TESIS DOCTORAL

Aplicación de algoritmos de aprendizaje automático para la detección de anomalías de tráfico en entornos IoT

Laura Victoria Vigoya Morales

2023

Doctorado en Tecnologías de la Información y las Comunicaciones

UNIVERSIDADE DA CORUÑA

Aplicación de algoritmos de aprendizaje automático para la detección de anomalías de tráfico en entornos IoT

Laura Victoria Vigoya Morales

TESIS DOCTORAL

Enero, 2023

Dirigida por:
Víctor Manuel Carneiro Díaz
Diego Fernández Iglesias

Doctorado en Tecnologías de la Información y las Comunicaciones

UNIVERSIDADE DA CORUÑA

Dr. Víctor Manuel Carneiro Díaz
Titular de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

Dr. Diego Fernández Iglesias
Profesor contratado doctor
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

CERTIFICAN

Que la memoria titulada: *"Aplicación de algoritmos de aprendizaje automático para la detección de anomalías de tráfico en entornos IoT"* ha sido realizada por Laura Victoria Vigoya Morales bajo nuestra dirección en el Departamento de Ciencias de la Computación y Tecnologías de la Información de la Universidade da Coruña, y concluye la Tesis Doctoral que presenta para optar al grado de Doctor en Ingeniería Informática.

En A Coruña, a 23 de Enero de 2023

Fdo.: Víctor Manuel Carneiro Díaz
Director de la Tesis Doctoral

Fdo.: Diego Fernández Iglesias
Director de la Tesis Doctoral

Fdo.: Laura Victoria Vigoya Morales
Autor de la Tesis Doctoral

A Martha, Jorge y Yesi, por construirme unas alas, decorarlas, e incentivarme a volar.

Agradecimientos

Quiero agradecer a mis tutores Diego y Víctor, por su paciencia, su entrega constante a lo largo de estos años y por abrir la puerta a un mar de nuevos conocimientos. A Manu, por los tantos fines de semana conectado, su apoyo constante y la gran biblioteca de frases de humor Español característico. A profesores y compañeros del grupo de Telemática, LBD y API, que iban haciendo aportes al conocimiento en cada reunión o café.

A Los Vigoya Morales, por estar ahí para todo, amigos incondicionales, porque nada de esto hubiera sido posible sin ustedes.

A Los Canedo Freire y Su por ser mi familia, animarme y demostrarme que de las que parecen las peores cosas se obtienen las más grandes recompensas. A las Lauras, Rebe, Marta, Elvira y Diego por hacer esta ruta conmigo, una travesía memorable y feliz. A Nydia, Mónica, Cesar, Luz, Ana, Lore y demás compis de piso, de equipo y a todos los amigos y personas que directa o indirectamente estuvieron ahí para cuando lo necesitaba, como dijo Cerati... "No sólo no hubiéramos sido nada sin ustedes, sino con toda la gente que estuvo a nuestro alrededor desde el comienzo. Algunos siguen hasta hoy... **¡Gracias totales**!".

Laura

¿Cuál es la más digna acción del ánimo, sufrir los tiros penetrantes de la fortuna injusta, u oponer los brazos a este torrente de calamidades y darles fin con atrevida resistencia?

–Víctor Carneiro

Índice general

Índice de figuras

Índice de tablas

Acrónimos

6LoWPAN *IPv6 over Low power Wireless Personal Area Networks*. 17

ACCS *Cyber Range Lab del Australian Centre for Cyber Security*. 14

ACF Función de autocorrelación. 31

ACK *acknowledgement*. 15, 42, 45, 55

AdaBoost *Adaptive Boosting*. 6, 11, 14, 18, 19, 74, 78, 83, 88, 90, 92, 96

AIC Criterio de información de Akaike. 31

ANN Redes Neuronales Artificiales. 10, 11, 13, 19

APC *American Power Conversion*. 24

API interfaz de programación de aplicaciones. 17

ARIMA Modelo autorregresivo integrado de media móvil. 20, 27, 31, 32

ARM *Association Rule Mining*. 10

ARMA Modelo autorregresivo de media móvil. 20, 27, 31

ARP *Address Resolution Protocol*. 14–16, 19, 59, 96

AWS Amazon Web Services. 11

BIC Criterio de información Bayesiano. 31

BNB Bernoulli Naive Bayes. 73, 76, 77

CIDAD *CoAP-IoT Anomaly Detection Dataset*. 41, 50, 59, 61, 68, 70, 90, 96, 97

CITIC Centro de Investigación en Tecnologías de la Información y las Comunicaciones. 3, 23, 96

CNN Redes Neuronales Convolucionales. 14–16, 19

CoAP *Constrained Application Protocol*. 5, 17, 19, 41–45, 58–61, 93, 96, 97

CPD centro de procesamiento de datos. 3, 4, 7, 18, 23–25, 33, 38, 96

CRISP-DM *Cross Industry Standard Process for Data Mining*. 49

CSV Valores separados por comas. 40, 47

cv validacion cruzada. 65, 68, 69, 72, 76, 81, 84, 96

DAD *Annotated Dataset for Anomaly Detection in a Data Center with IoT Sensors*. 5, 35, 41, 50–52, 55, 56, 58, 68, 70, 82, 87, 96, 97

