio distribuidos (DDoS), particularmente en las infraestructuras de la IoT.

Desarrollar un modelo de aprendizaje automático para detectar ataques DDoS en infraestructuras IoT, se presenta como un tema de vital relevancia para el control oportuno de las mismas. El crecimiento exponencial de la IoT en los últimos años ha conducido a la interconexión de miles de millones de dispositivos en todo el mundo haciendo posible una vulneración masiva de datos (Vargas, 2021).

Estos posibles ataques no solo tienen el potencial para interrumpir servicios esenciales como la atención médica o los sistemas de infraestructura urbana, también son sensibles ante la exposición de datos que pueden llegar a generar pérdidas económicas significativas dentro de entidades bancarias.

Por ello la seguridad de la IoT se convierte en una prioridad ineludible para preservar la privacidad de los usuarios y garantizar la continuidad de las operaciones empresariales. Sin embargo, a pesar de los avances en este campo, la detección y mitigación de ataques DDoS siguen representando un desafío sustancial para la comunidad de sistemas informáticos.

Las soluciones tradicionales, en la actualidad ya no cumplen con las características oportunas para el control del avance cibernético. Para los dispositivos IoT las restricciones de base a las que están sujetos, permiten fugas, debido al carácter heterogéneo de los nuevos dispositivos en el mercado. Se espera que el mercado de seguridad en IoT continúe creciendo significativamente debido al aumento en la adopción de dispositivos IoT y la creciente preocupación por la seguridad. Las empresas están invirtiendo más en soluciones de seguridad para proteger sus infraestructuras IoT.

El desarrollo de soluciones avanzadas de detección y respuesta a amenazas está en aumento, enfocándose en la identificación temprana de ataques DDoS y la mitigación antes de que causen daños significativos.

El incremento de la frecuencia y sofisticación de los ataques DDoS para la IoT ha generado una serie de necesidades esenciales en la comunidad científica. Como punto de partida, este problema abre paso a la importancia de una investigación exhaustiva de métodos avanzados de detección para ataques DDoS adaptables a las particularidades de cada dispositivo IoT. Esto implica el desarrollo y estudio de algoritmos de Machine Learning y técnicas de análisis de tráfico diseñados específicamente para este entorno.

El volumen de datos generado por un sistema de dispositivos inteligentes, comúnmente conocido como "Big Data", puede llegar a ser verdaderamente extenso. Por ello, la integración de esta gran cantidad de información en los sistemas existentes y la configuración del análisis para su aplicación efectiva presentan complejidades durante el proceso, lo que requiere una mayor inversión en tiempo de revisión (RedHat, 2023).

La creciente creación y adopción de dispositivos IoT amplió la superficie de ataques cibernéticos, exponiendo a organizaciones dentro de una situación de constante vulnerabilidad frente a los ataques informáticos, por medio de la filtración o acceso a la información de dispositivos inteligentes o redes sociales. En ocasiones, estos desperfectos han llegado a causar pérdidas multimillonarias para las partes afectadas (Greenberg, 2018). Debido a que el libre internet no establece requisitos de control de flujo adicionales a los hosts finales, constantemente surgen diversos ataques destinados a sobrecargar los sistemas finales. Los conocidos ataques de Denegación de Servicio (DoS) enlistan problemas como la pérdida de tiempo y dinero, además de afecciones para la reputación de una empresa.

Los ataques de Denegación de Servicio Distribuidos (DDoS), también pueden ser usados como distracción ante objetivos más fuertes, como el derribar servidores o la interrupción temporal de servicios específicos de necesidad básica. Estas técnicas aprovechan de manera efectiva los datos previamente disponibles (Peneti & Hemalatha, 2021).

Es por ello por lo que el presente documento de investigación pretende evaluar una solución estable y oportuna ante la creciente aparición de fugas de información sujetas a la libertad de la web. Como principal elemento resolutivo se expone un modelo de Machine Learning diseñado específicamente para la detección de ataques DDoS en entornos IoT.

Esta investigación multidisciplinaria aborda problemas comunes de importancia crítica y los resultados de la aplicación de modelos de Machine Learning en el fortalecimiento de la ciberseguridad en las empresas. Este modelo de detección fortalecerá la competitividad al garantizar la disponibilidad y seguridad de los servicios en línea, generando confianza entre los clientes y optimizando el uso de recursos seguros.

1.2. Antecedentes

El desarrollo de un modelo de Machine Learning para la detección de ataques DDoS (Distributed Denial of Service) en entornos IoT (Internet of Things) se enmarca en una

evolución tecnológica y de seguridad informática que ha sido particularmente dinámica en las últimas décadas.

Se ha identificado artículos relacionados a la detección de ataques DDoS con el uso de Machine Learning, entre estos el artículo elaborado por José Montes, Gustavo Icaza, y Néstor Duque en el año 2023 “Efecto de la selección de atributos en el desempeño de un IDS basado en machine learning para detección de intrusos en ataques DDoS”, en el cual se destaca la importancia de la selección de atributos en la mejora del desempeño de los sistemas de detección de intrusos basados en machine learning, especialmente en la detección de ataques DDoS, y proporciona evidencia empírica utilizando el dataset CICIDS-2017 (Montes, Icaza, & Duque, 2023).

Los datasets como CICIDS-2017 han proporcionado una base sólida para entrenar y evaluar modelos de detección de ataques DDoS. Estos suelen contener una variedad de escenarios y tipos de ataques, lo que ayuda a los investigadores a crear modelos más robustos. A menudo, los datasets disponibles no reflejan completamente la diversidad de dispositivos y configuraciones presentes en entornos IoT reales. La falta de datos representativos puede llevar a modelos que no generalizan bien a situaciones nuevas o no contempladas durante el entrenamiento.

En el año 2023, Omar Vargas expuso en su proyecto de tesis “Detección de Amenazas en Redes IoT Empleando Modelos Híbridos de Machine Learning y Redes Neuronales” la evolución e importancia de las tecnologías IoT, las crecientes amenazas y vulnerabilidades asociadas, y presentó una solución híbrida que combina técnicas de Machine Learning y Deep Learning para detectar y mitigar ataques de DoS de manera efectiva y rápida (Vargas O. , 2023).

Los modelos híbridos, como los discutidos por Vargas, combinan la capacidad de detección rápida de Machine Learning con la precisión de Deep Learning, ofreciendo una solución robusta contra amenazas complejas. Estos modelos pueden ser computacionalmente intensivos, lo que dificulta su implementación en dispositivos IoT con

recursos limitados. Además, la complejidad de los modelos de Deep Learning puede hacerlos difíciles de interpretar y ajustar.

El uso de Machine Learning en la seguridad informática ha ganado tracción en la última década. Los modelos de aprendizaje automático ofrecen capacidades avanzadas para la detección y mitigación de amenazas al identificar patrones y anomalías en los datos de red que podrían indicar un ataque. Esta tecnología ha sido adoptada por su capacidad para adaptarse y evolucionar frente a nuevas amenazas, algo crítico en un campo donde los atacantes constantemente desarrollan nuevas técnicas.

La relevancia de desarrollar modelos de Machine Learning específicos para la detección de ataques DDoS en entornos IoT radica en la necesidad urgente de proteger esta infraestructura crítica. Los ataques DDoS pueden tener consecuencias devastadoras, no solo en términos de costos económicos, sino también en la confianza y seguridad pública. La investigación en esta área está enfocada en crear modelos precisos y eficientes que puedan detectar y responder a estos ataques en tiempo real, minimizando el daño y asegurando la continuidad del servicio.

1.3. Planteamiento del problema

La creciente expansión de dispositivos en las redes IoT ha dado lugar a un entorno altamente cambiante y diverso, en el que las capacidades de procesamiento, protocolos de comunicación y demás patrones de tráfico informativo se presentan como una data expuesta o de posible violación.

En una investigación realizada por Kaspersky se reveló la existencia de una próspera economía en la web oscura, centrada en servicios vinculados al Internet de las cosas (IoT). Se ha identificado una fuerte demanda, especialmente entre los hackers, de ataques de denegación de servicio distribuido (DDoS) orquestados a través de botnets de IoT.

Durante la primera mitad del 2023, los expertos en Inteligencia de Digital Footprint de Kaspersky detectaron más de 700 anuncios que ofrecían servicios para llevar a cabo

ataques DDoS en diversos foros de la web oscura. El valor de estos servicios variaba según factores como el nivel de protección contra DDoS, la inclusión de CAPTCHA y la verificación de JavaScript por parte de la víctima, oscilando entre US$20 por día y US$10,000 por mes. En promedio, la investigación detectó que los anuncios ofertaban dichos servicios a un costo de US$63.5 por día o US$1,350 por mes (kaspersky, 2023).

Esta situación compleja crea el ecosistema perfecto para una extensa y variada superficie de ataques de ciberdelincuentes, que buscan explotar las debilidades propias de la diversidad de dispositivos conectados. Así, los ataques de denegación de servicio distribuido (DDoS) podrían representar una amenaza que obstruya el correcto funcionamiento de las redes IoT.

Los ataques DDoS se han considerado como una amenaza crítica para la comunidad empresarial, exponiéndolas a serios problemas como la sobrecarga de sus capacidades de gestión de tráfico, comúnmente presentes en sectores bancarios, minoristas, gaming, etc. De aquí pueden surgir los grandes robos a entidades bancarias con sala virtual, exposición de información personal de clientes, obtención de datos privados de dispositivos electrónicos, entre muchas otras consecuencias.

Es fundamental que las empresas comprendan la gravedad de esta amenaza y adopten medidas proactivas para salvaguardarse de estos ataques cibernéticos (BitLife, 2023).

Existen métodos y tecnologías para detectar ataques DDoS una vez que ocurren, pero la capacidad de predecirlos y evitarlos antes de que causen daño es un campo que aún necesita desarrollo.

La comunidad científica se encuentra en una encrucijada crítica ante la necesidad imperante de abordar los ataques de denegación de servicios distribuidos (DDoS) en las redes IoT. A medida que la adopción de dispositivos conectados continúa en aumento, se ha vuelto evidente que las soluciones actuales de detección de DDoS no son fácilmente extrapolables a entornos IoT, generando una brecha de seguridad que requiere una

atención inmediata y específica, evitando pérdidas multimillonarias para las partes afectada (Greenberg, 2018).

Los métodos convencionales de detección a menudo dependen de un análisis intensivo y rápido del tráfico, una tarea desafiante en entornos donde los recursos son escasos. Además, la variabilidad en los patrones de tráfico legítimo de dispositivos IoT dificulta la distinción entre actividades normales y ataques maliciosos (Aveleira, y otros, 2020).

Ante la problemática, algunas organizaciones vinculadas al desarrollo web han optado por la implementación de revisiones activas y permanentes en los procesos, pero ¿Cuáles son las estrategias y tecnologías más efectivas para la detección de ataques DDoS en el entorno IoT para mejorar la seguridad en redes de datos? El presente documento de investigación expone al sistema de Machine Learning como una opción viable para la solución progresiva de estas alteraciones en la seguridad de la información web.

La diversidad en los dispositivos IoT conlleva una amplia gama de capacidades de procesamiento, protocolos de comunicación y patrones de tráfico. Esta variabilidad complica la implementación de soluciones de seguridad que sean aplicables de manera uniforme a todos los dispositivos. Los modelos actuales de detección de DDoS pueden no ser lo suficientemente flexibles para adaptarse a las características únicas de cada tipo de dispositivo, lo que puede resultar en tasas de falsos positivos o falsos negativos elevadas.

La solución a este desafío se materializará mediante la creación de un modelo de detección de ataques DDoS, específicamente diseñado para su implementación en entornos IoT. Este será desarrollado y sometido a rigurosas pruebas con el fin de evaluar su eficacia en la prevención de ataques DDoS y su capacidad para mantener la disponibilidad y la integridad en el contexto de la IoT.

La investigación adoptará un enfoque multidisciplinario que integrará conocimientos en ciberseguridad, aprendizaje automático y tecnología IoT, con el propósito de abordar de manera exhaustiva este desafío crítico.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un modelo de clasificación de Machine Learning para la detección de ataques DDoS en entornos IOT a través del uso de algoritmos.

1.4.2. Objetivos Específicos

- Efectuar una revisión de la literatura científica y los informes técnicos relacionados con la seguridad de la información, con un enfoque particular en los ataques DDoS.
- Explorar y analizar las características del dataset para identificar patrones y correlaciones relevantes para la detección de ataques DDoS.
- Seleccionar y entrenar los algoritmos de Machine Learning más adecuados para la detección de ataques DDoS en entornos IoT.
- Evaluar la precisión del modelo, en la detección de ataques DDoS en entornos IoT.

1.5. Justificación

El presente proyecto está sujeto a la importancia de analizar, indagar y resolver las problemáticas de la sobre exposición informática en la nueva era del Internet de las cosas. Su relevancia radica en que los trabajos de investigación actuales, aunque valiosos, aún no han logrado llenar completamente esta brecha de conocimiento y habilidades. La evaluación limitada de los métodos propuestos en escenarios IoT específicos deja preguntas sin respuesta y plantea dudas sobre la aplicabilidad real de estas soluciones en entornos del mundo real.

Por ello, desde un enfoque práctico se analizará los sistemas para la detección de intrusos como parte de las soluciones de defensas para las redes IoT, los cuales se encargarán de identificar los ataques para implementar medidas de mitigación ante ellos, minimizando el costo de incurrencia de estos en tipos de requerimientos en la seguridad de la información como disponibilidad, confidencialidad e integridad. (Polanía, 2021).

Con la detección oportuna de ataques, se protege mejor la información personal de los usuarios, evitando la exposición y posible uso indebido de datos sensibles. Además, las empresas podrán gestionar de manera más efectiva sus recursos de seguridad al contar con un modelo de detección adaptado específicamente a las características de su infraestructura IoT, mejorando la eficiencia operativa.

El desarrollo de un modelo de detección de ataques DDoS específicamente diseñado para entornos IoT es un avance metodológico importante. Este modelo requerirá la adaptación de técnicas de aprendizaje automático existentes para abordar las peculiaridades de los entornos IoT, lo que permitirá la mejora de la seguridad y reducir tiempos de respuestas frente a amenazas cibernéticas.

Al abordar las limitaciones de los métodos actuales y proporcionar un enfoque adaptado a la realidad de los entornos IoT, la investigación contribuirá significativamente al cuerpo de conocimiento en ciberseguridad, proporcionando bases para futuros estudios y desarrollos en esta área.

Estos puntos destacan cómo el proyecto no solo abordará problemas específicos de seguridad en entornos IoT, sino que también avanzará en el campo de la ciberseguridad al proporcionar nuevas metodologías y soluciones prácticas para desafíos actuales.

2. Revisión de la literatura

2.1. Marco teórico fundamental

2.1.1. Definición de ataques DDoS

Un ataque DDoS, o ataque distribuido de denegación de servicio, es un tipo de ciberataque que intenta hacer que un sitio web o recurso de red no esté disponible colapsándolo con tráfico malintencionado para que no pueda funcionar correctamente (Akamai, 2023).

Los ataques de red distribuidos, comúnmente llamados ataques de denegación de servicio distribuida (DDoS), explotan las limitaciones de capacidad inherentes a cualquier recurso de red, como la infraestructura que respalda el sitio web de una empresa. En un ataque DDoS, se envían múltiples solicitudes al recurso web atacado con la intención de sobrepasar la capacidad del sitio web para manejar estas solicitudes, provocando así su mal funcionamiento. Algunos de los objetivos más frecuentes de los ataques DDoS incluyen: sitios de compras en línea, casinos en línea, empresas u organizaciones que se base en la provisión de servicios en línea.

Estos ataques buscan afectar la disponibilidad y el rendimiento de los servicios en línea, creando una congestión en los recursos de red y obstaculizando la operación normal del sitio web o servicio afectado (Kaspersky, 2023).

2.1.2. Impacto de los ataques DDoS

Los ataques DDoS pueden tener un impacto significativo en las organizaciones, causando interrupciones del servicio, pérdida de ingresos y daño a la reputación. En el contexto de IoT, estos ataques también pueden poner en riesgo la seguridad y la privacidad de los usuarios (Akamai, 2023).

2.1.3. Ventajas de la detección de ataques DDoS

La detección temprana y el monitoreo del tráfico son elementos esenciales en la prevención de ataques DDoS. Por lo tanto, es imperativo gestionar, supervisar y filtrar el tráfico en las páginas web. Esto requiere una inspección minuciosa de los paquetes de

datos y solicitudes maliciosas que pueden ser enviados al dominio, utilizando programas para realizar escaneo inteligente y firewalls. Este enfoque proactivo fortalece las defensas de tu sistema al identificar y neutralizar posibles amenazas antes de que puedan afectar la disponibilidad y el rendimiento de tu sitio web (EASYDMARC, 2022).

2.1.4. Definición de machine learning

De acuerdo con lo descrito por IBM en el 2021, machine learning constituye una disciplina dentro de la inteligencia artificial (IA) y la informática, focalizándose en emplear datos y algoritmos para replicar el proceso de aprendizaje observado en los seres humanos, logrando una mejora progresiva en su precisión (IBM, 2021).

2.1.5. Machine learning y su aplicación en la seguridad IoT

El campo del machine learning abarca diversos tipos de modelos de aprendizaje automático y emplea una variedad de técnicas algorítmicas. La elección entre aprendizaje supervisado, no supervisado, semisupervisado o por refuerzo depende de la naturaleza de los datos y del objetivo que se desea alcanzar. Dentro de cada uno de estos modelos, se pueden aplicar una o varias técnicas algorítmicas, adaptadas a los conjuntos de datos utilizados y los objetivos perseguidos. Los algoritmos de machine learning están fundamentalmente diseñados para clasificar elementos, identificar patrones, anticipar resultados y tomar decisiones informadas. Estos algoritmos pueden emplearse de manera individual o combinarse estratégicamente para lograr la máxima precisión, especialmente cuando se trata de datos complejos y poco predecibles (Recuerdo, 2021).

Aprendizaje No Supervisado. – Este tipo de aprendizaje, también conocido como machine learning no supervisado, utiliza algoritmos para explorar y agrupar conjuntos de datos sin etiquetas en clústeres. Estos algoritmos descubren agrupaciones o patrones subyacentes en los datos sin requerir intervención humana. Esta capacidad para identificar las similitudes y aquellas diferencias en la información lo permite contar con una herramienta que brinda efectividad para el análisis exploratorio de datos, las estrategias de venta cruzada, segmentación de clientes y reconocimiento de imágenes.

Aprendizaje Semisupervisado. - Une el aprendizaje supervisado y las técnicas de aprendizaje no supervisado para resolver sus desafíos clave. Con él, se entrena un modelo inicial en unas pocas muestras etiquetadas y, a continuación, se aplica de forma iterativa al mayor número de datos sin etiquetar.

Aprendizaje de Refuerzo. - Es una técnica de machine learning (ML) que capacita al software para tomar decisiones con el objetivo de obtener resultados óptimos. Esta técnica imita el proceso de aprendizaje por ensayo y error utilizado por los humanos para alcanzar sus metas. Las acciones del software que contribuyen al logro de su objetivo se refuerzan, mientras que aquellas que se desvían del objetivo se ignoran.

2.1.6. Internet de las cosas (IoT)

El Internet de las Cosas (IoT) es el mecanismo que facilita la conexión de objetos físicos cotidianos a la red de Internet. Esto abarca desde elementos comunes del hogar, como las bombillas, hasta dispositivos médicos utilizados en la atención sanitaria, pasando por ropa y accesorios personales inteligentes, e incluso sistemas integrados en ciudades inteligentes.

Los dispositivos IoT que se hallan en estos objetos físicos generalmente se clasifican en dos categorías: son interruptores (que transmiten instrucciones a un objeto) o son sensores (que recogen datos y los envían a otro lugar) (Red Hat, 2023).

2.1.7. Evolución de IoT

La rápida evolución de Internet ha hecho que el Internet de las Cosas (IoT) pase de ser una mera proyección futurista a una realidad tangible. La popularidad de esta tecnología se debe principalmente a la amplia gama de aplicaciones y oportunidades que ofrece, tanto para enriquecer la vida diaria de las personas como para optimizar los entornos empresariales, donde ya se ha estado implementando desde hace un tiempo (García, 2023).

2.2. Marco teórico conceptual

2.2.1. Tipos de ataques DDoS y características

Los ataques DDoS se dividen en tres categorías principales: ataques volumétricos, ataques de protocolo y ataques a la capa de aplicación. Entre los ejemplos más frecuentes de ataques DDoS se encuentran la amplificación de DNS, la inundación SYN y la inundación UDP (Stackscale, 2022).

Ataques Volumétricos. - Los ataques volumétricos tienen la intención de sobrecargar el ancho de banda del recurso o servicio objetivo y son comúnmente denominados "ataques basados en volumen". Al emplear una considerable cantidad de tráfico a través de una red de robots o botnet, los perpetradores buscan retardar o bloquear el flujo normal de tráfico de los usuarios legítimos.

Ataques de Protocolo. - Los ataques de protocolo se centran en consumir y agotar los recursos efectivos de servidores y equipos intermedios, como los firewalls, y son comúnmente referidos como "ataques basados en protocolo". Para llevar a cabo este tipo de ataques, los agresores emplean solicitudes de conexión maliciosas con el fin de comprometer las comunicaciones del protocolo.

Ataques a la capa de Aplicación. - Los ataques dirigidos a la capa de aplicación buscan inducir la caída de un servidor web mediante solicitudes que aparentan ser legítimas. Este tipo de ataque, conocido también como "ataques DDoS de capa 7", se caracteriza por ser relativamente sencillo de ejecutar, complicado de detener o ralentizar, y tiende a estar enfocado en aplicaciones específicas.

2.2.2. Protocolos de redes IoT

De acuerdo con lo descrito por Rob Faludi (2021), los dispositivos IoT interactúan de diversas maneras, empleando una amplia variedad de protocolos. La comunicación de estos dispositivos varía según sus características, su ubicación, los dispositivos y sistemas con los que deben conectar y el tipo de información que necesitan transmitir. No hay un protocolo universal que sirva como un "lenguaje" común para la transmisión de mensajes

entre dispositivos IoT. La selección del protocolo adecuado siempre debe basarse en los requisitos particulares de la aplicación en cuestión (Faludi, 2021).

2.2.2.1. Uso de protocolos TCP y UDP en redes IoT.

De acuerdo con lo señalado por Microsoft, el protocolo TCP es ampliamente utilizado en la mayoría de las conexiones a Internet. Este protocolo gestiona la comunicación entre dispositivos dividiendo grandes volúmenes de datos en paquetes más pequeños, que luego se envían y ensamblan nuevamente para asegurar una transmisión eficiente y precisa. Por otro lado, UDP es un protocolo de comunicación que opera sobre IP, permitiendo la interacción entre procesos. A diferencia de TCP, UDP optimiza la velocidad de transferencia de datos y es especialmente útil para aplicaciones que requieren la transmisión de datos.

2.2.3. Machine learning (ML)

El aprendizaje automático es la disciplina que se ocupa del desarrollo de algoritmos y modelos estadísticos que los sistemas informáticos utilizan para realizar tareas sin recibir instrucciones explícitas, sino más bien basándose en patrones e inferencias. Los sistemas informáticos utilizan algoritmos de aprendizaje automático para procesar grandes volúmenes de datos históricos y detectar patrones en ellos. Esto les permite producir resultados más precisos a partir de un conjunto de datos. Por ejemplo, los científicos de datos pueden entrenar una aplicación médica para diagnosticar el cáncer utilizando imágenes de rayos X provenientes de un vasto almacén de millones de imágenes escaneadas y sus diagnósticos correspondientes (AWS, 2023).

Existen varios modelos de Machine Learning que pueden ser utilizados para la detección de ataques DDoS. Algunos de los más comunes incluyen:

2.2.3.1. Random forest

Este algoritmo de aprendizaje automático popularizado por Leo Breiman y Adele Cutler, integra las respuestas de múltiples árboles de decisión para obtener una solución

singular. Su sencillez y versatilidad han fomentado su uso generalizado, ya que es capaz de abordar tanto problemas de clasificación como de regresión (IBM, 2023).

2.2.3.2. K-Nearest neighbors (KNN)

Es un clasificador no paramétrico de aprendizaje supervisado que se basa en la cercanía para realizar clasificaciones o predicciones sobre la categorización de un punto de datos específico. Aunque puede aplicarse a problemas de regresión o clasificación, se utiliza comúnmente como un algoritmo de clasificación, bajo la premisa de que los puntos similares tienden a encontrarse en proximidad uno del otro (IBM, 2023).

2.2.3.3. XGboost (Extreme Gradient Boosting)

Es una implementación de árboles de decisión potenciados por gradiente diseñados para la velocidad y el rendimiento creadas por Tianqi Chen, ahora con contribuciones de muchos desarrolladores. Pertenece a una colección más amplia de herramientas bajo el paraguas de la Comunidad de Aprendizaje Automático Distribuido o DMLC, que también son los creadores de la popular biblioteca de aprendizaje profundo mxnet (Brownlee, 2021).

2.2.4. Hiperparámetros de ML

Los hiperparámetros son variables externas de configuración que los científicos de datos emplean para controlar el proceso de entrenamiento de modelos de machine learning. Conocidos también como hiperparámetros de modelos, se configuran manualmente antes de entrenar un modelo. Estos difieren de los parámetros, que son elementos internos derivados automáticamente durante el proceso de aprendizaje y no son configurados por los científicos de datos (AWS, 2024).

El uso de hiperparámetros de los modelos Random Forest, KNN, y XGBoost se lleva a cabo mediante la validación cruzada para encontrar los valores que proporcionen el mejor rendimiento los modelos modelo.

2.2.5. Validación cruzada

La validación cruzada es una técnica utilizada para evaluar modelos de machine learning entrenándolos en diferentes subconjuntos de los datos disponibles y luego

evaluando su rendimiento en los subconjuntos restantes. Este método ayuda a identificar y mitigar el sobreajuste al asegurar que el modelo generalice bien a datos no vistos, que ocurre cuando un modelo no logra generalizar un patrón a nuevos datos (AWS, 2016).

2.2.6. Machine learning como método de herramienta de seguridad

Las soluciones de seguridad para el machine learning difieren de la percepción común de la inteligencia artificial, pero son actualmente las herramientas de ciberseguridad más robustas disponibles. En el ámbito de esta tecnología, se emplean patrones de datos para evaluar la probabilidad de que ocurra un evento específico.

El machine learning contrasta en ciertos aspectos con la inteligencia artificial genuina. En particular, el enfoque del machine learning está más orientado hacia la "precisión" que hacia el "éxito". Este enfoque implica aprender de un conjunto de datos específicamente diseñado para tareas particulares, buscando lograr el rendimiento óptimo en una tarea específica, incluso si la solución no es la ideal según los datos. Aunque el machine learning no proporciona una interpretación genuina de los datos, esa responsabilidad aún recae en los equipos humanos.

El machine learning destaca especialmente en la ejecución de tareas monótonas, como la identificación y adaptación de patrones de datos. Estas tareas suelen generar fatiga en los humanos y una baja capacidad para manejar la monotonía. En consecuencia, aunque la interpretación de los análisis de datos sigue siendo responsabilidad humana, el machine learning puede facilitar la presentación de los datos de manera comprensible y lista para su análisis (Kaspersky, 2023).

2.2.7. Dataset

Un Dataset es, como su nombre indica, un conjunto de datos organizado según un sistema de almacenamiento que proporciona las pautas principales para la búsqueda o el directorio de la información que se desea utilizar (Cáceres, 2023).

En esencia, se trata del contenido de una tabla en una base de datos, la cual tiene varias columnas en las que se registran datos en cada fila. Estas filas pueden considerarse como las categorías de los datos, mientras que las columnas representan las posibles variables que las componen. La combinación de columnas y filas es lo que se denomina Dataset.

2.2.8. Lenguaje R

R es un lenguaje de programación y una plataforma de desarrollo que se usa principalmente para analizar grandes volúmenes de datos y hacer cálculos estadísticos. Es un programa gratuito y de código abierto que incluye muchas herramientas y bibliotecas útiles para manejar datos y crear gráficos (Worsley, 2024).

Lo que hace especial a R es su capacidad para trabajar con grandes cantidades de datos, su flexibilidad para realizar diferentes tipos de análisis, y la gran comunidad de usuarios y desarrolladores que siempre está compartiendo conocimientos y mejorando el software.

2.3. Marco teórico situacional

2.3.1. Estado actual de la seguridad en IoT

A medida que la adopción de dispositivos IoT aumenta, también lo hace la superficie de ataque para los ciberdelincuentes. Los ataques a estos dispositivos pueden tener consecuencias graves, como la interrupción de servicios esenciales y la exposición de datos sensibles.

La seguridad en IoT es un área de creciente preocupación debido al aumento de la adopción de dispositivos IoT. Los ataques a estos dispositivos pueden tener consecuencias graves, como la interrupción de servicios esenciales y la exposición de datos sensibles.

2.3.2. Actores involucrados

Los actores involucrados en la seguridad de IoT incluyen a los usuarios de dispositivos IoT, los proveedores de servicios de IoT, los administradores de red y los ciberdelincuentes que llevan a cabo los ataques DDoS. Cada uno de estos actores tiene un

papel importante en la seguridad de IoT y puede ser un objetivo potencial para los ataques DDoS.

2.3.3. Recursos y limitaciones

Los recursos disponibles para abordar la seguridad en IoT incluyen las tecnologías de seguridad existentes, así como los avances en Machine Learning que pueden ser aplicados para la detección de ataques DDoS. Sin embargo, también existen limitaciones, como la gran cantidad de datos generados por los dispositivos IoT y la necesidad de proteger la privacidad del usuario.

2.3.4. Desafíos en la detección de ataques DDoS en IoT

La detección de ataques DDoS en entornos IoT presenta varios desafíos. Estos incluyen la diversidad y la gran cantidad de datos generados por los dispositivos IoT, la necesidad de proteger la privacidad del usuario y la necesidad de desarrollar modelos que puedan adaptarse a patrones de ataque en constante evolución.

2.3.5. Diversidad y volumen de datos

Uno de los principales desafíos en la detección de ataques DDoS en IoT es la diversidad y el volumen de los datos generados por los dispositivos IoT. Estos datos pueden variar en términos de formato, tamaño y frecuencia, lo que puede dificultar la detección de patrones anómalos.

2.3.6. Protección de la privacidad del usuario

Otro desafío importante es la necesidad de proteger la privacidad del usuario. Los dispositivos IoT a menudo recopilan y transmiten datos sensibles, lo que puede ser un objetivo atractivo para los ciberdelincuentes. Por lo tanto, es crucial desarrollar métodos de detección de ataques que sean efectivos sin comprometer la privacidad del usuario.

2.3.7. Adaptabilidad a patrones de ataque en evolución

Los patrones de ataque DDoS están en constante evolución, con los ciberdelincuentes desarrollando continuamente nuevas tácticas y técnicas. Esto presenta un

desafío para la detección de ataques, ya que los modelos deben ser capaces de adaptarse a estos patrones de ataque en constante cambio.

2.4. Marco teórico contextual

2.4.1. Impacto de los ataques DDoS en la sociedad y la economía

Los ataques DoS y DDoS inundan servidores con solicitudes falsas, sobrepasando su capacidad y dejándolos inutilizables. Los DDoS son más efectivos, ya que se realizan desde múltiples fuentes. Suelen usar botnets, redes de dispositivos infectados, para enviar las solicitudes. Estos ataques pueden tener graves consecuencias económicas y comprometer la disponibilidad de la información. En el contexto de IoT, estos ataques también pueden poner en riesgo la seguridad y la privacidad de los usuarios (WeLiveSecurity, 2024).

2.4.2. Interrupción de servicios esenciales

Los ataques DDoS pueden interrumpir servicios esenciales, como la atención médica, el transporte y la energía. Esto puede tener un impacto significativo en la sociedad, ya que estos servicios son fundamentales para el funcionamiento de nuestras comunidades.

2.4.3. Pérdidas económicas

Las pérdidas económicas causadas por los ataques DDoS pueden ser significativas. Estas pérdidas pueden ser directas, como la pérdida de ingresos debido a la interrupción del servicio, o indirectas, como el costo de la recuperación después de un ataque.

2.4.4. Daño a la reputación

El daño a la reputación de una organización puede ser uno de los impactos más duraderos de un ataque DDoS. Este daño puede resultar en la pérdida de confianza del cliente y puede tener un impacto a largo plazo en el éxito de la organización.

2.4.5. Oportunidades para la mejora de la seguridad en IoT

A pesar de los desafíos, también hay oportunidades significativas para mejorar la seguridad en IoT. Estas incluyen el desarrollo de nuevas técnicas y enfoques para la

detección de ataques, la mejora de la conciencia de seguridad entre los usuarios de IoT y la promoción de mejores prácticas de seguridad en la industria de IoT.

2.4.6. Desarrollo de nuevas técnicas y enfoques

Una de las principales oportunidades para mejorar la seguridad en IoT es el desarrollo de nuevas técnicas y enfoques para la detección de ataques. Esto incluye la aplicación de avances en Machine Learning y otras tecnologías emergentes.

2.4.7. Mejora de la conciencia de seguridad

Otra oportunidad importante es la mejora de la conciencia de seguridad entre los usuarios de IoT. Esto puede lograrse a través de la educación y la formación, así como a través de la promoción de mejores prácticas de seguridad.

3. Metodología

3.1. Enfoque de la investigación

A pesar de los desafíos, también hay oportunidades significativas para mejorar la seguridad en IoT. Estas incluyen el desarrollo de nuevas técnicas y enfoques para la detección de ataques, la mejora de la conciencia de seguridad entre los usuarios de IoT y la promoción de mejores prácticas de seguridad en la industria de IoT.

El enfoque metodológico de la investigación consiste en un enfoque mixto, el cual ha sido seleccionado con el objeto de complementar, con un enfoque cualitativo evidenciado en las entrevistas a expertos, un enfoque cuantitativo basado en el entrenamiento de modelos de machine learning, incluyendo todas las etapas previas de análisis de datos que esta conlleva.

Este enfoque ha sido seleccionado pues permite enriquecer la investigación al aunar los enfoques antes mencionados, pues los criterios de los expertos aportan a obtener una perspectiva basada en la experiencia y una opinión profesional sobre el desarrollo de los modelos de machine learning. De esta manera, se configura un enfoque que expone lo mejor de la expertise de profesionales y la aplicación directa de la ciencia de datos en la rama de ciberseguridad con el objeto de producir un proyecto robusto y factible de implementación.

De hecho, Llugsa (2022), en su tesis denominada “Prototipo basado en internet de las cosas (IoT) para el monitoreo de invernaderos”, señala que el enfoque de su investigación es mixto pues su estudio emplea tanto métodos cuantitativos al crear un prototipo para medir humedad y temperatura con IoT inalámbrico como métodos cualitativos al obtener datos de entrevistas con ingenieros agrónomos y agricultores, que indican mejoras organizacionales gracias al IoT.

Por ello, se puede concluir que el enfoque de la investigación de Llugsa coincide con el enfoque de la presente investigación al ser mixto, combinando los enfoques cuantitativos y cualitativos.

3.2. Alcance de la investigación

El alcance de la investigación es exploratorio pues se ejecutan etapas de investigación que incluyen la construcción de un marco teórico que permite comprender el estado del arte de una problemática de ciberseguridad, como lo constituyen los ataques DDoS. Otra etapa que refuerza el sentido exploratorio de la investigación es la aplicación de aprendizaje automático para la optimización y entrenamiento de distintos modelos de machine learning orientados a predecir la ocurrencia del problema en cuestión, los ataques DDoS.

También, cabe mencionar que, antes de llevar a cabo el entrenamiento de los modelos ML, se efectuó una exhaustiva investigación sobre los diferentes recursos web que permitieran obtener el dataset apropiado para ajustarlos. Por tanto, esta etapa previa permitió explorar y abordar la problemática desde el análisis de diferentes datasets para seleccionar el más conveniente, lo cual refuerza el enfoque exploratorio de la investigación.

3.3. Delimitación de la investigación

En el enfoque cuantitativo de la investigación, se tiene como población a un dataset de 88 variables con 20,364,525 registros, el cual fue seleccionado de entre otros dos conjuntos de datos obtenidos de diferentes fuentes. El criterio de selección se basa en las siguientes variables apropiadas que contiene, las cuales permiten ajustar, de mejor manera, los modelos de ML al favorecer la segmentación de los tipos de ataques (LDAP, MSSQL, NetBIOS, entre otros) por grupos de registros aportando a la facilidad del modelo para efectuar la clasificación con nuevos datos:

Sin embargo, los otros datasets, a pesar de contener información relevante, no permitían dicha segmentación, lo cual dificultaría el proceso de entrenamiento en cuanto a clasificación. Además, los datos de estos, al provenir de recolección de data en función a diferentes protocolos, no podrían ser anexados para su análisis y construcción de los modelos.

No obstante, en el enfoque cualitativo se considerarán los criterios de dos expertos para concluir sobre los resultados de los modelos ML, determinar los aspectos característicos de TI para la implementación de estos y recomendar sobre el presente trabajo a los lectores y/o futuros investigadores.

El periodo de la investigación inicia con la delimitación del objetivo de la investigación en enero de 2024, y culmina con la redacción de las conclusiones y recomendaciones del presente trabajo en agosto del mismo año.

3.4. Métodos empleados

El método empleado corresponde a una adaptación propia de Kanban, pues, tal como lo indica el anexo 5.1. del presente proyecto, se ha diseñado un board que permite clasificar las actividades en cuatro grupos: pendiente, en proceso, en revisión y completado.

De hecho, la aplicación coincide con Anderson, Sienkiewicz, Brezočnik, Majer, Alqudah, Razali, Morales & Subra, quienes señalan que, en la implementación de Kanban para gestionar procesos de TI, es crucial: (i) hacer visible el flujo de trabajo usando un tablero Kanban, (ii) medir y optimizar el flujo como otra práctica esencial, (iii) clarificar las políticas de gestión del flujo de trabajo y (iv) mantener una retroalimentación constante y un control continuo (Fuentes & Sebastían, 2022, pág. 1663)

También, como evidencia de aplicación del modelo, se ha medido y optimizado el flujo de manera que, semanalmente, se ha calculado un porcentaje de avance obtenido entre el número de tareas completadas sobre el número total de estas. Ejemplo, en la quinta semana, se tuvo un total de 7 tareas completadas de 12, por ello, en aquella instancia, el porcentaje de avance fue del 58.33%.

Así mismo, se establecieron las siguientes políticas para la gestión del flujo de trabajo:

1) Planificación y definición de objetivos.

Establecer objetivos claros y plazos para cada fase del proyecto, incluyendo la recopilación de datos, el desarrollo y evaluación de modelos KNN, Random Forest y XGBoost, y la redacción del trabajo. Programar entrevistas a expertos en momentos clave para integrar sus perspectivas.

2) Gestión de datos y preprocesamiento.

Aplicar técnicas rigurosas para la recolección, limpieza y preprocesamiento de datos. Definir claramente cómo manejar valores faltantes y qué métodos de normalización usar. Documentar todos los procedimientos para asegurar transparencia.

3) Documentación y escritura del trabajo.

Estructurar el trabajo en capítulos: Introducción, Marco Teórico, Metodología, Resultados, Conclusiones y Recomendaciones. Cada sección debe abordar: el contexto y objetivos, la revisión de literatura, el desarrollo de modelos y entrevistas, los hallazgos y sus implicaciones.

4) Validación y evaluación de modelos de ML.

Utilizar validación cruzada y métricas de desempeño como precisión, recall y F1-score para evaluar los modelos RF y XGBoost. Documentar los resultados detalladamente para facilitar el análisis.

5) Retroalimentación y colaboración.

Mantener comunicación constante con asesores y expertos. Programar reuniones regulares para recibir retroalimentación y ajustar el proyecto. Integrar las recomendaciones obtenidas en las entrevistas y registrar todos los cambios.

En definitiva, la adaptación de Kanban y las políticas establecidas han permitido una gestión efectiva del flujo de trabajo, con una planificación clara, un manejo riguroso de

datos, y una evaluación continua, garantizando un avance sistemático y organizado del proyecto.

3.5. Procesamiento y análisis de la información

3.5.1. De la fuente de datos

Los datos utilizados en esta investigación provienen del conjunto de datos CICDDoS2019, que contiene información detallada y actualizada sobre los ataques de Denegación de Servicio Distribuido (DDoS) más comunes, así como datos de tráfico benigno. Este conjunto de datos es considerado una referencia clave en la investigación de ataques DDoS debido a su similitud con los datos reales de redes del mundo real. Incluye tanto los archivos de captura de paquetes (PCAP) como los resultados del análisis del tráfico de red, que han sido procesados mediante la herramienta CICFlowMeter-V3. Los flujos de datos están etiquetados con detalles específicos como la marca de tiempo, las direcciones IP de origen y destino, los puertos de origen y destino, los protocolos utilizados y la naturaleza del tráfico, ya sea benigno o relacionado con ataques DDoS, presentados en formato CSV.

3.5.2. Herramientas utilizadas para la analítica de datos

3.5.2.1. Lenguaje de programación

Para llevar a cabo el análisis de datos y el ajuste de modelos predictivos, se utilizó el lenguaje de programación R en su versión 4.3.2. El trabajo se realizó en el entorno de desarrollo integrado de RStudio, que proporciona una interfaz amigable, con herramientas que hacen que el desarrollo en R sea más eficiente y accesible, en especial para la analítica de datos y tareas estadísticas.

3.5.2.2. Librerías utilizadas

Readr

La librería readr, parte del ecosistema tidyverse, se emplea para la importación eficiente de datos en formato de texto, como archivos CSV y TSV. Su diseño optimizado

permite una lectura rápida y precisa de grandes volúmenes de datos, facilitando la carga y el manejo eficiente de la información para su posterior análisis.

Tidyverse

Tidyverse es un conjunto integral de paquetes en R que proporciona herramientas para la manipulación, visualización y análisis de datos. Incluye paquetes como dplyr para la transformación de dataframes, ggplot2 para la creación de gráficos, y tidyr para el ordenamiento y limpieza de datos. Este conjunto de herramientas facilita un flujo de trabajo coherente y eficaz en el análisis de datos.

Caret

La librería caret (Classification and Regression Training) ofrece una amplia gama de funciones para el preprocesamiento de datos, selección de características y entrenamiento de modelos de aprendizaje automático. Proporciona una interfaz unificada que permite la implementación, comparación y evaluación de diversos algoritmos de clasificación y regresión, optimizando el proceso de modelado.

PerformanceEstimation

Está diseñada para la evaluación del rendimiento de modelos predictivos. Esta librería proporciona herramientas para realizar validación cruzada y análisis de errores, permitiendo una evaluación detallada y precisa de la efectividad de los modelos predictivos mediante métricas específicas.

ROSE

El paquete ROSE (Random Over-Sampling Examples) se utiliza para abordar problemas de desbalanceo de clases en conjuntos de datos. Mediante técnicas de sobremuestreo, ROSE facilita la generación de muestras sintéticas para mejorar la representatividad de las clases minoritarias, optimizando el rendimiento de los modelos de clasificación en situaciones con desequilibrio de clases.

Openxlsx

La librería `openxlsx` en R permite crear, leer y modificar archivos Excel (.xlsx) de manera eficiente. Facilita la manipulación de hojas de cálculo, incluyendo la edición de celdas, el formato de datos y la gestión de hojas, sin necesidad de tener Excel instalado. Es útil para automatizar tareas relacionadas con el manejo de datos en Excel.

3.5.3. Importación y consolidación de los datos

En esta sección, se llevará a cabo la importación y consolidación de los múltiples archivos de datos relacionados con ataques DDoS. A continuación, se describe el proceso en detalle:

3.5.3.1. Importación de datos

Se utilizan las funciones read_csv () de la librería readr para importar archivos CSV que contienen datos sobre diferentes tipos de ataques DDoS. Cada archivo representa un tipo específico de ataque y se importa individualmente. Los archivos leídos son:

- LDAP.csv
- MSSQL.csv
- NetBIOS.csv
- Portmap.csv
- Syn.csv
- UDP.csv
- UDPLag.csv

3.5.3.2. Consolidación de datos

Una vez importados, los datos de cada archivo se combinan en un único dataframe llamado df. Esto se logra utilizando la función rbind(), que une los dataframes de forma vertical. El proceso se realiza de la siguiente manera:

- Se leen los archivos LDAP.csv y MSSQL.csv y se combinan en el dataframe df. A continuación, se eliminan los dataframes individuales ldap y mssql de la memoria usando la función “rm()”, para liberar recursos.