DDoS Denegación de servicio distribuido. 12, 14, 19

DFF *Deep Feed Forward*. 15, 19

DHCP *Dynamic Host Configuration Protocol*. 15

DNN *Deep Neural Network*. 17, 19

DNS sistema de nombres de dominio. 11, 15

DoS denegación de servicio. 10, 14–16, 18, 19

DT Árboles de decisión. 10–13, 15–19

DTLS Datagram Transport Layer Security. 43

ECM Error cuadrático medio. 30

ET *Extra Tree*. 11

FIN *finished*. 55

FN falsos negativos. 88, 90, 93, 97

FP falsos positivos. 88, 90, 91, 93, 97

FTP *File Transfer Protocol*. 11

GRU *Gated Recurrent Unit*. 17, 19

H2M humano a máquina. 17

HTTP *Hypertext Transfer Protocol*. 11, 15, 18, 19, 41–44

IBM *International Business Machines Corporation*. 36

ICMP *Internet Control Message Protocol*. 14, 19, 59

ICMPV6 Internet Control Message Protocol v6. 17, 59

ID3 *Iterative Dichotomiser 3*. 14, 19

IDS Sistemas de Detección de Intrusiones. 2–5, 7, 9–11, 13, 14, 16, 18, 35, 95

IETF *Internet Engineering Task Force*. 14, 41

IIoT Internet Industrial de las Cosas. 12, 13, 17–19

IMAP *Internet Message Access Protocol*. 11

IoT Internet de las cosas. 1–7, 9–19, 26, 27, 31, 32, 35, 36, 38, 41, 42, 53, 54, 95–98

IoT-NID *IoT Network Intrusion Dataset*. 15

IP *Internet Protocol*. 12, 24, 46, 51–53, 56, 58–60, 66, 96, 98

IPV6 Internet Protocol v6. 17, 59, 96

IRC *Internet Relay Chat*. 15

KDD *Knowledge Discovery and Data Mining*. 10

kNN k vecinos más cercanos. 10, 12–14, 16–19

LAN red de área local. 10, 11

LDA Análisis Discriminante Lineal. 15, 16, 18, 19

LLdoS *Lincoln Laboratory DDoS*. 10

LOIC *Low Orbit Ion Cannon*. 18

LR Regresiones Lineales. 6, 13, 15–17, 19, 88, 89, 91, 93, 96, 97

LSTM-RNN *Long-Short Term Memory Recurrent Neural Network*. 14, 19

LSVC *Linear Support Vector Classification*. 18, 19

LSVM Maquinas de vectores de soporte lineal. 12, 19

M2H máquina a humano. 17

M2M máquina a máquina. 1, 17, 36, 38, 42

MiTM *Man-in-the-Middle*. 15, 16, 19

ML aprendizaje automático. 2–7, 9, 10, 12, 14, 16–19, 50, 75

MLP Multi-layer Perceptron. 13, 14, 16, 17, 19

MQTT *MQ Telemetry Transport*. 5, 14, 16, 17, 19, 35–39, 51–56, 60, 93, 96, 97

MQTT-SN MQTT para redes de sensores. 36

MTU Unidad Terminal Maestra. 12

MUD *Manufacturer Usage Description*. 14

NB Naive Bayes. 6, 10, 11, 13–19, 82, 88, 90, 93, 96, 97

NIDS Sistemas de Detección de Intrusiones de Red. 11

NIMS *Network Information Management and Security*. 11

NN Redes Neuronales. 12, 16, 19

NTP *Network Time Protocol*. 39, 53

PACF Función de autocorrelación parcial. 31

PCA Análisis de Componentes Principales. 15

PDU *Power Distributor Unit*. 23

PLC controlador lógico programable. 13

POP3 *Post Office Protocol*. 11

QDA Análisis Discriminante Cuadrático. 14, 19

QoS Calidad de servicio. 37, 42

R2 *R-squared*. 20

R2L *Root to Local*. 10

RCNN Random Forest con CNN. 16, 19

REST transferencia de estado representacional. 42, 43

RF *Random Forest*. 6, 11–19, 73, 88, 90, 92, 93, 96, 97

RFE Eliminación recursiva de caracteristicas. 65, 67, 69, 96

RNN Redes Neuronales Recurrentes. 14, 15, 19

ROC AUC área bajo la curva característica operativa del receptor. 75, 77–80, 82–84, 96