- Este proceso se repite para los archivos NetBIOS.csv, Portmap.csv, Syn.csv, UDP.csv, y UDPLag.csv, agregando cada uno de ellos al dataframe df y luego eliminando los objetos temporales correspondientes de la memoria.

La eliminación de los dataframes individuales después de su combinación (rm()) es una medida para optimizar el uso de memoria. Esto asegura que solo el dataframe consolidado df se mantenga en memoria, reduciendo el riesgo de sobrecargar el sistema con datos innecesarios.

3.5.3.3. Obtención de información general del dataset consolidado

Visualización general del dataframe

Se inicia con una revisión general del dataframe df utilizando la función glimpse(). Esta función de la librería dplyr proporciona una visión compacta y detallada de la estructura del dataframe, incluyendo los nombres y tipos de las variables, así como una muestra de los primeros valores en cada columna. Este paso permite obtener una comprensión preliminar del contenido del dataframe y facilita la identificación de posibles problemas estructurales.

Exploración de valores únicos

Se procede a examinar los valores únicos presentes en varias columnas del dataframe. Para ello, se emplea la función unique(). Esta función se aplica a las columnas Protocol, Label, SimillarHTTP, y Inbound para identificar las categorías distintas o valores presentes en estas variables. Esta exploración ayuda a comprender la diversidad y el rango de los datos en cada columna.

Conteo de Filas

La función nrow() se utiliza para determinar el número total de filas en el dataframe df. Este conteo proporciona información sobre el tamaño del conjunto de datos, lo cual es fundamental para la planificación de análisis y cálculos posteriores.

Conteo de Valores Faltantes

Se emplea la función is.na() combinada con sum() para calcular el número total de valores faltantes en el dataframe. Esta evaluación es esencial para identificar la cantidad y

distribución de datos ausentes, lo que permite tomar decisiones informadas sobre las estrategias de preprocesamiento y limpieza de datos.

Análisis de la Distribución por Categoría

Para analizar la distribución de los datos en función de la variable Inbound, se utilizan las funciones de la librería dplyr, incluyendo group_by(), count(), y arrange(). Primero, group_by() agrupa los datos según la variable Inbound. Luego, count() calcula el número de observaciones en cada grupo, y arrange() ordena los resultados en orden descendente según el conteo. Este análisis permite evaluar cómo se distribuyen los datos entre las distintas categorías de la variable Inbound.

3.5.4. Preprocesamiento de datos

El preprocesamiento de datos fue una etapa muy importante pues permitió preparar el dataframe para el entrenamiento de los modelos ML. A continuación, se detallan los pasos a seguir en este proceso:

3.5.4.1. Eliminación de variables no utilizadas

Se procederá a eliminar las columnas que no aportan información relevante para el análisis. Para ello, se utilizará la función select() de la librería dplyr, especificando las columnas a eliminar. Las columnas eliminadas incluyen Unnamed: 0, Flow ID, Source IP, Source Port, Destination IP, Destination Port, Timestamp, Flow Bytes/s, y Flow Packets/s. Esta acción reduce la complejidad del dataframe al mantener solo las variables necesarias para el análisis.

3.5.4.2. Creación de la etiqueta de respuesta

Se añadirá una nueva columna llamada response que indicará si una observación es benigna o maliciosa. Para esto, se empleará la función mutate() junto con ifelse() para transformar la columna Label en una variable binaria. En este nuevo campo, se asignará el valor 0 para las observaciones marcadas como "BENIGN" y el valor 1 para las demás. Posteriormente, se eliminará la columna original Label mediante select().

3.5.4.3. Examinación de valores ausentes

La presencia de valores faltantes se evaluará utilizando map_dbl() en combinación con una función anónima que cuenta los valores NA en cada columna. Esta evaluación permitirá identificar las columnas con datos ausentes, facilitando decisiones sobre la limpieza de datos. Cabe mencionar que se evitarán los NA en la columna SimillarHTTP, que se eliminará más adelante.

3.5.4.4. Eliminación de variables con un solo valor

Se identificará y eliminará la columna SimillarHTTP, que contiene un único valor, mediante la función unique(). Dado que una variable con un solo valor no proporciona información útil para el análisis, se procederá a eliminarla con select().

3.5.4.5. Filtrado de variables con pocas observaciones

Se evaluará la frecuencia de las categorías en la columna Protocol y se eliminarán las categorías con pocas observaciones. Esto se realizará usando group_by() y count() para agrupar y contar las observaciones por protocolo, seguido de un filtrado para excluir el protocolo con escasas observaciones.

3.5.4.6. Eliminación de variables dicotómicas con escasas observaciones

Se revisará la frecuencia de observaciones en varias columnas dicotómicas, incluyendo Fwd PSH Flags, SYN Flag Count, RST Flag Count, URG Flag Count, y CWE Flag Count. Las variables que presenten niveles con muy pocas observaciones se eliminarán utilizando select().

3.5.4.7. Eliminación de variables con valor cero en todas las observaciones

Se identificará y eliminará cualquier columna que contenga solo valores cero en todas las observaciones. Esto se realizará utilizando la función unique() para verificar los

valores en cada columna, y se eliminarán las columnas con valores uniformes mediante select().

3.5.4.8. Transformación de variables categóricas

Se procederá a convertir las variables categóricas en factores para su correcta utilización en análisis y modelos estadísticos. Para ello, se utilizarán las funciones as.factor() para transformar las columnas response y Protocol en variables categóricas. Esta conversión facilita el manejo y análisis de variables categóricas en modelos que requieren datos en formato de factor.

3.5.4.9. Análisis de correlación entre variables continuas

Se llevará a cabo un análisis de correlación para identificar relaciones significativas entre las variables continuas del dataframe. Para ello, se empleará la función cor() para calcular la matriz de correlación entre las variables, excluyendo response y Protocol. La matriz resultante se almacenará en un dataframe llamado correlaciones. Posteriormente, se identificará y se extraerá un conjunto de nombres de variables que presentan una alta correlación (mayor a 0.8), utilizando un bucle for y funciones de manipulación de datos. Las variables con alta correlación se almacenarán en el vector nombres_cor, que se depurará para eliminar duplicados y mantener solo las variables únicas con fuerte correlación.

3.5.4.10. Identificación de variables con varianza cercana a cero

Se evaluará la varianza de las variables para identificar aquellas con varianza próxima a cero, utilizando la función nearZeroVar() del paquete caret. Esta función calcula la varianza de cada variable, excluyendo response y Protocol, y devuelve una tabla con métricas relacionadas con la varianza. Las variables con varianza cercana a cero, que no aportan información significativa, se identificarán y se almacenarán en el dataframe varianzas.

3.5.4.11. Eliminación de variables con varianza cero

Basándose en el análisis anterior, se eliminarán las variables con varianza cercana a cero. Para ello, se filtrarán las variables identificadas con nzv == T (indicador de varianza cercana a cero) utilizando filter(), y se almacenarán sus nombres en el vector variables_nzv. Finalmente, se eliminarán estas variables del dataframe df utilizando select() en combinación con all_of() para asegurar la eliminación precisa de las variables no deseadas.

3.5.5. Análisis exploratorio de datos

3.5.5.1. Análisis de la variable de respuesta

En primer lugar, el análisis exploratorio de los datos se centró en la variable de respuesta, que es fundamental para determinar si un evento dentro del entorno IoT es benigno o corresponde a un ataque DDoS. Para entender mejor la distribución de esta variable, se procedió a calcular las frecuencias absolutas, que representan el número total de ocurrencias de cada categoría (benigno o ataque DDoS), y las frecuencias relativas, que expresan estas ocurrencias como un porcentaje del total. Este paso fue crucial para obtener una visión inicial de la proporción de casos benignos frente a los ataques dentro del conjunto de datos.

Para facilitar la interpretación, ambas frecuencias se combinaron en una única data frame, que permitió organizar la información de manera clara y concisa. Esta tabla de frecuencias fue fundamental para construir una base sólida desde la cual se pudiera interpretar la distribución de la variable de respuesta.

A continuación, se generó un gráfico de barras utilizando la librería ggplot2, una herramienta poderosa en la visualización de datos que permite una personalización detallada. Este gráfico mostró visualmente cómo se distribuyen los eventos benignos y los ataques DDoS en el conjunto de datos, permitiendo una comparación directa. Se seleccionaron colores específicos para representar cada categoría, lo que ayudó a diferenciar visualmente los tipos de eventos y a mejorar la claridad del gráfico. Adicionalmente, se ajustaron las etiquetas y la leyenda del gráfico para asegurar que la

información fuera presentada de manera comprensible y estéticamente agradable, maximizando así la eficacia comunicativa del análisis.

3.5.5.2. Análisis de las variables cualitativas

El análisis se extendió luego a las variables cualitativas del conjunto de datos, comenzando con la variable Protocol, que describe los tipos de protocolos de red utilizados. Al igual que con la variable de respuesta, se calcularon tanto las frecuencias absolutas como las relativas para cada categoría de protocolo. Este análisis fue esencial para identificar la prevalencia de cada tipo de protocolo en el entorno IoT y para comprender mejor cómo estos podrían estar relacionados con la ocurrencia de ataques DDoS.

La visualización de esta variable cualitativa también se realizó mediante un gráfico de barras utilizando ggplot2. La visualización fue diseñada para resaltar las diferencias en la prevalencia de cada protocolo. En este caso, se eliminó la leyenda del gráfico para simplificar la visualización y hacer que el foco principal fuera la comparación de frecuencias entre los diferentes protocolos.

3.5.5.3. Análisis de las variables cuantitativas

Finalmente, el análisis exploratorio abordó las variables cuantitativas del conjunto de datos, como Min Packet Length, Fwd Packet Length Min, y Avg Fwd Segment Size. Para estas variables, se calcularon diversas estadísticas descriptivas clave: el valor mínimo, el primer cuartil (Q1), la mediana, la media, el tercer cuartil (Q3), el valor máximo, la varianza y la desviación estándar. Estas estadísticas proporcionaron una comprensión detallada de cómo se distribuyen los datos dentro de cada variable, permitiendo identificar posibles sesgos, variaciones y tendencias que podrían ser indicativos de patrones subyacentes relevantes para la detección de ataques DDoS.

Para visualizar estas variables cuantitativas, se generaron gráficos de densidad segmentados por la variable de respuesta (response). Estos gráficos fueron diseñados para mostrar cómo varían las distribuciones de las variables cuantitativas entre eventos benignos

y ataques DDoS. La segmentación por la variable de respuesta permitió identificar diferencias significativas en las distribuciones, lo cual podría sugerir características clave que separan los eventos benignos de los maliciosos.

Este análisis exploratorio no solo proporcionó una visión integral de las características del conjunto de datos, sino que también estableció las bases para las etapas subsiguientes de modelado predictivo. Al comprender en profundidad las relaciones y distribuciones dentro de los datos, fue posible informar decisiones cruciales en la selección, ajuste y evaluación de modelos de detección de ataques DDoS en entornos IoT.

3.5.5.4. Preparación del conjunto de datos para el ajuste de los modelos predictivos

En esta etapa, se llevarán a cabo varios pasos para preparar los datos para el análisis y la modelización. A continuación, se detallan las acciones a seguir:

Separación del conjunto de datos en entrenamiento y prueba (Data splitting)

Se dividirá el conjunto de datos df en dos subconjuntos: uno para entrenamiento y otro para prueba. Utilizando la función createDataPartition() del paquete caret, se generará un índice de partición que seleccionará aleatoriamente el 70% de los datos para el conjunto de entrenamiento (df_train) y el 30% restante para el conjunto de prueba (df_test). La partición se fijará con set.seed(456) para garantizar la reproducibilidad del proceso. Posteriormente, se eliminarán las tablas temporales utilizadas en el proceso de partición, como df, train, correlaciones, y varianzas, con la función rm().

Balanceo de la clase de respuesta

Con el objetivo de abordar cualquier desbalance en la variable de respuesta, se aplicará un método de balanceo en el conjunto de entrenamiento. La función ovun.sample() del paquete ROSE será utilizada para realizar un muestreo bajo (method = "under") para equilibrar las clases de la variable response en df_train. Se establecerá un porcentaje de la clase minoritaria de 30% (p = 0.3) para obtener un conjunto de datos balanceado. El resultado del muestreo se almacenará en df_balanced. Además, se renombrarán las

variables de df_train y df_test para asegurar que no contengan espacios ni caracteres especiales, utilizando las funciones gsub().

Estandarización de variables continuas

Se normalizarán las variables numéricas en los conjuntos de datos balanceado y de prueba para asegurar que todas las variables contribuyan de manera equitativa al análisis. Esto se logrará utilizando la función mutate_if() del paquete dplyr con la función scale(), que estandariza las variables numéricas a una media de 0 y una desviación estándar de 1. Esta estandarización se aplicará a df_balanced y df_test.

3.5.6. Ajuste de los modelos predictivos

3.5.6.1. Ajuste del modelo KNN

Para la implementación del modelo K-Nearest Neighbors (KNN), se definieron previamente los hiperparámetros y las configuraciones de validación cruzada con repetición, elementos clave para garantizar un proceso de entrenamiento robusto y riguroso. Para el ajuste de este modelo se siguieron las etapas expuestas a continuación:

Configuración de hiperparámetros y validación

El primer paso en la configuración del modelo KNN fue la selección de los hiperparámetros, específicamente el número de vecinos (k). Se optó por evaluar tres valores distintos para k (5, 20 y 50), permitiendo así explorar cómo la variación en el número de vecinos impacta en la precisión del modelo. Estos valores se definieron en un data frame, el cual será utilizado posteriormente en el proceso de ajuste del modelo.

A continuación, se estableció el esquema de validación cruzada con repetición, configurando el modelo para realizar 2 particiones con 2 repeticiones cada una. Esto permite obtener una evaluación más fiable del rendimiento del modelo, ya que la validación cruzada ayuda a mitigar posibles sesgos asociados con la división específica del conjunto de datos.

Además, para asegurar la reproducibilidad de los resultados y un control más estricto sobre el proceso de entrenamiento, se generaron semillas aleatorias. Estas semillas fueron

configuradas para cada repetición de la validación cruzada, utilizando un proceso de muestreo que asigna valores aleatorios a cada combinación de partición y repetición. De esta forma, se asegura que cada iteración del proceso de entrenamiento y validación utilice los mismos valores aleatorios, facilitando la replicabilidad del experimento.

Definición del entrenamiento del modelo

Con los hiperparámetros y las configuraciones de validación definidos, se procedió a establecer los controles del proceso de entrenamiento mediante la función trainControl. En este paso, se especificaron los métodos de validación cruzada, el número de repeticiones, las semillas generadas previamente, y se desactivó la visualización de los detalles iterativos del proceso (verboseIter = FALSE) para simplificar el flujo de trabajo.

Ajuste del modelo KNN

Finalmente, se realizó el ajuste del modelo KNN utilizando la función train. Este proceso involucró el uso del conjunto de datos balanceado (df_balanced), donde se ajustó el modelo para predecir la variable de respuesta en función de las demás variables presentes en el conjunto de datos. La métrica principal utilizada para evaluar el rendimiento del modelo fue la precisión (Accuracy), un indicador esencial en la clasificación binaria, especialmente en el contexto de la detección de ataques DDoS.

El proceso de ajuste del modelo también tomó en cuenta la grilla de hiperparámetros definida anteriormente, permitiendo así seleccionar el valor de k que maximiza la precisión del modelo en las particiones de validación cruzada.

3.5.6.2. Ajuste del modelo Random Forest

El ajuste de este modelo requirió de las siguientes etapas:

Definición de hiperparámetros

Se establecerán los hiperparámetros a evaluar para el modelo de Random Forest (RF). Aunque en el código se ha comentado la expansión de una cuadrícula de hiperparámetros, se utilizará una cuadrícula simplificada que solo incluye el parámetro mtry con valores de 5, 20 y 35. Esta cuadrícula se generará utilizando la función expand.grid(),

que crea un dataframe con todas las combinaciones posibles de los valores especificados para mtry.

Configuración de la repetición y semillas

Para garantizar la reproducibilidad y la robustez en el entrenamiento del modelo, se establecerán particiones y repeticiones del proceso de validación cruzada. Se definirán dos particiones (particiones = 2) y dos repeticiones (repeticiones = 2). Se generarán semillas aleatorias utilizando la función sample.int() para cada combinación de partición y repetición. Estas semillas se almacenarán en una lista denominada seeds, la cual será utilizada para controlar la aleatoriedad en la validación cruzada.

Definición del control de entrenamiento

Se configurará el proceso de entrenamiento del modelo utilizando la función trainControl() del paquete caret. El método de validación cruzada será "repeatedcv" con las particiones y repeticiones previamente definidas. La lista de semillas seeds se utilizará para asegurar la reproducibilidad del proceso. Además, se especificará que se devuelvan los resultados finales (returnResamp = "final") y se desactivará la salida detallada (verboseIter = FALSE).

Ajuste del modelo

Se ajustará el modelo de Random Forest (rf) utilizando la función train() del paquete caret. Se aplicará el modelo sobre el conjunto de datos balanceado (df_balanced) y se evaluará la métrica de precisión (Accuracy). La cuadrícula de hiperparámetros definida anteriormente se pasará a través del parámetro tuneGrid. La función train() utilizará el control de entrenamiento configurado para optimizar el modelo.

Predicciones con el modelo óptimo

Se realizará la predicción sobre el conjunto de prueba (df_test) utilizando el modelo ajustado. La función predict() se empleará para obtener las predicciones del modelo en función de las características del conjunto de prueba. Las predicciones se convertirán en un factor con niveles correspondientes a la variable de respuesta de prueba.

Evaluación del modelo

La evaluación del modelo se llevará a cabo mediante la matriz de confusión utilizando la función confusionMatrix(), que comparará las predicciones del modelo con las etiquetas reales del conjunto de prueba. Se evaluará el rendimiento general del modelo en la clasificación de las instancias.

Predicciones con umbral modificado

Se ajustarán las predicciones del modelo utilizando un umbral modificado para la probabilidad de clase positiva. Se obtendrán las probabilidades predichas para la clase positiva utilizando la opción type = "prob" en la función predict(). Luego, se calculará la curva ROC para evaluar el rendimiento del modelo en diferentes umbrales de clasificación. Las predicciones se ajustarán según un umbral de 0.25, y se realizará una nueva evaluación con la matriz de confusión utilizando estas predicciones ajustadas.

3.5.6.3. Ajuste del modelo XGBoost

En esta fase, se ajustará un modelo de XGBoost para la clasificación, siguiendo un enfoque detallado para optimizar los hiperparámetros y evaluar el rendimiento del modelo. A continuación, se describen los pasos que se llevarán a cabo:

Definición de parámetros de entrenamiento

Se establecerán las particiones y repeticiones para la validación cruzada del modelo de XGBoost. Se configurarán dos particiones (particiones = 2) y dos repeticiones (repeticiones = 2). Esto permitirá evaluar el modelo en múltiples subdivisiones del conjunto de datos para obtener una estimación más robusta de su rendimiento.

Configuración de hiperparámetros

Se definirán los hiperparámetros a ajustar para el modelo de XGBoost utilizando la función expand.grid(). Los parámetros especificados incluyen:

- lambda: Término de regularización L2 (se utilizará el valor 0).
- alpha: Término de regularización L1 (se utilizará el valor 0).

- nrounds: Número de iteraciones para el entrenamiento (se evaluarán 50, 100 y 200 iteraciones).
- eta: Tasa de aprendizaje (se evaluarán los valores 0.3, 0.2 y 0.5).

Generación de semillas aleatorias

Para garantizar la reproducibilidad del proceso de validación cruzada, se generarán semillas aleatorias utilizando la función sample.int(). Se creará una lista de semillas (seeds) con el tamaño adecuado para las particiones y repeticiones, asegurando que cada partición y repetición tenga una semilla específica. La lista de semillas se utilizará para controlar la aleatoriedad durante la validación cruzada.

Configuración del control de entrenamiento

Se definirá el control de entrenamiento del modelo utilizando la función trainControl(). El método de validación cruzada será "repeatedcv", con las particiones y repeticiones especificadas anteriormente. La lista de semillas seeds se aplicará para asegurar la consistencia en la validación cruzada. Se configurará para devolver los resultados finales (returnResamp = "final") y se desactivará la salida detallada (verboseIter = FALSE).

Ajuste del modelo XGBoost

Se ajustará el modelo de XGBoost utilizando la función train() del paquete caret. Se aplicará el modelo sobre el conjunto de datos balanceado (df_balanced) con el método "xgbLinear". Se utilizarán los hiperparámetros definidos anteriormente para encontrar la mejor configuración. La métrica de evaluación será la precisión (Accuracy). La función train() se ejecutará con el control de entrenamiento previamente configurado.

Predicciones con el modelo óptimo

Una vez ajustado el modelo, se realizará la predicción sobre el conjunto de prueba (df_test). Utilizando la función predict(), se obtendrán las predicciones del modelo. Estas predicciones se convertirán en un factor para alinearse con los niveles de la variable de respuesta del conjunto de prueba. Se evaluará el rendimiento del modelo mediante la matriz

de confusión, utilizando la función confusionMatrix() para comparar las predicciones con las etiquetas reales.

Evaluación con probabilidades

Se obtendrán las probabilidades predichas para la clase positiva utilizando la opción type = "prob" en la función predict(). La función roc.curve() se empleará para generar la curva ROC y evaluar el rendimiento del modelo en términos de sensibilidad y especificidad a diferentes umbrales.

3.5.7. Elementos metodológicos específicos para TI

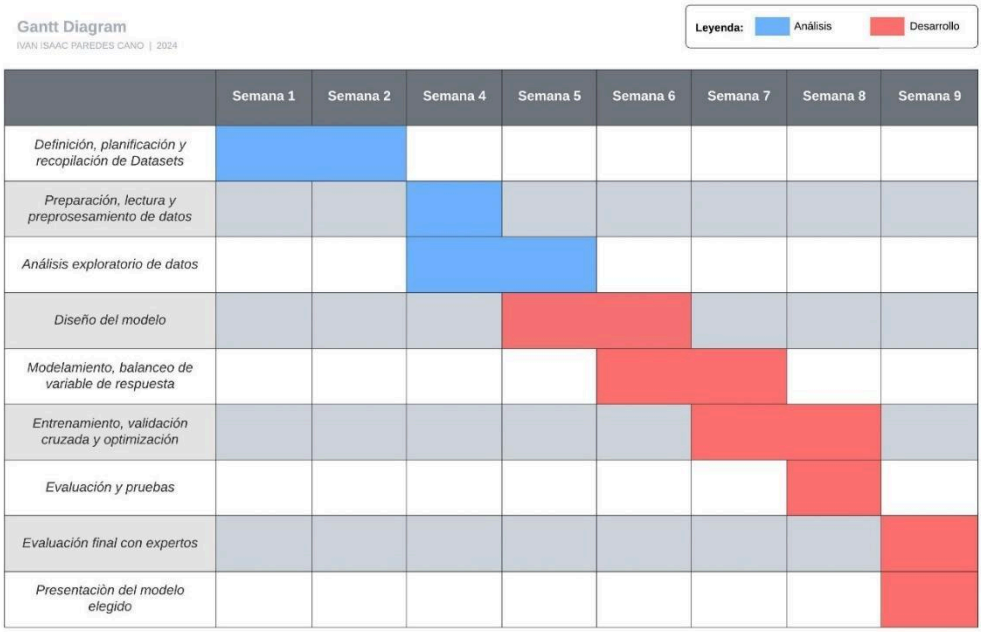
- **Diseño del proyecto**
  - **Tipo de proyecto**: Desarrollo de un modelo de machine learning.
  - **Enfoque metodológico**: Kanban
  - **Modelo de desarrollo**: Entrenamiento de modelos de machine learning.
- **Recopilación de información**
  - **Fuentes de datos:** Expertos en Redes y Analítica de datos, conjunto de datos CICDDoS2019.
  - **Herramientas de recopilación**: Entrevistas.
  - **Procedimientos de recopilación**: Se realizará entrevistas a los expertos en las áreas más significativas para el proyecto.
- **Desarrollo y diseño**
  - **Planificación**: Definición de etapas para el desarrollo.
  - **Requisitos del sistema**: Sistema Operativo Windows 11, 12 GB de RAM, procesador 3.7 GHz.
  - **Diseño del sistema/producto**: Desarrollo del modelo utilizando algoritmos KNN, Random Forest, XGboost.
  - **Tecnologías y herramientas**: Uso de lenguaje de programación R, en el entorno de desarrollo de RStudio. Se utilizan librerías como Readr, Tidyverse, Caret , performanceEstimation y ROSE.

- **Análisis de datos**
    - **Métodos de análisis**: Exploratorio.
    - **Herramientas de análisis**: Wireshark para la exportación los dataset captura de paquetes (PCAP), Excel, RStudio.
    - **Interpretación de resultados**: Datos cuantitativos y cualitativos (datasets) para el entrenamiento de modelos de machine learning.
- **Evaluación del proyecto**
    - **Criterios de evaluación**: Efectividad y precisión del modelo para detectar ataques DDoS.
    - **Instrumentos de evaluación**: RStudio.
- **Cronograma de actividades**

A continuación, se detalla el cronograma de las actividades que se han planificado para el desarrollo del modelo propuesto:

Ilustración 1

Planificación de actividades. Elaborado por el equipo de trabajo.



4. Resultados

La ejecución de las etapas expuestas en el capítulo de Metodología del presente proyecto permitió obtener los siguientes resultados, alcanzando los objetivos específicos planteados en el capítulo correspondiente. Estos resultados se exponen en los subapartados del presente capítulo.

4.1. De la consolidación de los datos

Para lograr la consolidación de los diferentes conjuntos de datos en un único dataset, se emplearon varias funciones clave de manipulación de datos. En primer lugar, se utilizó la función read_csv() para cargar cada uno de los archivos CSV que contenían datos de distintos tipos de ataques DDoS, como LDAP, MSSQL, NetBIOS, Portmap, Syn, UDP y UDPLag.

Una vez cargados los datos, la función rbind() fue utilizada repetidamente para combinar estos conjuntos de datos en un solo marco de datos unificado. Cada vez que se añadía un nuevo conjunto de datos al marco consolidado, el archivo original se eliminaba de la memoria mediante la función rm(), optimizando así el uso de recursos computacionales durante el proceso.

Este enfoque sistemático permitió la creación de un dataset robusto, compuesto por 20,364,525 filas y 88 columnas, el cual fue procesado y usado en el entrenamiento y evaluación de los modelos de detección de ataques DDoS.

Unnamed: 0	Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration
13605	172.16.0.5-192.168.50.4-870-2908-17	172.16.0.5	870	192.168.50.4	2908	17	2018-11-03 10:09:00	1
62631	172.16.0.5-192.168.50.4-871-53796-17	172.16.0.5	871	192.168.50.4	53796	17	2018-11-03 10:09:00	48
143869	172.16.0.5-192.168.50.4-648-40660-17	172.16.0.5	648	192.168.50.4	40660	17	2018-11-03 10:09:00	1
16171	172.16.0.5-192.168.50.4-872-54308-17	172.16.0.5	872	192.168.50.4	54308	17	2018-11-03 10:09:00	1
80845	172.16.0.5-192.168.50.4-873-40653-17	172.16.0.5	873	192.168.50.4	40653	17	2018-11-03 10:09:00	1
107859	172.16.0.5-192.168.50.4-874-52210-17	172.16.0.5	874	192.168.50.4	52210	17	2018-11-03 10:09:00	1

Ilustración 2: Consolidación de datos

4.2. De la inspección general del dataset

Se realizó un análisis preliminar del conjunto de datos para comprender su estructura y contenido. En esta fase, se examinó la variedad de variables presentes, sus formatos, y la dimensión total del conjunto de datos. Además, se revisaron los valores únicos de cada variable para identificar cuáles debían ser clasificadas como cualitativas. Esta revisión incluyó un conteo exhaustivo de los valores faltantes, que fue registrado para un análisis más detallado posterior.

Asimismo, se llevó a cabo un proceso de limpieza de datos eliminando aquellas variables que no aportaban valor al análisis, tales como identificadores de registros que no contribuyen a la información analítica esencial. Esta preparación inicial permitió una visión clara de los datos disponibles y facilitó la selección de variables relevantes para el análisis posterior.

```
Rows: 20,364,525
Columns: 88
$ `Unnamed: 0`               <dbl> 13605, 62631, 143869, 16171, 80845, 107859, 39249, 9812…
$ `Flow ID`                  <chr> "172.16.0.5-192.168.50.4-870-2908-17", "172.16.0.5-192.…
$ `Source IP`                <chr> "172.16.0.5", "172.16.0.5", "172.16.0.5", "172.16.0.5",…
$ `Source Port`              <dbl> 870, 871, 648, 872, 873, 874, 875, 876, 877, 648, 878, …
$ `Destination IP`           <chr> "192.168.50.4", "192.168.50.4", "192.168.50.4", "192.16…
$ `Destination Port`         <dbl> 2908, 53796, 40660, 54308, 40653, 52210, 34461, 10877, …
$ Protocol                   <dbl> 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17, 17,…
$ Timestamp                  <dttm> 2018-11-03 10:09:00, 2018-11-03 10:09:00, 2018-11-03 1…
$ `Flow Duration`            <dbl> 1, 48, 1, 1, 1, 1, 1, 48, 1, 1, 1, 1, 2, 1, 1, 48, 1, 1…
$ `Total Fwd Packets`        <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2…
$ `Total Backward Packets`   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ `Total Length of Fwd Packets` <dbl> 422, 530, 458, 458, 458, 458, 494, 530, 458, 458, 458, …
$ `Total Length of Bwd Packets` <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ `Fwd Packet Length Max`    <dbl> 211, 265, 229, 229, 229, 229, 247, 265, 229, 229, 229, …
$ `Fwd Packet Length Min`    <dbl> 211, 265, 229, 229, 229, 229, 247, 265, 229, 229, 229, …
$ `Fwd Packet Length Mean`   <dbl> 211, 265, 229, 229, 229, 229, 247, 265, 229, 229, 229, …
$ `Fwd Packet Length Std`    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ `Bwd Packet Length Max`    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ `Bwd Packet Length Min`    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ `Bwd Packet Length Mean`   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
$ `Bwd Packet Length Std`    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
```

Ilustración 3: Limpieza de datos

4.3. Del preprocesamiento de datos

4.3.1. Del tratamiento de los valores ausentes

La tabla 1 presenta la cantidad de valores ausentes para cada variable en el conjunto de datos. Se observó que la variable «SimillarHTTP» contenía 1,738 valores faltantes, los cuales fueron eliminados en una etapa posterior del proceso de limpieza de datos.

Tabla 1 Valores ausentes para cada variable

Nombre de variable	Cantidad de faltantes
Protocol	0
Flow Duration	0
Total Fwd Packets	0
Total Backward Packets	0
Total Length of Fwd Packets	0
Total Length of Bwd Packets	0
Fwd Packet Length Max	0
Fwd Packet Length Min	0
Fwd Packet Length Mean	0
Fwd Packet Length Std	0
Bwd Packet Length Max	0
Bwd Packet Length Min	0
Bwd Packet Length Mean	0

Bwd Packet Length Std	0
Flow IAT Mean	0
Flow IAT Std	0
Flow IAT Max	0
Flow IAT Min	0
Fwd IAT Total	0
Fwd IAT Mean	0
Fwd IAT Std	0
Fwd IAT Max	0
Fwd IAT Min	0
Bwd IAT Total	0
Bwd IAT Mean	0
Bwd IAT Std	0
Bwd IAT Max	0
Bwd IAT Min	0
Fwd PSH Flags	0
Bwd PSH Flags	0
Fwd URG Flags	0
Bwd URG Flags	0

Fwd Header Length	0
Bwd Header Length	0
Fwd Packets/s	0
Bwd Packets/s	0
Min Packet Length	0
Max Packet Length	0
Packet Length Mean	0
Packet Length Std	0
Packet Length Variance	0
FIN Flag Count	0
SYN Flag Count	0
RST Flag Count	0
PSH Flag Count	0
ACK Flag Count	0
URG Flag Count	0
CWE Flag Count	0
ECE Flag Count	0
Down/Up Ratio	0
Average Packet Size	0

Avg Fwd Segment Size	0
Avg Bwd Segment Size	0
Fwd Header Length.1	0
Fwd Avg Bytes/Bulk	0
Fwd Avg Packets/Bulk	0
Fwd Avg Bulk Rate	0
Bwd Avg Bytes/Bulk	0
Bwd Avg Packets/Bulk	0
Bwd Avg Bulk Rate	0
Subflow Fwd Packets	0
Subflow Fwd Bytes	0
Subflow Bwd Packets	0
Subflow Bwd Bytes	0
Init_Win_bytes_forward	0
Init_Win_bytes_backward	0
act_data_pkt_fwd	0
min_seg_size_forward	0
Active Mean	0
Active Std	0
Active Max	0
Active Min	0
Idle Mean	0
Idle Std	0
Idle Max	0
Idle Min	0

SimillarHTTP	1738
Inbound	0
response	0

4.3.2. Del tratamiento de variables dicotómicas desbalanceadas

Se eliminaron las variables dicotómicas en las que uno de los niveles presentaba un número significativamente bajo de observaciones en comparación con el otro nivel. Este comportamiento se detalla en la tabla 2.

Tabla 2 Observaciones en variable "Fwd PSH Flags"

Fwd PSH Flags	Número de observaciones	Proporción
0	20,354,219	0.9994939
1	10,306	0.0005061

4.3.3. Del tratamiento a variables con 0 como único valor

Se eliminaron variables cuyo único valor en la base de datos era 0.

Tabla 3 Variables con 0 como único valor

Variable	Valor
Bwd PSH Flags	0
Fwd URG Flags	0
Bwd URG Flags	0
FIN Flag Count	0
PSH Flag Count	0
ECE Flag Count	0
Fwd Avg Bytes/Bulk	0

Fwd Avg Packets/Bulk	0
Bwd Avg Bulk Rate	0
Fwd Avg Bulk Rate	0
Bwd Avg Bytes/Bulk	0
Bwd Avg Packets/Bulk	0

4.3.4. Del tratamiento a variables con varianza cercana a 0

Se eliminaron las variables con una varianza cercana a cero, identificadas por el valor "verdadero" en la columna “nzv”. Estas variables, al tener una varianza extremadamente baja, no aportan variabilidad significativa a los datos y, por lo tanto, no contribuyen de manera útil al análisis predictivo. La función `nearZeroVar()` del paquete `caret` se utilizó para detectar estas variables, las cuales fueron marcadas con un valor "true" en el indicador “nzv” del resultado generado por la función. Este proceso de eliminación ayuda a simplificar el conjunto de datos y a reducir la complejidad del modelo al eliminar características redundantes o no informativas.

Tabla 4 Variables con varianza cercana a 0

Variables	freqRatio	percentUnique	zeroVar	nzv
Fwd Packet Length Std	31.23	0.03	FALSO	VERDADERO
Bwd Packet Length Std	81,080.41	0.03	FALSO	VERDADERO
Flow IAT Std	383.29	6.83	FALSO	VERDADERO
Fwd IAT Std	7,668.20	6.75	FALSO	VERDADERO
Bwd IAT Std	83,192.49	1.39	FALSO	VERDADERO
Bwd Packets/s	182.38	2.05	FALSO	VERDADERO

Packet Length Std	31.18	0.04	FALSO	VERDADERO
Packet Length Variance	31.18	0.04	FALSO	VERDADERO
Active Mean	265.79	0.41	FALSO	VERDADERO
Active Std	4,351.07	0.61	FALSO	VERDADERO
Active Max	265.79	0.39	FALSO	VERDADERO
Active Min	44.61	0.04	FALSO	VERDADERO
Idle Mean	1,801,333.18	2.35	FALSO	VERDADERO
Idle Std	4,964,267.50	2.21	FALSO	VERDADERO
Idle Max	1,100,814.72	2.29	FALSO	VERDADERO
Idle Min	535,531.49	2.23	FALSO	VERDADERO

4.3.5. De la transformación de variables categóricas

En el proceso de preparación de datos, se identificaron dos variables importantes para el análisis: la variable de respuesta y el protocolo de las conexiones de red. Ambas variables se clasificaron como cualitativas debido a su naturaleza categórica. Para garantizar que el lenguaje de programación R las tratara adecuadamente, se procedió a transformar estas variables a un formato que R pudiera reconocer como factor.

En R, la conversión a factor se realiza mediante la función as.factor(). Esta transformación es crucial porque permite que el software distinga y gestione cada categoría o nivel de la variable cualitativa de manera efectiva. Al convertir estas variables a factores, se asegura que R maneje correctamente las categorías en los análisis posteriores, evitando

posibles interpretaciones erróneas que podrían surgir si se trataran como variables numéricas.

La transformación a factor facilita la aplicación de técnicas estadísticas adecuadas para variables categóricas y mejora la precisión en el análisis de datos. Esto asegura que los algoritmos de Machine Learning y los modelos estadísticos puedan interpretar y utilizar correctamente las variables en su contexto categórico, contribuyendo a resultados más fiables y significativos en el análisis.

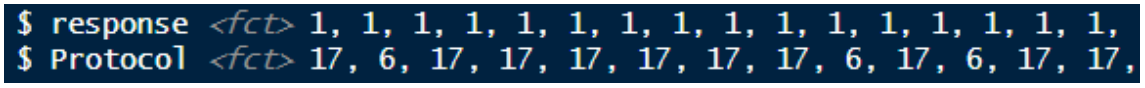


Ilustración 4: Transformación de variables categóricas

4.3.6. Del análisis de correlación entre variables continuas

Este examen reveló que muchas de las variables mostraban fuertes correlaciones entre sí, lo que se atribuyó a la naturaleza de las variables, que a menudo eran derivadas de operaciones matemáticas sobre otras variables. Estas altas correlaciones no resultaron ser un factor determinante para la eliminación de variables del conjunto de datos.

El motivo de esta decisión radica en la capacidad de los métodos de Machine Learning para manejar la multicolinealidad de manera efectiva. Los algoritmos de Machine Learning, especialmente aquellos que utilizan técnicas de regularización o que son inherentemente robustos a la multicolinealidad, pueden mitigar los efectos negativos que la alta correlación entre variables podría provocar en el análisis. En consecuencia, en lugar de eliminar variables basándose únicamente en la correlación, se optó por conservarlas y permitir que el modelo de Machine Learning optimizara su rendimiento durante el entrenamiento.

Este enfoque permite mantener una mayor cantidad de información potencialmente útil, lo cual es fundamental para el desarrollo de modelos predictivos más robustos y precisos. Así, el análisis de correlaciones se utilizó como una herramienta complementaria, pero no determinante, en el proceso de selección de variables, permitiendo que el modelo final se beneficiara de la riqueza completa del conjunto de datos disponible.

Tabla 5: Análisis de correlación entre variables continuas

RowNames	Flow.Duration	Total.Fwd.Packets	Total.Backward.Packets
Flow Duration	1	0.25	0.10
Total Fwd Packets	0.25	1	0.93
Total Backward Packets	0.10	0.93	1
Total Length of Fwd Packets	-0.11	0.19	0.03
Total Length of Bwd Packets	0.01	0.92	0.99
Fwd Packet Length Max	-0.15	-0.04	-0.03
Fwd Packet Length Min	-0.15	-0.05	-0.03

4.4. Del análisis exploratorio de datos

4.4.1. Análisis descriptivo de la variable de respuesta

La variable de respuesta muestra un claro desbalanceo de un nivel con respecto a otro, los registros que corresponden a un ataque DDoS conforma la mayor parte del conjunto de datos.

Tabla 6: Frecuencias de la variable de respuesta

Respuesta	Frecuencia absoluta	Frecuencia relativa
0	55,904	0.002745362

1	20,307,169	0.997254638

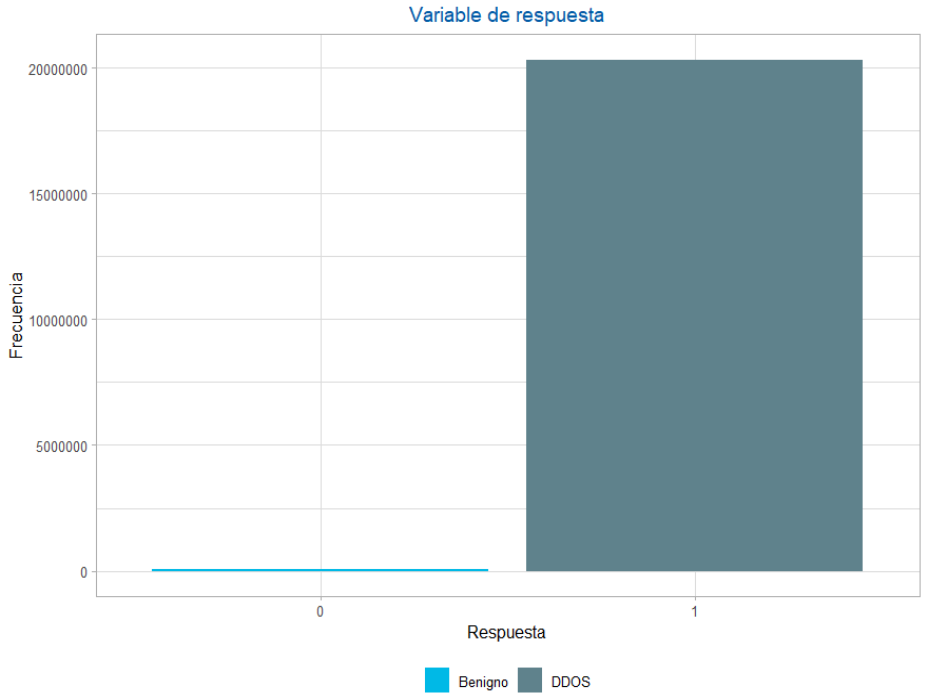


Ilustración 5: Gráfico de barras de la variable de respuesta

4.4.2. Análisis descriptivo de las variables predictoras

4.4.2.1. Análisis descriptivo de las variables cualitativas

La única variable cualitativa que forma parte de las variables predictoras en el conjunto de datos es el protocolo, la cual cuenta con 2 valores: TCP y UDP respectivamente. Vemos que el protocolo UDP representa más del 70% de los registros, este es un desbalanceo leve, por lo que no causa problemas en el desarrollo de los modelos de Machine Learning.

Tabla 7: Frecuencias de la variable "Protocolo".

Protocolo	Frecuencia absoluta	Frecuencia relativa
6	4,927,385	0.24
17	15,435,688	0.76

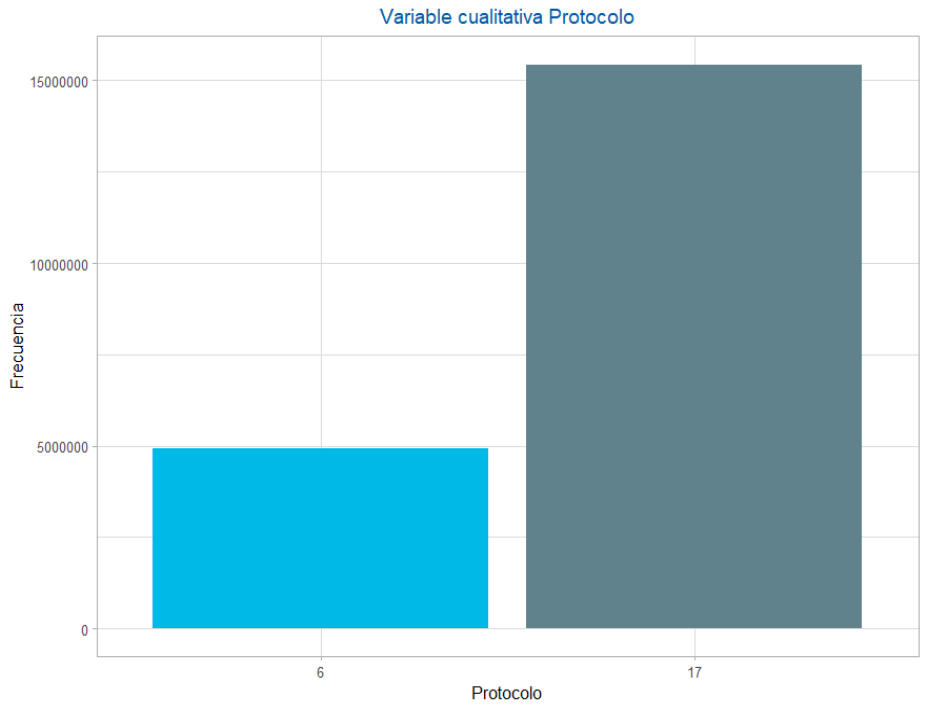


Ilustración 6: Gráfico de barras de la variable Protocolo.

4.4.2.2. Análisis descriptivo de las variables cuantitativas

Variable "Min Packet Length"

Esta variable contiene el tamaño mínimo de paquetes, vemos que la mayor parte de los registros tienen un tamaño mínimo de paquetes que va desde 0 a 469. La distribución de esta variable muestra una diferencia cuando se trata de conexiones benignas y DDOS, se observa que en las observaciones DDOS se llegan a tener valores mucho más altos en esta variable.

Tabla 8: Estadísticos de la variable "Min Packet Length"

Min	Q1	Mediana	Media	Q3	Max	Varianza	SD
0	211	330	409.92	469	1,472	170,712.34	413.17

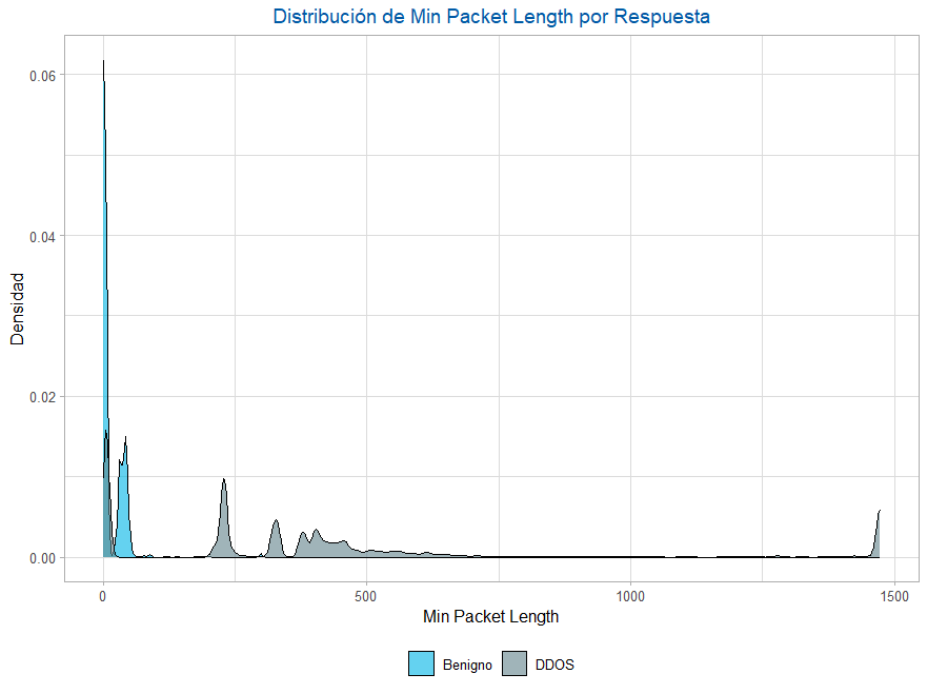


Ilustración 7 Gráfico de densidad de la variable "Min Packet Length"

Variable "Fwd Packet Length Min"

Esta variable contiene el tamaño mínimo de paquetes que están siendo enviados desde el origen al destino en una comunicación de red, vemos que la mayor parte de los registros tienen un tamaño mínimo de paquetes que va desde 0 a 469. La distribución de esta variable muestra una diferencia cuando se trata de conexiones benignas y DDOS, se observa que en las observaciones DDOS se llegan a tener valores mucho más altos en esta variable.

Tabla 9 Estadísticos de la variable "Fwd Packet Length Min"

Min	Q1	Mediana	Media	Q3	Max	Varianza	SD
0	211	330	409.93	469	2,131	170,714.07	413.18

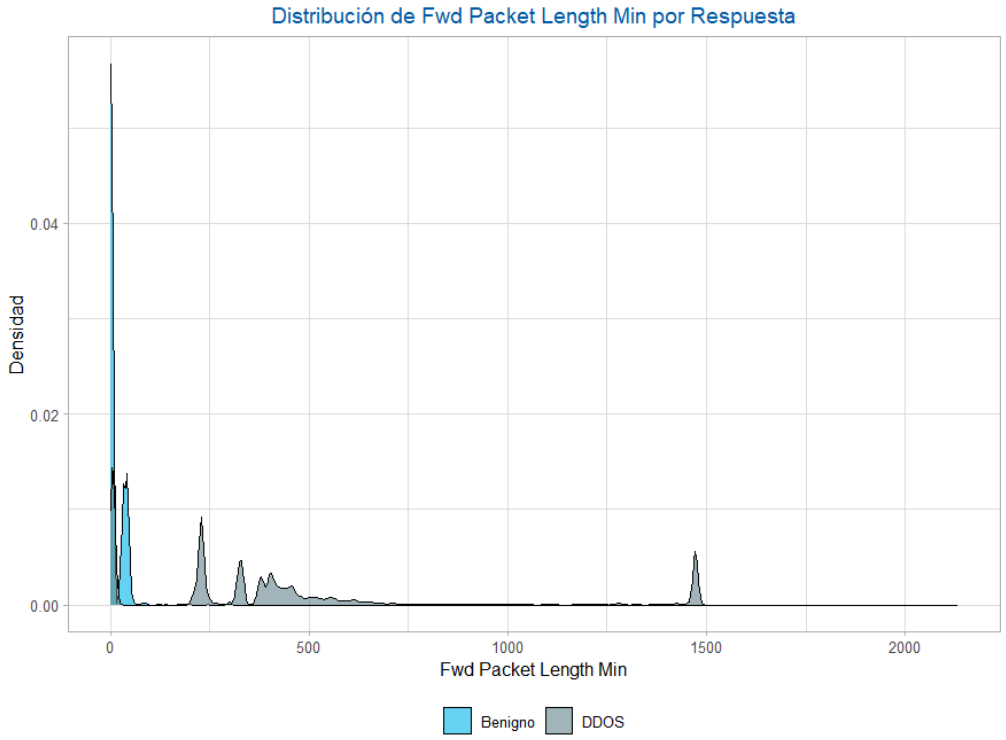


Ilustración 8 Gráfico de densidad de la variable "Fwd Packet Length Min"

Variable "Avg Fwd Segment Size"

Esta variable contiene el tamaño promedio de los segmentos enviados en la dirección de cliente a servidor para cada conexión registrada. Se observa que la mayoría de las conexiones benignas se concentran en valores de 0 a 250 aproximadamente. No obstante, las conexiones DDOS se distribuyen en valores más altos para esta variable.

Tabla 10 Estadísticos de la variable "Avg Fwd Segment Size"

Min	Q1	Mediana	Media	Q3	Max	Varianza	SD
0	211	360	412.48	469	2131	170,320.02	412.70

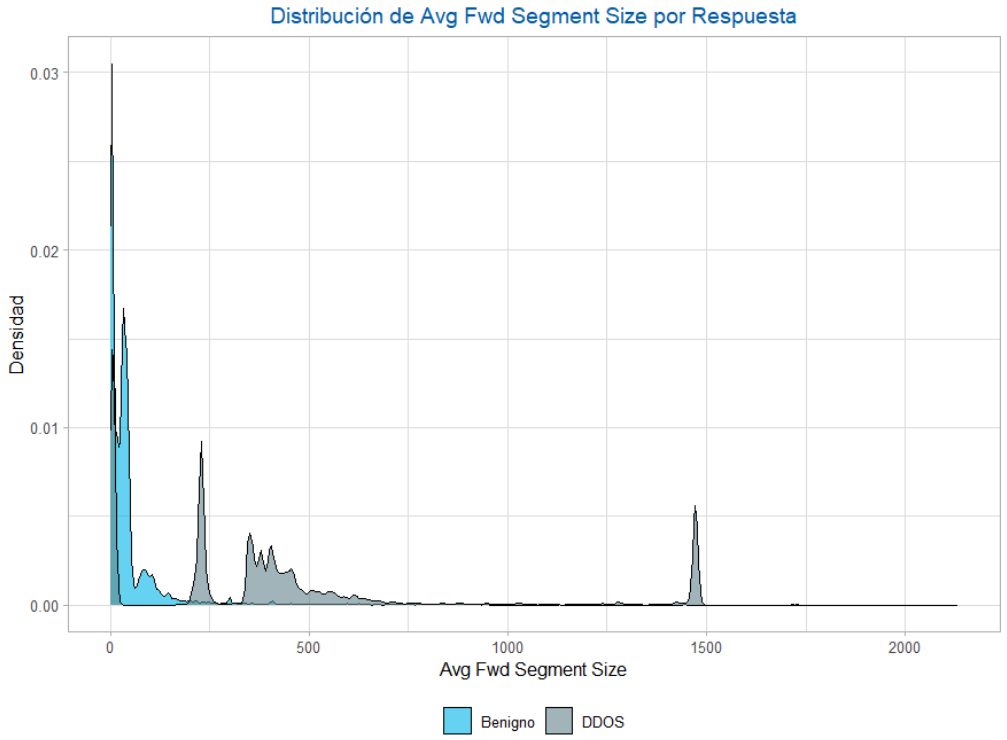


Ilustración 9 Gráfico de densidad de la variable "Avg Fwd Segment Size"

4.5. De la preparación del conjunto de datos para el ajuste de los modelos predictivos

4.5.1. De la separación del conjunto de datos en datos de entrenamiento y prueba (data splitting)

Se separa el conjunto de datos con una proporción del 70% para el conjunto de entrenamiento y un 30% para el conjunto de prueba.

```
train <- createDataPartition(y = df$response, p = 0.7, list = FALSE, times = 1)
df_train <- df[train, ]
df_test  <- df[-train, ]
```

Ilustración 10 Data splitting.

4.5.2. Del balanceo de la clase de respuesta

Se utiliza el método de submuestreo para balancear la variable de respuesta, la proporción de desbalanceo quedará en 30% para la clase minoritaria y 70% para la clase mayoritaria.

```
df_balanced <- ovun.sample(response~., data = df_train, method = "under", p = 0.3, seed = 456)
df_balanced <- df_balanced[[3]]
```

Ilustración 11 Balanceo de la clase de respuesta.

4.5.3. De la estandarización de variables continuas

Las variables continuas se estandarizaron para tener valores dentro de la misma escala. A continuación, se visualiza una muestra de los datos.

Tabla 11 Estandarización de variables continuas .

Protocol	Flow_Duration	Total_Fwd_Packets	Total_Backward_Packets
17	-0.22	-0.04	-0.04
6	-0.22	-0.04	-0.01
17	-0.22	-0.04	-0.04
17	-0.22	-0.04	-0.04
17	-0.22	-0.01	-0.04
17	-0.22	-0.04	-0.04

Finalmente, luego de la ejecución de las etapas anteriores de preprocesamiento de datos, el conjunto de datos finales tiene una cantidad de 20,363,073 de filas y 44 variables incluida la variable de respuesta.

4.6. Del ajuste de los modelos predictivos

4.6.1. Del ajuste del modelo KNN

4.6.1.1. Configuración del proceso de entrenamiento

```
particiones  <- 2
repeticiones <- 2

# Hiperparámetros
hiperparametros <- data.frame(k = c(5, 20, 50))

set.seed(456)
seeds <- vector(mode = "list", length = (particiones * repeticiones) + 1)
for (i in 1:(particiones * repeticiones)) {
 seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
 }
seeds[[(particiones * repeticiones) + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
control_train <- trainControl(method = "repeatedcv", number = particiones,
                              repeats = repeticiones, seeds = seeds,
                              returnResamp = "final", verboseIter = FALSE)

# AJUSTE DEL MODELO
set.seed(456)
modelo_knn <- train(response ~ ., data = df_balanced,
                    method = "knn",
                    tuneGrid = hiperparametros,
                    metric = "Accuracy",
                    trControl = control_train)
```

Ilustración 12 Configuración del proceso de entrenamiento de KNN.

Se utilizaron 2 particiones y 2 repeticiones para la validación cruzada aplicada en el procedimiento de optimización de hiperparámetros. Los valores que se probaron para el hiperparámetro “k” fueron: 5, 20 y 50. En la variable “seeds” se almacenaron las semillas necesarias para cada uno de los ajustes del modelo en la validación cruzada.

4.6.1.2. Ajuste del modelo

```
Something is wrong; all the Accuracy metric values are missing:
    Accuracy       Kappa
 Min.   : NA   Min.   : NA
 1st Qu.: NA   1st Qu.: NA
 Median : NA   Median : NA
 Mean   :NaN   Mean   :NaN
 3rd Qu.: NA   3rd Qu.: NA
 Max.   : NA   Max.   : NA
 NA's   :3     NA's   :3
Error: Stopping
Además: There were 13 warnings (use warnings() to see them)
```

Ilustración 13 Ajuste del modelo KNN.

No es posible entrenar un modelo KNN para la detección de ataques DDOS. Este modelo no es apropiado porque se basa en clasificar una nueva observación de acuerdo con la similitud que tiene con las observaciones del conjunto de entrenamiento, pero al haber tantas observaciones similares entre sí, se produce un error al clasificar. Ya no se considerará en el resto del proceso.

4.6.2. Del ajuste del modelo random forest

4.6.2.1. Configuración del proceso de entrenamiento

```
particiones  <- 2
repeticiones <- 2

hiperparametros <- expand.grid(mtry = c(5, 20, 35))

set.seed(456)
seeds <- vector(mode = "list", length = (particiones * repeticiones) + 1)
for (i in 1:(particiones * repeticiones)) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[(particiones * repeticiones) + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
control_train <- trainControl(method = "repeatedcv", number = particiones,
                              repeats = repeticiones, seeds = seeds,
                              returnResamp = "final", verboseIter = FALSE)
```

Ilustración 14 Configuración del proceso de enXabtrenamiento de Random Forest.

Se utilizaron 2 particiones y 2 repeticiones para la validación cruzada aplicada en el procedimiento de optimización de hiperparámetros. Los valores que se probaron para el hiperparámetro “mtry” fueron: 5, 20 y 35. En la variable “seeds” se almacenaron las semillas necesarias para cada uno de los ajustes del modelo en la validación cruzada.

4.6.2.2. Ajuste del modelo

```
Random Forest

130357 samples
    43 predictor
     2 classes: '1', '0'

No pre-processing
Resampling: Cross-Validated (2 fold, repeated 2 times)
Summary of sample sizes: 65178, 65179, 65178, 65179
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   5    0.9998159  0.9995619
  20    0.9998082  0.9995436
  35    0.9998005  0.9995253

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 5.
```

Ilustración 15 Ajuste del modelo Random Forest.

El modelo que maximiza las métricas en el conjunto de entrenamiento tiene el hiperparámetro mtry = 5.

4.6.3. Del ajuste del modelo XGBoost

4.6.3.1. Configuración del proceso de entrenamiento

```
hiperparametros <- expand.grid(lambda = c(0),
                               alpha = c(0),
                               nrounds = c(50, 100, 200),
                               eta = c(0.3, 0.2, 0.5))

set.seed(456)
seeds <- vector(mode = "list", length = (particiones * repeticiones) + 1)
for (i in 1:(particiones * repeticiones)) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[(particiones * repeticiones) + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
control_train <- trainControl(method = "repeatedcv", number = particiones,
                              repeats = repeticiones, seeds = seeds,
                              returnResamp = "final", verboseIter = FALSE)


set.seed(456)
modelo_xg <- train(response ~ ., data = df_balanced,
                   method = "xgbLinear",
                   tuneGrid = hiperparametros,
                   metric = "Accuracy",
                   trControl = control_train)
```

Ilustración 16 Configuración del proceso de entrenamiento de XGBoost.

Se utilizaron 2 particiones y 2 repeticiones para la validación cruzada aplicada en el procedimiento de optimización de hiperparámetros. Los valores que se probaron para el hiperparámetro “nrounds” fueron: 50, 100 y 200; para “eta” fueron: 0.3, 0.2 y 0.5; se estableció el valor de 0 para los hiperparámetros “lambda” y “alpha”. En la variable “seeds” se almacenaron las semillas necesarias para cada uno de los ajustes del modelo en la validación cruzada.

4.6.3.2. Ajuste del modelo

El modelo que maximiza las métricas en el conjunto de entrenamiento tiene los siguientes hiperparámetros: nrounds = 200, lambda = 0, alpha = 0 and eta = 0.2.

```
eXtreme Gradient Boosting

130357 samples
    43 predictor
     2 classes: '1', '0'

No pre-processing
Resampling: Cross-Validated (2 fold, repeated 2 times)
Summary of sample sizes: 65178, 65179, 65178, 65179
Resampling results across tuning parameters:

  nrounds  eta  Accuracy   Kappa
   50      0.2  0.9997967  0.9995162
   50      0.3  0.9997967  0.9995162
   50      0.5  0.9997967  0.9995162
  100      0.2  0.9998044  0.9995345
  100      0.3  0.9998044  0.9995345
  100      0.5  0.9998044  0.9995345
  200      0.2  0.9998082  0.9995436
  200      0.3  0.9998082  0.9995436
  200      0.5  0.9998082  0.9995436

Tuning parameter 'lambda' was held constant at a value of 0
Tuning parameter 'alpha' was
 held constant at a value of 0
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 200, lambda = 0, alpha = 0 and eta = 0.2.
```

Ilustración 17 Ajuste del modelo XGBoost.

4.7. Evaluación de los modelos de machine learning

Para evaluar la capacidad predictiva de los modelos entrenados y su comportamiento con nuevos datos, se realizan predicciones en el conjunto de prueba y se comparan los valores reales con los predichos.

4.7.1. Resultados de la evaluación del modelo random forest

```
Confusion Matrix and Statistics

          Reference
Prediction       0       1
         0   16771 4728006
         1       0 1364144

               Accuracy : 0.226
                 95% CI : (0.2257, 0.2264)
    No Information Rate : 0.9973
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0016

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.223918
            Specificity : 1.000000
         Pos Pred Value : 1.000000
         Neg Pred Value : 0.003535
             Prevalence : 0.997255
         Detection Rate : 0.223304
   Detection Prevalence : 0.223304
      Balanced Accuracy : 0.611959

       'Positive' Class : 1
```

Ilustración 18 Resultados de la evaluación del modelo Random Forest

La matriz de confusión con sus diferentes métricas nos muestra que el modelo Random Forest es capaz de clasificar muy bien las conexiones benignas. Sin embargo, la mayoría de las conexiones que son realmente parte de un ataque DDOS no se clasifican correctamente, provocando una gran cantidad de falsos negativos y una precisión baja.

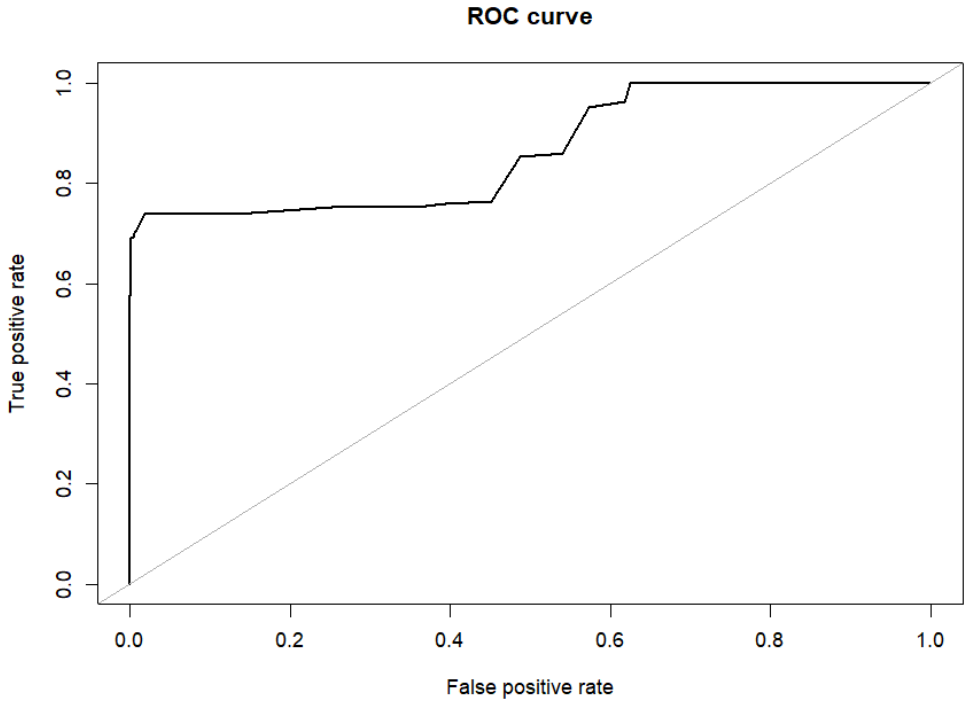


Ilustración 19 Curva ROC para Random Forest.

Área bajo la curva (AUC): 0.866

El área bajo la curva obtenida del modelo Random Forest es un valor alto, lo cual indica que un cambio en el umbral de aceptación para la clase positiva podría causar una mejoría en las predicciones.

4.7.2. Resultados de la evaluación del modelo XGBoost

```
Confusion Matrix and Statistics

          Reference
Prediction       0       1
         0   16771 2862330
         1       0 3229820

               Accuracy : 0.5315
                 95% CI : (0.5311, 0.5318)
    No Information Rate : 0.9973
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0062

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.530161
            Specificity : 1.000000
         Pos Pred Value : 1.000000
         Neg Pred Value : 0.005825
             Prevalence : 0.997255
         Detection Rate : 0.528705
   Detection Prevalence : 0.528705
      Balanced Accuracy : 0.765080

       'Positive' Class : 1
```

Ilustración 20 Resultados de la evaluación del modelo XGBoost.

La matriz de confusión con sus diferentes métricas nos muestra que el modelo XGBoost es capaz de clasificar muy bien las conexiones benignas. Por otro lado, una buena parte de las conexiones que son realmente parte de un ataque DDOS son clasificadas correctamente, obteniéndose una precisión mayor al 50% y mostrando una capacidad predictiva regular.

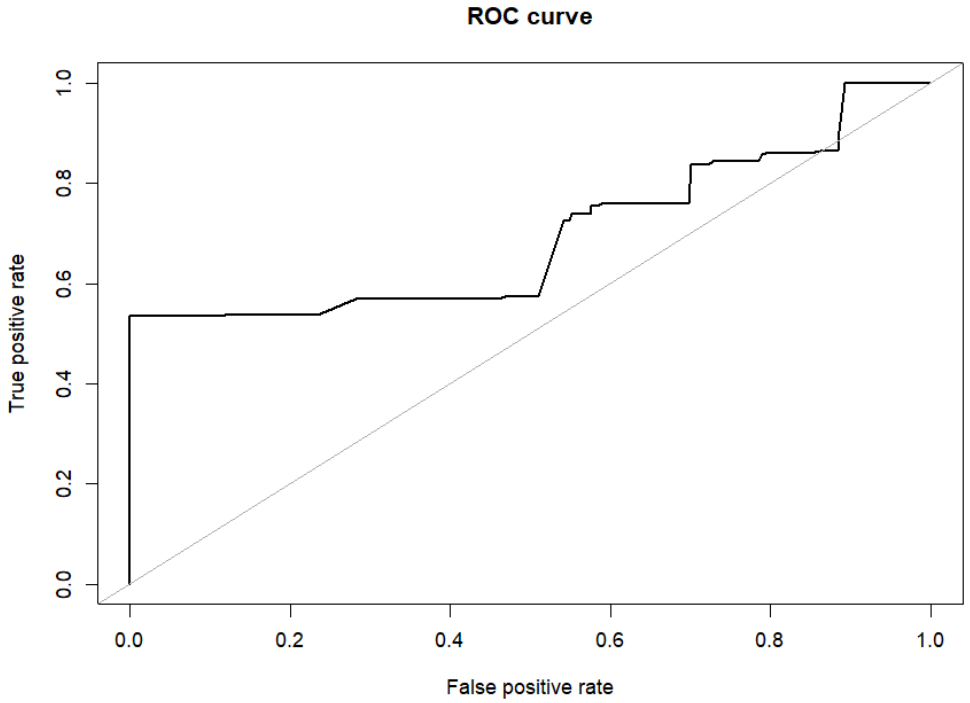


Ilustración 21 Curva ROC para XGBoost.

Área bajo la curva (AUC): 0.694

El área bajo la curva obtenida del modelo XGBoost es un valor regular, lo cual indica que un cambio en el umbral de aceptación para la clase positiva no mejoraría mucho la capacidad predictiva del modelo.

4.8. Selección del modelo

Ya que el modelo Random Forest presentaba un área bajo la curva alto, se cambió el umbral de aceptación hasta mejorar las predicciones obtenidas, y, la matriz de confusión obtenida con un umbral de aceptación de 0.25 es la siguiente:

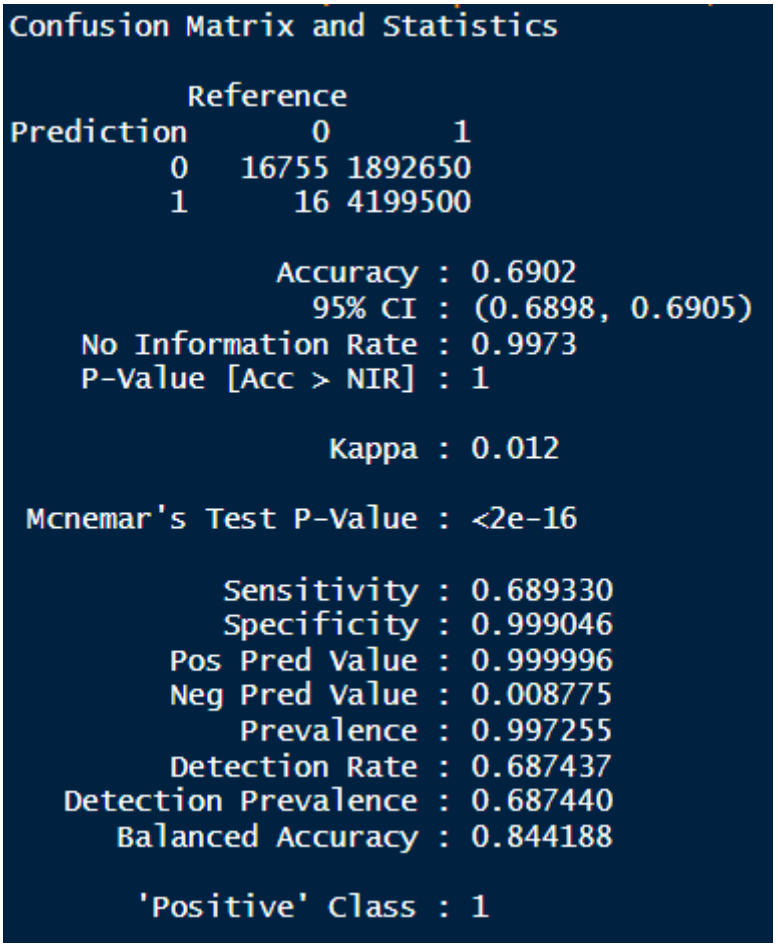


Ilustración 22 Resultados de la evaluación del modelo Random Forest con un 25% de probabilidad.

Previo a la selección, es importante realizar un análisis comparativo de las métricas más relevantes de cada uno de los modelos, así como sus ventajas y desventajas de aplicación.

Tabla 14 Comparación entre las métricas de los modelos de machine learning.

	KNN	Random Forest	XGBoost
Accuracy (Precisión)	No aplica	0.6902	0.5315
Intervalo de confianza del 95% para la precisión	No aplica	(0.6898, 0.6905)	(0.5311, 0.5318)
Tasa de información nula	No aplica	0.9973	0.9973
p-valor [acc > NIR]	No aplica	1.000000	1.000000
Kappa	No aplica	0.012	0.0062
Sensitivity (Sensibilidad)	No aplica	0.689330	0.530161
Specificity (Especificidad)	No aplica	0.999046	1.000000
Positive Predictive Value (Valor Predictivo Positivo)	No aplica	0.999996	1.000000
Negative Predictive Value (Valor Predictivo Negativo)	No aplica	0.008775	0.005825
Prevalence (Prevalencia)	No aplica	0.997255	0.997255
Detection Rate (Tasa de Detección)	No aplica	0.687437	0.528705
Detection Prevalence (Prevalencia de detección)	No aplica	0.687440	0.528705
Balanced Accuracy (precisión balanceada)	No aplica	0.844188	0.765080

Se evidencia que las conexiones benignas se siguen clasificando muy bien. Además, la mayoría de las conexiones que son parte de un ataque DDOS se clasifican correctamente, lo cual provoca una precisión buena y se evidencia un mejor rendimiento respecto a los modelos anteriormente evaluados. El modelo seleccionado es el Random Forest con un umbral de aceptación de 0.25.

Finalmente, con el fin de que los resultados de esta investigación sean replicables, se adjunta el código en R desarrollado en la presente investigación.

4.9. Evaluación del desempeño del producto/servicio de TI

Como parte de las pruebas del modelo de Machine Learning desarrollado, se mantuvo reuniones con los expertos en las áreas que intervienen para el análisis del proyecto.

4.9.1. Evaluación con experto estadístico.

Se realizó la demostración del modelo, de los cuales el entrevistado compartió sus comentarios:

Se confirmó que se siguieron de manera sistemática todos los pasos necesarios para su correcta implementación. Se observó que el código y el script utilizados adherían al framework establecido para el desarrollo de modelos en estudios de esta índole, asegurando así la calidad del modelo.

Durante el proceso, se realizó una importación adecuada de los datos, seguida de un tratamiento y un análisis exploratorio inicial, lo cual permitió una comprensión preliminar de los datos. Se llevaron a cabo transformaciones de variables, incluyendo el tratamiento de aquellas con varianza cercana a cero, las cuales suelen ser poco informativas para los modelos de machine learning debido a su baja variabilidad.

El entrenamiento del modelo se ejecutó correctamente, sin observaciones significativas. Se analizó el rendimiento del modelo a través de métricas y matrices de evaluación, y se realizaron ajustes de hiperparámetros cuando fue necesario.

En términos de implementación, se consideró que el modelo es factible para su implementación, aunque su efectividad podría depender de los procesos y requisitos específicos de cada empresa. Se sugirió que, como propuesta inicial, el modelo podría ser presentado para su posterior aplicación.

El entrevistado destacó que los modelos siempre pueden ser mejorados. Sugirió que, en caso de poner en producción el modelo, se realice un monitoreo periódico del rendimiento, por ejemplo, cada seis meses o anualmente, para asegurar su eficacia continua. Además, se mencionó la posibilidad de incorporar nuevas variables en el futuro, que podrían no estar disponibles en el momento de la implementación inicial, pero que podrían mejorar el modelo una vez obtenidas.

En el caso de empresas con diferentes conjuntos de datos, se indicó que el modelo podría ajustarse en función de las variables disponibles. Se planteó la posibilidad de modificar el modelo o realizar una limpieza de datos para adaptarse a las características específicas de la información de cada empresa. Esto permitiría que el modelo se ajuste a los datos particulares y pueda integrarse eficazmente en el entorno de la empresa.

4.9.2. Calificación del modelo por el experto estadístico.

Se solicitó al evaluado calificar el modelo desarrollado con base a los resultados obtenidos. Mediante la herramienta Google Forms se compartió la encuesta a través de la cual el experto calificó de la siguiente manera:

La escala del 1 al 5 en la rúbrica de calificación se puede interpretar de la siguiente manera: ***1 – Insuficiente; 2 – Deficiente; 3 – Adecuado; 4 – Bueno; 5 – Excelente***.

1. ***Preprocesamiento de Datos:*** *Excelente preprocesamiento, con técnicas avanzadas de manejo de datos y características, como reducción de dimensionalidad.*

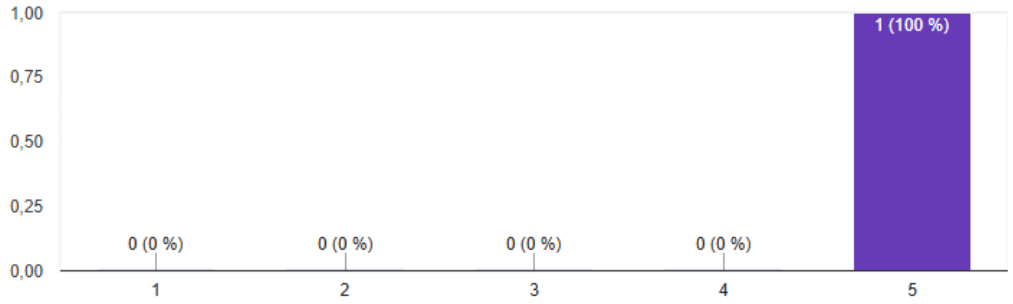


Ilustración 23: Resultado Pregunta 1

2. ***Selección de Modelos:*** *Se realizó una selección de modelos exhaustiva con justificación teórica y empírica para cada uno.*

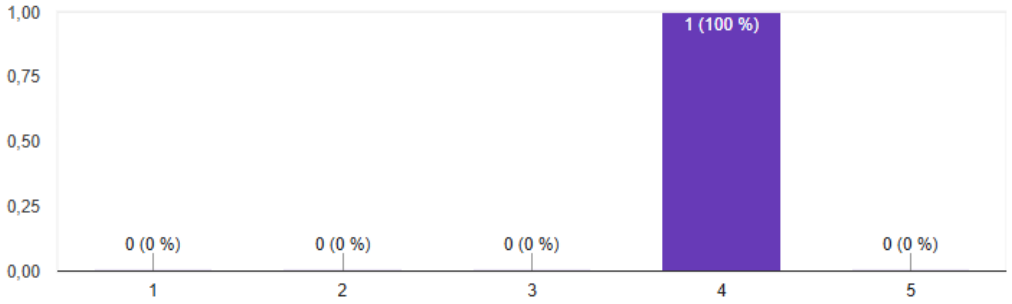


Ilustración 24: Resultado Pregunta 2

3. ***Evaluación del Modelo:*** *Evaluación exhaustiva con interpretación de métricas y análisis de errores.*

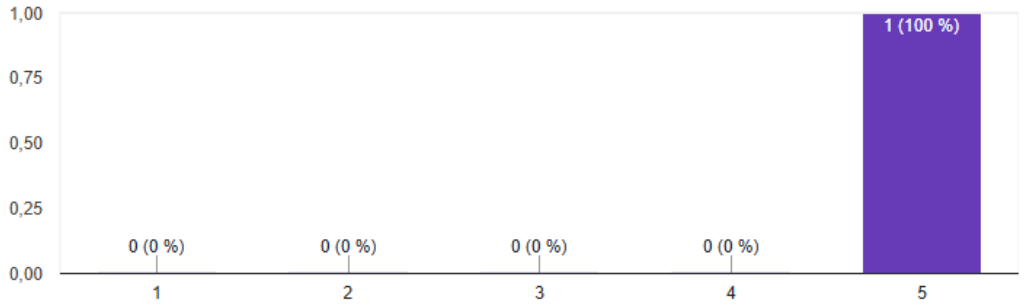


Ilustración 25: Resultado Pregunta 3

4. ***Interpretación de Resultados:*** *Interpretación completa con recomendaciones basadas en los resultados y sugerencias para mejorar el modelo.*

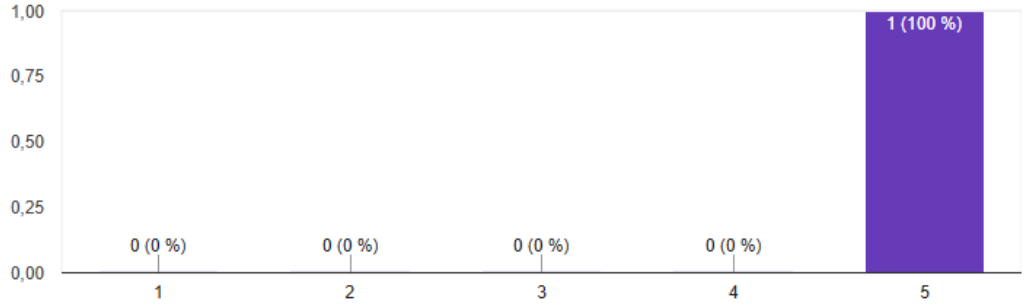


Ilustración 26: Resultado Pregunta 4

5. ***Documentación y Presentación:*** *Excelente documentación y presentación, incluyendo gráficos informativos y conclusiones claras.*

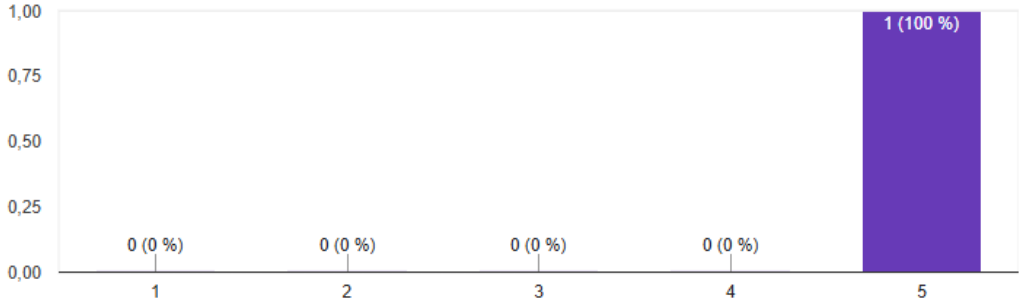


Ilustración 27: Resultado Pregunta 5

4.9.3. Evaluación con experto en redes y seguridades.

Se realizó la demostración del modelo, de los cuales el entrevistado compartió sus comentarios:

Se abordaron aspectos clave sobre el desarrollo y la evaluación del modelo de machine learning. El entrevistado cuestionó la forma en que se mide la efectividad del modelo, específicamente cómo se determina la eficiencia en función de los algoritmos utilizados. Se explicó que la efectividad está determinada por los algoritmos internos de los modelos, que ofrecen métricas específicas basadas en sus propios criterios.

Se identificó un interés por las variables seleccionadas para el modelo y su relevancia en los distintos algoritmos. Además, se comentó sobre el impacto del tamaño de los paquetes de datos durante los ataques de DDoS en la evaluación de la eficiencia del modelo.

Se discutió la importancia de los datos y la necesidad de utilizar un dataset más representativo del entorno real en futuros estudios. Se expuso que el dataset actual era controlado y se basaba en datos desde el inicio hasta el final de un ataque. El entrevistado sugirió que, para una evaluación más precisa, se deberían considerar datos del mundo real en lugar de solo datos simulados.

Finalmente, el entrevistado elogió el análisis y el estudio de lo desarrollado, destacando la importancia de comprender el fondo de los algoritmos de detección para evaluar su efectividad. También mencionó que, en un entorno productivo, se deben considerar múltiples variables y aplicaciones, ya que los ataques de DDoS se dirigen en función del análisis previo del atacante.

El entrevistado concluyó que el estudio cumple con el objetivo de evaluar la efectividad del modelo y que es importante realizar múltiples evaluaciones para asegurar su efectividad en un entorno productivo.

4.9.4. Calificación del modelo por el experto en redes y seguridades.

Se solicitó al evaluado calificar el modelo desarrollado con base a los resultados obtenidos. Mediante la herramienta Google Forms se compartió la encuesta a través de la cual el experto calificó de la siguiente manera:

La escala del 1 al 5 en la rúbrica de calificación se puede interpretar de la siguiente manera: ***1 – Insuficiente; 2 – Deficiente; 3 – Adecuado; 4 – Bueno; 5 – Excelente***.

1. ***Preprocesamiento de Datos:*** *Excelente comprensión de las variables del dataset, usadas en el modelamiento.*

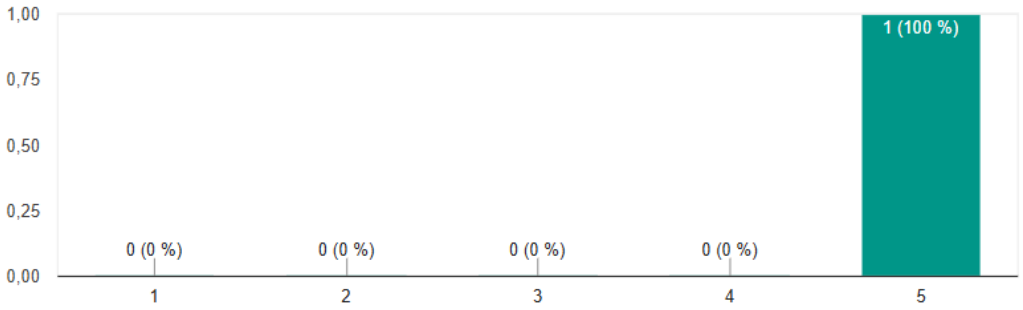


Ilustración 28: Resultado Pregunta 1

2. ***Selección de Modelos:*** *Se realizó una selección adecuada del método de machine learning considerando el contexto de ataques de denegación de servicios distribuidos.*

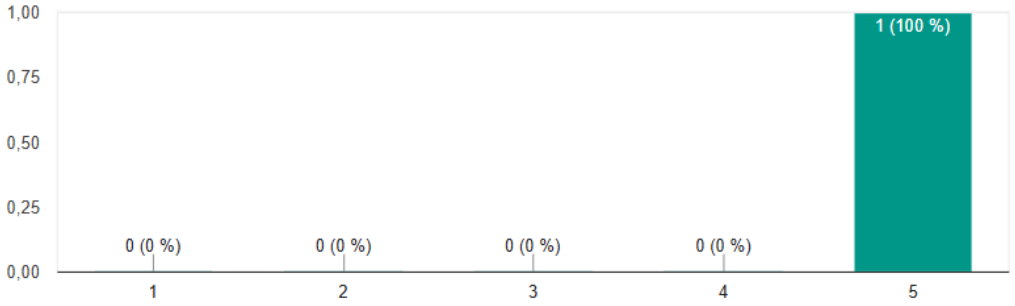


Ilustración 29: Resultado Pregunta 2

3. ***Evaluación del Modelo:*** *La importancia de las variables asignadas por cada modelo se justifican en gran medida a escenarios del mundo real.*

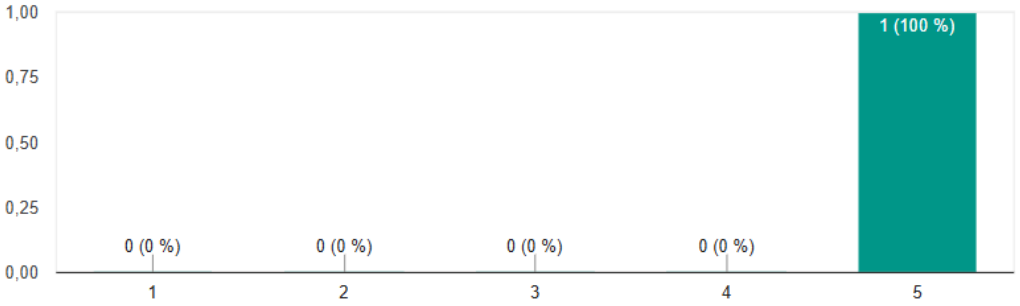


Ilustración 30: Resultado Pregunta 3

4. ***Interpretación de Resultados:*** *Excelente comprensión e interpretación de los resultados obtenidos, sugiere adecuadamente recomendaciones en el marco de los ataques de denegación de servicios distribuidos.*

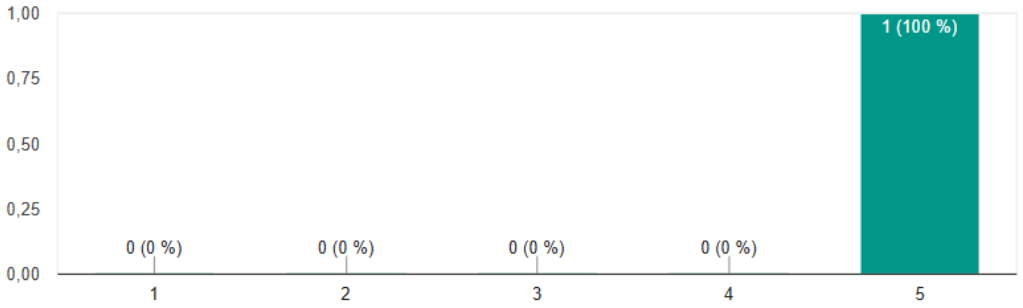


Ilustración 31: Resultado Pregunta 4

5. ***Documentación y Presentación:*** *Excelente documentación y presentación, los gráficos son concluyentes y se relacionan de manera clara con las variables analizadas.*

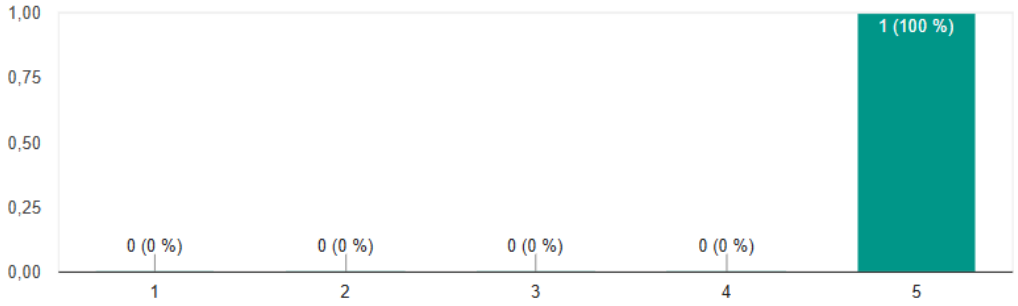


Ilustración 32: Resultado Pregunta 5

5. Conclusiones

El proyecto ha alcanzado de manera satisfactoria sus objetivos al desarrollar un modelo de Machine Learning para la detección de ataques DDoS en entornos IoT. En primer lugar, se realizó una revisión exhaustiva de la literatura científica y los informes técnicos relacionados con la seguridad de la información, enfocándose particularmente en los ataques DDoS. Esta revisión permitió establecer una sólida base teórica y entender las amenazas existentes en detalle.

Posteriormente, se analizó minuciosamente el dataset disponible para identificar patrones y correlaciones relevantes para la detección de ataques. Este análisis fue crucial para entender las características de los datos y seleccionar las variables más significativas que afectan la aparición de ataques DDoS.

Con esta información, se procedió a seleccionar y entrenar los algoritmos de Machine Learning más adecuados, incluyendo KNN, Random Forest y XGBoost. Se dedicó un esfuerzo considerable a la optimización de los hiperparámetros de estos modelos para garantizar su eficacia en la detección de ataques en entornos IoT.

Finalmente, se evaluó la precisión del modelo utilizando diversas métricas, como la matriz de confusión y las curvas ROC. Los resultados confirmaron que el modelo es efectivo y confiable, con una capacidad destacada para identificar correctamente los ataques DDoS y distinguirlos del tráfico legítimo. Sin embargo, el modelo seleccionado, random forest, requirió de un umbral de probabilidad del 0.25 o 25% para mostrar métricas de rendimiento aceptables y, por ende, que evidencien un buen performance en su despliegue en producción.

En conclusión, el proyecto no solo cumplió sus objetivos, sino que también ofreció una solución robusta y eficiente para mejorar la seguridad en entornos IoT. La combinación de una revisión teórica exhaustiva, un análisis detallado de los datos y una rigurosa evaluación del modelo resultó en un enfoque integral que refuerza la protección contra ataques DDoS.

6. Recomendaciones

Para optimizar la eficacia del modelo de detección de ataques DDoS en entornos IoT, se sugieren varias mejoras clave. En primer lugar, se recomienda expandir el conjunto de datos utilizado en el entrenamiento del modelo, incorporando información sobre ataques DDoS recientes y emergentes. Esto permitirá al modelo adaptarse mejor a nuevas variantes de ataques, mejorando su capacidad de detección.

Asimismo, es aconsejable explorar y probar otros algoritmos de Machine Learning. Aunque los modelos de KNN, Random Forest y XGBoost han mostrado buenos resultados, técnicas adicionales como redes neuronales profundas o métodos de aprendizaje de transferencia podrían ofrecer mejoras en precisión y robustez.

La actualización periódica del modelo es otra medida crítica. La naturaleza cambiante de los ataques DDoS requiere que el sistema de detección se ajuste regularmente. Implementar un proceso de retroalimentación continua, en el que el modelo se reentrene con nuevos datos, ayudará a mantener su eficacia a lo largo del tiempo.

En cuanto al despliegue, es fundamental configurar un entorno de ejecución estable que permita la integración del modelo con sistemas de monitoreo y alerta existentes. Utilizar herramientas de integración y entrega continua (CI/CD) facilitará actualizaciones sin interrumpir el sistema. Además, establecer protocolos de seguridad sólidos protegerá el modelo y los datos.

El mantenimiento del modelo también es esencial para su rendimiento continuo. Actualizar el modelo con nuevas técnicas y datos, implementar un sistema de retroalimentación para ajustes y monitorear el rendimiento con alertas automáticas son prácticas recomendadas.

Finalmente, el backtesting del modelo es crucial para evaluar su capacidad en condiciones similares a las reales. Realizar pruebas exhaustivas con datos históricos y simulaciones permitirá identificar áreas de mejora y ajustar el modelo antes de su despliegue completo. En resumen, la ampliación de datos, la exploración de nuevos

algoritmos, la actualización periódica, el despliegue meticuloso y el backtesting riguroso son fundamentales para fortalecer la capacidad del sistema de detección de ataques DDoS y mejorar la seguridad en entornos IoT.

7. Bibliografía

Akamai. (2023). *¿Qué es un ataque DDoS?* Obtenido de Akamai: https://www.akamai.com/es/glossary/what-is-ddos#:~:text=Significado%20de%20ataque%20DDoS,que%20no%20pueda%20funcionar%20correctamente.

Aveleira, J., Muñoz, Á., García, M., Benavides, C., Benítez, J., & Alaiz, V. (2020). Prototipo de IDS para detección de intrusiones con modelos de machine learning en sistemas IoT de la Industria 4.0. *Instituto Nacional de Ciberseguridad (INCIBE)*, 271.

AWS. (2 de agosto de 2016). *Amazon Machine Learning: Guía para desarrolladores*. Obtenido de docs.aws.amazon.com: https://docs.aws.amazon.com/es_es/machine-learning/latest/dg/cross-validation.html

AWS. (2023). *¿Qué es el aprendizaje automático?* Obtenido de Amazon Web Services: https://aws.amazon.com/es/what-is/machine-learning/

AWS. (2024). *¿Qué son los hiperparámetros?* Obtenido de AWS: https://aws.amazon.com/es/what-is/hyperparameter-tuning/

BitLife. (23 de Enero de 2023). *Ciberseguridad para pymes: las 5 principales amenazas a las que se enfrentan*. Obtenido de Bit Life Media: https://bitlifemedia.com/2023/01/pymes-ciberseguridad-amenazas/

Brownlee, J. (17 de Febrero de 2021). *A Gentle Introduction to XGBoost for Applied Machine Learning*. Obtenido de Machine Learning Marstery: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Cáceres, D. (1 de mayo de 2023). *Datasets: Qué son y cómo acceder a ellos*. Obtenido de OpenWebinars: https://openwebinars.net/blog/datasets-que-son-y-como-acceder-a-ellos/

EASYDMARC. (9 de julio de 2022). *¿Cómo detectar los ataques DDoS?* Obtenido de easydmarc.com: https://easydmarc.com/blog/es/como-detectar-los-ataques-ddos/

Faludi, R. (26 de Marzo de 2021). *¿Cómo se comunican los dispositivos de IoT ?* Obtenido de DIGI. Connect with Confidence: https://es.digi.com/blog/post/how-do-iot-devices-communicate

Fuentes, J., & Sebastían, M. (2022). *Análisis comparativo de la herramienta tablero en las metodologías ágiles Scrum, Kanban y Scrumban en proyectos de software.* International Project Management Association. Obtenido de http://dspace.aeipro.com/xmlui/handle/123456789/3286

García, M. (2023). *IoT - Internet Of Things*. Obtenido de Deloitte: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

Greenberg, A. (2018). The Untold Story of Notpetya, the most devasting cyberatack in history. *WIRED*, 11-12.

Greenberg, A. (2022 de Agosto de 2018). THEUNTOLD STORY OFNOTPETYA,THE MOST DEVASTATINGCYBERATTACKIN HISTORY. *THEUNTOLD STORY OFNOTPETYA,THE MOST DEVASTATINGCYBERATTACKIN HISTORY*. Obtenido de https://cyber-peace.org/wp-content/uploads/2018/10/The-Untold-Story-of-NotPetya-the-Most-Devastating-Cyberattack-in-History-_-WIRED.pdf

IBM. (17 de Agosto de 2021). *Funcionamiento de SVM*. Obtenido de IBM Documentación: https://www.ibm.com/docs/es/spss-modeler/saas?topic=models-how-svm-works

IBM. (2021). *Machine learning vs. deep learning vs. redes neuronales*. Obtenido de www.ibm.com: https://www.ibm.com/es-es/topics/machine-learning

IBM. (2023). *¿Qué es KNN?* Obtenido de www.ibm.com: https://www.ibm.com/mx-es/topics/knn

IBM. (2023). *¿Qué es un bosque aleatorio?* Obtenido de www.ibm.com: https://www.ibm.com/es-es/topics/random-forest

Kaspersky. (2023). *¿Qué son los ataques DDoS?* Obtenido de latam.kaspersky.com: https://latam.kaspersky.com/resource-center/threats/ddos-attacks

kaspersky. (25 de septiembre de 2023). *Kaspersky publica resumen general de las amenazas relacionadas con IoT en 2023*. Obtenido de latam.kaspersky.com: https://latam.kaspersky.com/about/press-releases/2023_kaspersky-publica-resumen-general-de-las-amenazas-relacionadas-con-iot-en-2023

Kaspersky. (2023). *La IA, el machine learning y la definición del aprendizaje profundo en la ciberseguridad*. Obtenido de La IA, el machine learning y la definición del aprendizaje profundo en la ciberseguridad: https://www.kaspersky.es/resource-center/definitions/ai-cybersecurity

Kolias, C. K. (2017). DDoS in the IoT: Mirai and other botnets . *Computer*, 50(7), 80-84.

Llugsa. (2022). *Prototipo basado en internet de las cosas (IoT) para el monitoreo de invernaderos.* Obtenido de https://repositorio.puce.edu.ec/handle/123456789/7139

Microsoft. (2024). *Protocolos y tecnologías de IoT*. Obtenido de azure.microsoft.com: https://azure.microsoft.com/es-mx/solutions/iot/iot-technology-protocols

Montes, J., Icaza, G., & Duque, N. (2023). Efecto de la selección de atributos en el desempeño de un IDS basado en machine learning para detección de intrusos en ataques DDoS. *South Florida Journal of Development*. Obtenido de https://ojs.southfloridapublishing.com/ojs/index.php/jdev/article/view/2510

Peneti, & Hemalatha. (Enero de 2021). DDOS Attack Identification using Machine Learning Techniques. *2021 International Conference on Computer Communication and Informatics, ICCCI 2021.*, 4 - 8. Coimbatore, India.

Polanía, L. (2021). *Evaluación de modelos de Machine Learning para Sistemas de Detección de Intrusos en redes IoT.* Bogotá: Universidad de los Andres.

Recuerdo, P. (2 de diciembre de 2021). *Tipos de aprendizaje en Machine Learning: supervisado y no supervisado*. Obtenido de Telefónica Tech: https://telefonicatech.com/blog/que-algoritmo-elegir-en-ml-aprendizaje

Red Hat. (20 de Enero de 2023). *¿Qué es el Internet de las cosas (IoT)?* Obtenido de Red Hat: https://www.redhat.com/es/topics/internet-of-things/what-is-iot

RedHat. (20 de Enero de 2023). *¿Qué es el Internet de las cosas (IoT)?* Obtenido de Red Hat: https://www.redhat.com/es/topics/internet-of-things/what-is-iot

Salazar, J., & Silvestre, S. (2016). *Internet de las cosas.*

Stackscale. (1 de 2 de 2022). *Ataques DDoS: tipos, protección y mitigación*. Obtenido de Ataques DDoS: tipos, protección y mitigación: https://www.stackscale.com/es/blog/ataques-ddos/#:~:text=Los%20ataques%20DDoS%20se%20pueden,inundaci%C3%B3n%20SYN%20e%20inundaci%C3%B3n%20UDP.

Vargas, A. (2021). *Sistema de Detección de Intrusos Basado en Machine Learning.* Ambato: Universidad Técnica de Ambato.

Vargas, O. (2023). *Detección de Amenazas en Redes IoT Empleando Modelos Híbridos de Machine Learning y Redes Neuronales.* Obtenido de Repositorio Institucional Séneca:

https://repositorio.uniandes.edu.co/server/api/core/bitstreams/e4d0cea6-f9a5-4ca0-a37d-d090bede7c1f/content

WeLiveSecurity. (7 de Febrero de 2024). *Qué es un ataque DDoS y cuáles son sus consecuencias*. Obtenido de WeLiveSecurity: https://www.welivesecurity.com/es/seguridad-corporativa/que-ataque-ddos-consecuencias/

Worsley, S. (julio de 2024). *¿Qué es R? Introducción a la potencia del cálculo estadístico*. Obtenido de DataCamp: https://www.datacamp.com/es/blog/all-about-r

8. ANEXOS

8.1. Elaboración de actividades usando metodología kanban.

Tabla 15 Tabla de actividades durante la elaboración del proyecto. Equipo de trabajo

Pendiente	En Proceso	En Revisión	Completado
Modelamiento de datos utilizando los algoritmos Random forest, KNN, XGBoost	Análisis exploratorio de modelo ML	Resultados de la entrevista a experto estadístico.	Revisión de literatura científica e informes técnicos relacionados con la seguridad de la información y ataques DDoS.
Entrenamiento (validación cruzada y optimización de hiperparámetros)	Entrevista a experto en redes y seguridades		Recolección y selección de datos para entrenar el modelo.
Selección del modelo más eficaz.			Selección de herramientas y tecnologías necesarias (RStudio, librerías).
			Lectura, preprocesamiento y transformación de datos

Tabla 16 Tabla de actividades finalizadas. Equipo de trabajo

Pendiente	En Proceso	En Revisión	Completado
No aplica	No aplica	No aplica	Revisión de literatura científica e informes técnicos relacionados con la seguridad de la información y ataques DDoS.
No aplica	No aplica	No aplica	Recolección y selección de datos para entrenar el modelo.
No aplica	No aplica	No aplica	Selección de herramientas y tecnologías necesarias (RStudio, librerías).
No aplica	No aplica	No aplica	Lectura, preprocesamiento y transformación de datos
No aplica	No aplica	No aplica	Análisis Exploratorio de modelo ML

No aplica	No aplica	No aplica	Modelamiento de datos utilizando los algoritmos Random forest, KNN, XGBoost
No aplica	No aplica	No aplica	Entrenamiento (validación cruzada y optimización de hiperparámetros)
No aplica	No aplica	No aplica	Selección del modelo más eficaz.
No aplica	No aplica	No aplica	Entrevista a experto en redes y seguridades
No aplica	No aplica	No aplica	Entrevista a experto estadístico

8.2. Proceso para el desarrollo del modelo ML

Elaborado en Lucidchart por el grupo de trabajo. Enlace:

https://lucid.app/lucidchart/adeb74c1-a12a-4388-818d-811eaaa0c5cd/edit?invitationId=inv_47d7fe3b-712c-4825-ae24-1c3465535503

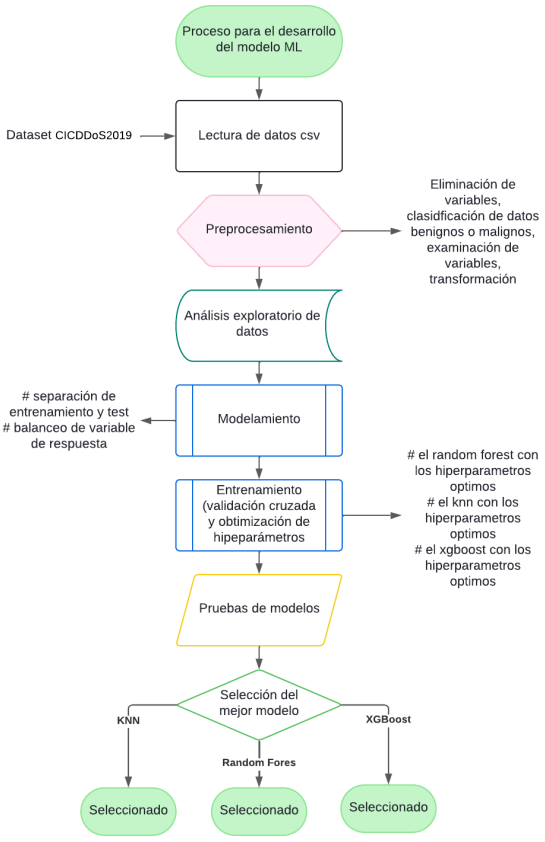


Ilustración 33: Proceso para desarrollar modelo de Machine Learning. Elaborado por el grupo de trabajo

8.3. Cuadro comparativo para la selección del dataset.

En la tabla 17 se resume las características relevantes de aquellos dataset que formaron parte de la investigación y decisión del conjunto de datos adecuado:

Tabla 17 Cuadro comparativo para la selección del Dataset. Elaborado por el grupo de trabajo

	Dataset 1	Dataset 2	Dataset 3
Fuente u origen	University Of New Brunswick	IEEE DataPort	IEEE DataPort
Nombre Dataset	CICDDoS2019	DDOS Attack Dataset	IEEE P2668-Compliant Multi-Layer IoT-DDoS Dataset
Numero de subdatasets	7 dataset segmentaddos por tipo de ataque DDoS	No aplica	6 dataset segmentados por protocolos
Número total de variables	88 variables	61 variables	6 variables, dependencia del tipo de protocolo añadiendo información adicional
Número total de registros	20.364.525	400.000	22.984.509
Descripción general	Contiene ataques benignos y los más actualizados y comunes ataques DDoS, que se asemejan a los verdaderos datos del mundo real (PCAPs)	Este conjunto de datos de ataques DDoS se puede utilizar para evaluar el rendimiento de clasificadores de aprendizaje automático y modelos de aprendizaje profundo.	Se compone de numerosos tipos de ataques DDoS generados a partir de múltiples protocolos líderes aplicados en IoT.
Desventajas	● Puede ser demasiado grande para ciertos tipos de análisis rápidos. ● Necesita almacenamiento considerable debido a su tamaño.	● Menor cantidad de registros que puede no representar adecuadamente todas las variaciones posibles de DDoS.	● Solo 6 variables, limitando el análisis detallado. ● Centrado únicamente en ataques IoT, lo que puede

		● No está segmentado, lo que puede dificultar el análisis específico	sesgar el análisis debido al enfoque específico de la investigación
Ventajas	● Alta variedad de variables, permitiendo un análisis más detallado. ● Segmentación por tipo de ataque facilita análisis específicos. ● Gran número de registros que asegura una representación completa de escenarios de ataque.	● Fácil de manejar debido a su tamaño más pequeño. ● Adecuado para pruebas rápidas de modelos de aprendizaje automático.	● Relevante para IoT, que es una tendencia creciente.

8.4. Desarrollo del modelo en RStudio.

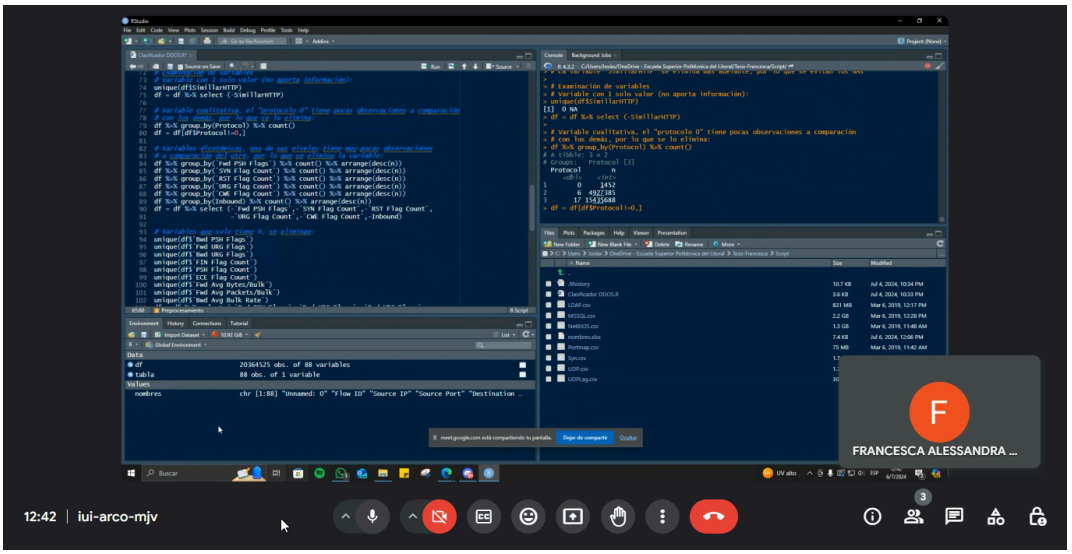


Ilustración 34 Desarrollo de los modelos en R Studio.

8.5. Perfiles y entrevistas a expertos

8.5.1. Experto estadístico

Perfil

Nombre y apellidos: Christian Javier Salas Márquez

Título profesional: Ingeniero Estadístico - ESPOL

Función: Data Analyst

Certificaciones:

- Supervised Learning with scikit-learn
- Data Analyst with Python
- Business Inteliggence con Power BI Avanzado
- Analítica y Modelado de Datos con SQL
- Reporting with R Markdown

Entrevista

Lugar, fecha y hora: Guayaquil, 7 de julio del 2024, 14h00.

Forma: Vía Google Meet

Tiempo de entrevista: 45 minutos

Entrevistador: Francesca Alessandra Calle Mera

Preguntas

1. ¿Puede contarnos sobre su trayectoria profesional como analista de datos y su experiencia con R y RStudio?

   El ingeniero estadístico comenzó su trayectoria profesional como analista de datos en GrandSouth SA, una exportadora de cacao, donde se trabajó en la preparación y análisis de bases de datos utilizando R y RStudio. Además, fue ayudante de docencia en cursos relacionados con datos categóricos, ingeniería de datos y controles estadísticos de procesos.

   En su carrera, ha trabajado extensamente con bases de datos almacenadas en SQL y ha desarrollado modelos predictivos, realizando pruebas de diferentes algoritmos. También ha trabajado en la creación de modelos tradicionales para predecir la tasa de deserción escolar y el índice de robos.

2. ¿Qué consideraciones son esenciales al realizar ingeniería de datos para garantizar la calidad y relevancia de los datos?

   Al realizar ingeniería de datos, considera crucial comenzar con un análisis exploratorio de los datos. Esto implica identificar y manejar variables faltantes y asegurarse de que los datos sean consistentes y mantengan su integridad. La explotación adecuada de los datos permite identificar cualquier inconsistencia y garantizar que los datos utilizados sean relevantes y de alta calidad.

3. ¿Cómo implementa controles estadísticos de procesos en sus análisis y qué herramientas utiliza para ello?

   Para implementar controles estadísticos de procesos, utiliza librerías como Qcc y herramientas de control estadístico de procesos (STC). Estas

herramientas le permiten monitorear y controlar la calidad de los procesos mediante la evaluación de la variabilidad y el comportamiento del sistema.

4. ¿Puede describir su proceso para desarrollar y probar modelos de detección en R y RStudio?

El proceso para desarrollar y probar modelos de detección en R y RStudio, los definión en los siguientes pasos:

a. Definir el problema: Identificar claramente lo que se busca predecir.
b. Obtener el conjunto de datos: Recolectar los datos necesarios para el análisis.
c. Exploración de los datos: Realizar un análisis exploratorio para entender las características y la estructura de los datos.
d. Selección de características: Elegir las variables más relevantes para el modelo.
e. Desarrollo de modelos: Aplicar técnicas de machine learning para construir el modelo.
f. Optimización de hiperparámetros: Optimizar los parámetros del modelo para mejorar su desempeño.
g. Implementación en entorno de producción: Desplegar el modelo en un entorno de producción para su uso real.

5. ¿Cuáles son las métricas más importantes que utiliza para validar la precisión y efectividad de sus modelos de detección?

Las métricas más importantes que utiliza para validar la precisión y efectividad de los modelos de detección incluyen:

a. Precisión.
b. Sensibilidad.
c. F1 Score.

d. AUC (Área bajo la curva) o ROC (Curva de características operativas del receptor)
e. Matriz de confusión: Permite evaluar el desempeño del modelo al comparar las predicciones con los resultados reales.

6. ¿Qué desafíos ha enfrentado al validar modelos de clasificación y cómo los ha superado?

Uno de los desafíos principales al validar modelos de clasificación, es el desequilibrio en el conjunto de datos. Para superar este desafío, utiliza técnicas de sobremuestreo y verificación. Además, la variabilidad de los datos puede ser un problema, por lo que emplea valores SHAP para explicar modelos complejos y entender mejor sus predicciones.

7. ¿Cómo integra el machine learning en sus proyectos de análisis de datos y qué algoritmos ha encontrado más efectivos para la detección?

Integra machine learning en sus proyectos de análisis de datos a través de la validación cruzada y la división adecuada de los datos en conjuntos de entrenamiento y prueba. Utiliza Shiny para desarrollar aplicaciones interactivas en R, lo que facilita la visualización y el análisis de los resultados. Entre los algoritmos más efectivos para la detección, ha encontrado que XGBoost y Random Forest son especialmente útiles, ya que ofrecen alta precisión y robustez en grandes conjuntos de datos.

8. *En su experiencia, ¿Qué tipos de algoritmos considera más convenientes para desarrollar un modelo de detección en Machine Learning, siendo este orientado a los ataques DDoS (denegación de servicios distribuidos) en entorno de internet de las cosas?*

Para desarrollar modelos de detección de ataques DDoS en entornos IoT, considera que los algoritmos más convenientes son XGBoost y Random Forest. Estos algoritmos son eficaces para manejar grandes conjuntos de

datos y mejorar la precisión de las predicciones. Además, es esencial realizar una revisión bibliográfica exhaustiva para seleccionar los algoritmos más adecuados y adaptarlos a las características específicas del entorno IoT y del problema de detección de ataques DDoS.

Demostración del desarrollo del Modelo de Machine Learning en RStudio

Durante la entrevista, se demostró el desarrollo del modelo hasta la fase de análisis exploratorio de los datos. El ingeniero estadístico afirmó que se está llevando a cabo un preprocesamiento adecuado, asegurando la integridad de los datos para el modelamiento. Además, destacó el uso óptimo de RStudio, una herramienta de código abierto que permite su uso libremente. También mencionó que la adopción de estos modelos y herramientas dependerá del tipo de empresa, ya que algunos comercios requerirán mayor escalabilidad y seguridad, lo que implicará inversiones en actualizaciones.



Ilustración 35 Reunión Vía Google Meet con el experto en Estadística.

8.5.2. Experto en redes y seguridades.

Perfil

Nombre y apellidos: Joffre David Marcillo

Título profesional: Ingeniero Electrónica y Telecomunicaciones – ESPOL

Función: Network and Security Engineer

Certificaciones:

- Certified Ethical Hacker - EC Council
- CCNP: Route, Switch
- CCNA 1: Network Fundamentals
- CCNA 2: Routing Protocols & Concepts
- CCNA 3: LAN Switching & Wireless
- CCNA Cisco Certified

Entrevista

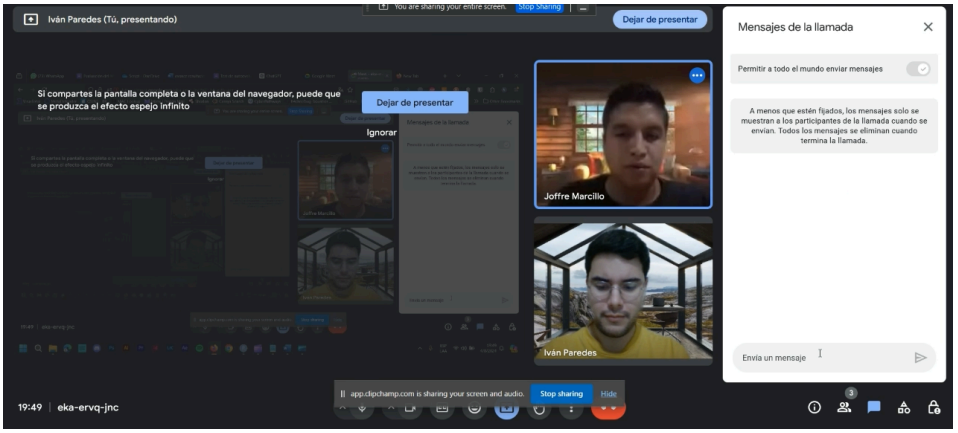


Ilustración 36 Reunión Vía Google Meet con el experto en redes y seguridad.

8.6. Código de R para el procesamiento y análisis de los datos.

```
library(readr)
library(tidyverse)
library(caret)
library(performanceEstimation)
library(ROSE)
library(openxlsx)
options(scipen = 999)

# Lectura de datos ----

ldap <- read_csv("LDAP.csv")
mssql <- read_csv("MSSQL.csv")

df = rbind(ldap,mssql)
rm(ldap,mssql)

netbios <- read_csv("NetBIOS.csv")

df = rbind(df,netbios)
rm(netbios)
```

```
portmap <- read_csv("Portmap.csv")

df = rbind(df,portmap)
rm(portmap)

syn <- read_csv("Syn.csv")

df = rbind(df,syn)
rm(syn)

udp <- read_csv("UDP.csv")

df = rbind(df,udp)
rm(udp)

udplag <- read_csv("UDPLag.csv")

df = rbind(df,udplag)
rm(udplag)

# Info dataset ----

glimpse(df)

unique(df$Protocol)
unique(df$Label)
unique(df$SimillarHTTP)
unique(df$Inbound)

nrow(df)

sum(is.na(df))

df %>% group_by(Inbound) %>% count() %>% arrange(desc(n))

# Preprocesamiento ----

# Se quitan variables no utilizadas
df = df %>% select(-`Unnamed: 0`,-`Flow ID`,-`Source IP`,-`Source Port`,-`Destination IP`,
          -`Destination Port`,-Timestamp,-`Flow Bytes/s`,-`Flow Packets/s`)

# Se coloca la etiqueta que indica si es benigno (0) o maligno (1)
df = df %>% mutate(response = ifelse(Label=="BENIGN", 0, 1))
df = df %>% select(-Label)

# Examinación de valores ausentes
val_aus = data.frame(map_dbl(df, .f = function(x){sum(is.na(x))}))
 # La variable "SimillarHTTP" se elimina más adelante, por lo que se evitan los NAs
```

```
# Examinación de variables
# Variable con 1 solo valor (no aporta información):
unique(df$SimillarHTTP)
df = df %>% select (-SimillarHTTP)

# Variable cualitativa, el "protocolo 0" tiene pocas observaciones a comparación
# con los demás, por lo que se lo elimina:
df %>% group_by(Protocol) %>% count()
df = df[df$Protocol!=0,]

# Variables dicotómicas, uno de sus niveles tiene muy pocas observaciones
# a comparación del otro, por lo que se elimina la variable:
df %>% group_by(`Fwd PSH Flags`) %>% count() %>% arrange(desc(n))
df %>% group_by(`SYN Flag Count`) %>% count() %>% arrange(desc(n))
df %>% group_by(`RST Flag Count`) %>% count() %>% arrange(desc(n))
df %>% group_by(`URG Flag Count`) %>% count() %>% arrange(desc(n))
df %>% group_by(`CWE Flag Count`) %>% count() %>% arrange(desc(n))
df %>% group_by(Inbound) %>% count() %>% arrange(desc(n))
df = df %>% select (-`Fwd PSH Flags`,-`SYN Flag Count`,-`RST Flag Count`,
                    -`URG Flag Count`,-`CWE Flag Count`,-Inbound)

# Variables que solo tiene 0, se eliminan:
unique(df$`Bwd PSH Flags`)
unique(df$`Fwd URG Flags`)
unique(df$`Bwd URG Flags`)
unique(df$`FIN Flag Count`)
unique(df$`PSH Flag Count`)
unique(df$`ECE Flag Count`)
unique(df$`Fwd Avg Bytes/Bulk`)
unique(df$`Fwd Avg Packets/Bulk`)
unique(df$`Bwd Avg Bulk Rate`)
unique(df$`Fwd Avg Bulk Rate`)
unique(df$`Bwd Avg Bytes/Bulk`)
unique(df$`Bwd Avg Packets/Bulk`)
df = df %>% select (-`Bwd PSH Flags`,-`Fwd URG Flags`,-`Bwd URG Flags`,
                    -`FIN Flag Count`,-`PSH Flag Count`,-`ECE Flag Count`,
                    -`Fwd Avg Bytes/Bulk`,-`Fwd Avg Packets/Bulk`,-`Bwd Avg Bulk Rate`,
                    -`Fwd Avg Bulk Rate`,-`Bwd Avg Bytes/Bulk`,-`Bwd Avg Packets/Bulk`)


# Transformación de variables
df$response = as.factor(df$response)
df$Protocol = as.factor(df$Protocol)


# Correlación entre variables continuas

correlaciones = data.frame(cor(df %>% select (-response,-Protocol)))

nombres_cor = c()
```

```
for (i in 1:nrow(correlaciones)) {
  nombres_temp = colnames(correlaciones)[(correlaciones[i,]>0.8 & correlaciones[i,]!=1)]
  nombres_cor = c(nombres_cor,nombres_temp)
}
nombres_cor = unique(nombres_cor)
# Hay muchas variables con fuerte correlacion porque son combinaciones entre ellas


# Variables con varianza próxima a cero

varianzas = df %>% select(-response,-Protocol) %>%
  nearZeroVar(saveMetrics = TRUE)


# Se eliminan las variables con varianza cercana a 0
variables_nzv = rownames(varianzas %>% filter(nzv==T))

df = df %>% select(-all_of(variables_nzv))


# La estandarización de las variables se aplica posterior a los gráficos descriptivos
# para mantener los valores reales de las variables.

# Análisis exploratorio ----

# Variable de respuesta

# Calcula las frecuencias absolutas
absolute_frequencies <- table(df$response)

# Calcula las frecuencias relativas
relative_frequencies <- prop.table(absolute_frequencies)

# Combina ambas tablas en un dataframe
frequency_table <- data.frame(
  Respuesta = names(absolute_frequencies),
  Frecuencia_absoluta = as.vector(absolute_frequencies),
  Frecuencia_relativa = as.vector(relative_frequencies)
)


ggplot(data = df, aes(x = response, y = after_stat(count), fill = response)) +
  geom_bar() +
  scale_fill_manual(values = c("#00BAEA", "#61858F"), labels = c("Benigno", "DDOS")) +
  labs(title = "Variable de respuesta", x = "Respuesta", y = "Frecuencia", fill = "") +
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "#035BA9"),
    legend.position = "bottom",
```

```
    legend.title = element_blank()
  )

# Variables cualitativas

# Calcula las frecuencias absolutas
absolute_frequencies <- table(df$Protocol)

# Calcula las frecuencias relativas
relative_frequencies <- prop.table(absolute_frequencies)

# Combina ambas tablas en un dataframe
frequency_table <- data.frame(
  Protocolo = names(absolute_frequencies),
  Frecuencia_absoluta = as.vector(absolute_frequencies),
  Frecuencia_relativa = as.vector(relative_frequencies)
)


ggplot(data = df, aes(x = Protocol, y = after_stat(count), fill = Protocol)) +
  geom_bar() +
  scale_fill_manual(values = c("#00BAEA", "#61858F"), labels = c("No ataque", "Ataque")) +
  labs(title = "Variable cualitativa Protocolo", x = "Protocolo", y = "Frecuencia", fill = "") +
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "#035BA9"),
    legend.position = "none"  # Elimina la leyenda
  )

# Variables cuantitativas

# Calcula las estadísticas resumidas para la variable `Min Packet Length`
summary_stats <- df %>%
  summarise(
    Min = min(`Min Packet Length`),
    Q1 = quantile(`Min Packet Length`, 0.25),
    Mediana = median(`Min Packet Length`),
    Media = mean(`Min Packet Length`),
    Q3 = quantile(`Min Packet Length`, 0.75),
    Max = max(`Min Packet Length`),
    Varianza = var(`Min Packet Length`),
    SD = sd(`Min Packet Length`)
  )



ggplot(data = df, aes(x = `Min Packet Length`, fill = factor(response))) +
  geom_density(alpha = 0.6) +
  scale_fill_manual(values = c("#00BAEA", "#61858F"), labels = c("Benigno", "DDOS")) +
```

```
  labs(title = "Distribución de Min Packet Length por Respuesta", x = "Min Packet Length", y =
"Densidad") +
  theme_light() +
  theme(
   plot.title = element_text(hjust = 0.5, color = "#035BA9"),  # Centra el título y cambia su
color
   legend.position = "bottom",  # Muestra la leyenda en la parte inferior
   legend.title = element_blank()  # Elimina el título de la leyenda
  )



 # Calcula las estadísticas resumidas para la variable `Fwd Packet Length Min`
 summary_stats <- df %>%
  summarise(
   Min = min(`Fwd Packet Length Min`),
   Q1 = quantile(`Fwd Packet Length Min`, 0.25),
   Mediana = median(`Fwd Packet Length Min`),
   Media = mean(`Fwd Packet Length Min`),
   Q3 = quantile(`Fwd Packet Length Min`, 0.75),
   Max = max(`Fwd Packet Length Min`),
   Varianza = var(`Fwd Packet Length Min`),
   SD = sd(`Fwd Packet Length Min`)
  )


 ggplot(data = df, aes(x = `Fwd Packet Length Min`, fill = factor(response))) +
  geom_density(alpha = 0.6) +
  scale_fill_manual(values = c("#00BAEA", "#61858F"), labels = c("Benigno", "DDOS")) +
  labs(title = "Distribución de Fwd Packet Length Min por Respuesta", x = "Fwd Packet Length
Min", y = "Densidad") +
  theme_light() +
  theme(
   plot.title = element_text(hjust = 0.5, color = "#035BA9"),  # Centra el título y cambia su
color
   legend.position = "bottom",  # Muestra la leyenda en la parte inferior
   legend.title = element_blank()  # Elimina el título de la leyenda
  )



 # Calcula las estadísticas resumidas para la variable `Avg Fwd Segment Size`
 summary_stats <- df %>%
  summarise(
   Min = min(`Avg Fwd Segment Size`),
   Q1 = quantile(`Avg Fwd Segment Size`, 0.25),
   Mediana = median(`Avg Fwd Segment Size`),
   Media = mean(`Avg Fwd Segment Size`),
   Q3 = quantile(`Avg Fwd Segment Size`, 0.75),
   Max = max(`Avg Fwd Segment Size`),
```

```
    Varianza = var(`Avg Fwd Segment Size`),
    SD = sd(`Avg Fwd Segment Size`)
  )



ggplot(data = df, aes(x = `Avg Fwd Segment Size`, fill = factor(response))) +
  geom_density(alpha = 0.6) +
  scale_fill_manual(values = c("#00BAEA", "#61858F"), labels = c("Benigno", "DDOS")) +
  labs(title = "Distribución de Avg Fwd Segment Size por Respuesta", x = "Avg Fwd Segment
Size", y = "Densidad") +
  theme_light() +
  theme(
    plot.title = element_text(hjust = 0.5, color = "#035BA9"),  # Centra el título y cambia su
color
    legend.position = "bottom",  # Muestra la leyenda en la parte inferior
    legend.title = element_blank()  # Elimina el título de la leyenda
  )

# Ajustes previos al entrenamiento ----

# Separación de dataset de entrenamiento y test
set.seed(456)
train <- createDataPartition(y = df$response, p = 0.7, list = FALSE, times = 1)
df_train <- df[train, ]
df_test  <- df[-train, ]

# Eliminación de tablas temporales
rm(df, train, correlaciones, varianzas)



# Balanceo de clase para la variable de respuesta
# Se renombran variables para pdoer aplicar la función
names(df_train) <- gsub(" ", "_", names(df_train))
names(df_train) <- gsub("/", "_", names(df_train))
names(df_test) <- gsub(" ", "_", names(df_test))
names(df_test) <- gsub("/", "_", names(df_test))

df_balanced <- ovun.sample(response~., data = df_train, method = "under", p = 0.3, seed =
456)
df_balanced <- df_balanced[[3]]


# Vistazo de las proporciones de los niveles de la variable de respuesta

table(df_balanced$response)
prop.table(table(df_balanced$response))
```

```
table(df_train$response)
prop.table(table(df_train$response))


# Estandarización de variables

df_balanced = df_balanced %>% mutate_if(is.numeric, scale)
df_test = df_test %>% mutate_if(is.numeric, scale)




# Entrenamiento----

# KNN

# HIPERPARÁMETROS, NÚMERO DE REPETICIONES Y SEMILLAS PARA CADA REPETICIÓN
particiones  <- 2
repeticiones <- 2

# Hiperparámetros
hiperparametros <- data.frame(k = c(5, 20, 50))

set.seed(456)
seeds <- vector(mode = "list", length = (particiones * repeticiones) + 1)
for (i in 1:(particiones * repeticiones)) {
 seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
 }
seeds[[(particiones * repeticiones) + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
control_train <- trainControl(method = "repeatedcv", number = particiones,
                      repeats = repeticiones, seeds = seeds,
                      returnResamp = "final", verboseIter = FALSE)

# AJUSTE DEL MODELO
set.seed(456)
modelo_knn <- train(response ~ ., data = df_balanced,
                method = "knn",
                tuneGrid = hiperparametros,
                metric = "Accuracy",
                trControl = control_train)



# Random Forest


particiones  <- 2
repeticiones <- 2
```

```
hiperparametros <- expand.grid(mtry = c(5, 20, 35))

set.seed(456)
seeds <- vector(mode = "list", length = (particiones * repeticiones) + 1)
for (i in 1:(particiones * repeticiones)) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[(particiones * repeticiones) + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
control_train <- trainControl(method = "repeatedcv", number = particiones,
                      repeats = repeticiones, seeds = seeds,
                      returnResamp = "final", verboseIter = FALSE)

# AJUSTE DEL MODELO
set.seed(456)
modelo_rf <- caret::train(response ~ ., data = df_balanced,
                    method = "rf",
                    tuneGrid = hiperparametros,
                    metric = "Accuracy",
                    trControl = control_train)

modelo_rf

# Pedricciones con modelo óptimo
pred_rf = predict(modelo_rf, df_test)

pred_rf <- factor(pred_rf, levels = levels(df_test$response))

confusionMatrix(data = pred_rf, reference = df_test$response, positive = "1")

# Pedricciones con umbral modificado
pred_prob_rf = predict(modelo_rf, df_test, type = "prob")

roc.curve(df_test$response, pred_prob_rf$`1`)

pred_rf_umbral = factor(ifelse(pred_prob_rf$`1` >= 0.25, "1", "0"), levels =
levels(df_test$response))

confusionMatrix(data = pred_rf_umbral, reference = df_test$response, positive = "1")

varImp(modelo_rf)

# Modelo XGBOOST
particiones  <- 2
repeticiones <- 2

# Hiperparámetros
hiperparametros <- expand.grid(lambda = c(0),
```

```
                    alpha = c(0),
                    nrounds = c(50, 100, 200),
                    eta = c(0.3, 0.2, 0.5))

set.seed(456)
seeds <- vector(mode = "list", length = (particiones * repeticiones) + 1)
for (i in 1:(particiones * repeticiones)) {
  seeds[[i]] <- sample.int(1000, nrow(hiperparametros))
}
seeds[[(particiones * repeticiones) + 1]] <- sample.int(1000, 1)

# DEFINICIÓN DEL ENTRENAMIENTO
control_train <- trainControl(method = "repeatedcv", number = particiones,
                    repeats = repeticiones, seeds = seeds,
                    returnResamp = "final", verboseIter = FALSE)


set.seed(456)
modelo_xg <- train(response ~ ., data = df_balanced,
              method = "xgbLinear",
              tuneGrid = hiperparametros,
              metric = "Accuracy",
              trControl = control_train)

modelo_xg

# Pedricciones con modelo óptimo
pred_xg = predict(modelo_xg, df_test)

pred_xg <- factor(pred_xg, levels = levels(df_test$response))

confusionMatrix(data = pred_xg, reference = df_test$response, positive = "1")


pred_prob_xg = predict(modelo_xg, df_test, type = "prob")

roc.curve(df_test$response, pred_prob_xg$`1`)

varImp(modelo_xg)
```