RRD *Round Robin Database*. 11

RST *reset*. 55

RTSP *Real Time Streaming Protocol*. 18, 19

RTU Unidad Terminal Remota. 12

SCADA sistemas de supervisión adquisición de datos. 12, 13, 19

SD desviación típica. 76, 81

SDN *Software-defined networking*. 14

SGD Descenso de gradientes estocástico. 18, 19

SMOTE *Synthetic Minority Over-sampling Technique*. 65, 68, 70, 75, 96

SMTP *Simple Mail Transfer Protocol*. 11

SSH *Secure Shell*. 11

SSL *Secure Sockets Layer*. 15

STL tendencia estacional utilizando el método LOESS. 21, 27, 32, 33, 96

SVC Clasificador de Vectores de Soporte Lineal. 73, 74, 84

SVM Maquinas de vectores de soporte. 6, 11–17, 19, 74, 80, 84, 88, 89, 91, 93, 96, 97

SYN *synchronization*. 55

TAR Temperatura de Aire de Retorno. 25, 26, 38, 44, 51, 59

TAS Temperatura de Aire de Suministro. 25, 26, 31, 33, 38, 44, 51, 59

TCP *Transmission Control Protocol*. 1, 10, 12–14, 18, 19, 36, 37, 42, 51–55, 66, 67, 96, 98

Telnet *Teletype Network*. 11, 15

TFEU Temperatura de Fluido que Entra en la Unidad. 24–26, 38, 44, 51, 59

TFSU Temperatura de Fluido que Sale de la Unidad. 24–26, 38, 44, 51, 59

TI tecnologías de la información. 23

TP verdaderos positivos. 89

TUIDS *Tezpur University Intrusion Detection System*. 11

U2R *User to Root*. 10

UDC Universidade da Coruña. 3, 23

UDP *User Datagram Protocol*. 1, 14–19, 36, 42, 44, 52, 58, 59, 61, 96

UNIBS Universidad de Brescia. 11

UNSW Universidad de New South Wales Sydney. 14, 17, 19

URI *Uniform Resource Identifier*. 42–44

URL *Uniform Resource Locator*. 18

VSL capa de estado virtual. 13

WLAN red de área local inalámbrica. 17

XCNN eXtreme Gradient Boosting con CNN. 16, 19

XGBoost eXtreme Gradient Boosting. 18, 19

Abstract

The huge number of lightweight devices interconnected in IoT infrastructures generates a large amount of vulnerable information. In this field, techniques based on supervised machine learning have gained credibility as a successful approach for anomaly detection in IoT networks. This thesis proposes the detection of traffic anomalies in IoT through the application of supervised machine learning algorithms. Given the absence of *datasets* in the required conditions, two new ones were generated using the MQTT protocol and the CoAP protocol, respectively. Both present three types of anomalies in the *payload* of the messages and are based on data from a real CPD network of temperature sensors. For the mathematical modeling of the temperature values, the STL method of time series decomposition was used. For the classification, Linear Regressions, Naive Bayes, Random Forest, Support Vector Machines, and AdaBoost were applied. After using various preprocessing techniques and algorithms, as well as hyperparameterization for model optimization, the results show, using different traditional metrics, that classifiers can achieve a high detection rate in all experiments. Tree-based models present the best results.

Resumen

El ingente número de dispositivos ligeros interconectados en las infraestructuras IoT genera una gran cantidad de información vulnerable. En este ámbito, las técnicas basadas en aprendizaje automático supervisado han ganado credibilidad como una aproximación exitosa para la detección de anomalías en redes IoT. Esta tesis propone la detección de anomalías de tráfico en IoT mediante la aplicación de algoritmos de aprendizaje automático supervisado. Ante la ausencia de *datasets* en las condiciones requeridas, se generaron dos nuevos empleando el protocolo MQTT y el protocolo CoAP, respectivamente. Ambos presentan tres tipos de anomalías en el *payload* de los mensajes, y están basados en los datos de una red de sensores de temperatura de un CPD real. Para el modelado matemático de los valores de temperatura, se empleó el método STL de descomposición de series temporales. Para la clasificación, se aplicaron Regresiones Lineales, Naive Bayes, Random Forest, Máquinas de Vectores de Soporte y AdaBoost. Tras utilizar diversas técnicas y algoritmos de pre-procesado, así como hiperparametrización para la optimización de los modelos, los resultados muestran, utilizando diferentes métricas tradicionales, que los clasificadores pueden lograr una alta tasa de detección en todos los experimentos. Los modelos basados en árboles presentan los mejores resultados.

Resumo

O inxente número de dispositivos lixeiros interconectados nas infraestruturas IoT xera unha grande cantidade de información vulnerable. Neste ámbito, as técnicas baseadas en aprendizaxe automática supervisada gañaron credibilidade como unha aproximación exitosa para a detección de anomalías en redes IoT. Esta tese propón a detección de anomalías de tráfico en IoT mediante a aplicación de algoritmos de aprendizaxe automática supervisada. Ante a ausencia de *datasets* nas condicións requiridas, xeráronse dous novos empregando o protocolo MQTT e o protocolo CoAP, respectivamente. Ambos presentan tres tipos de anomalías no *payload* das mensaxes, e están baseados nos datos dunha rede de sensores de temperatura dun CPD real. Para o modelado matemático dos valores de temperatura, empregouse o método STL de descomposición de series temporais. Para a clasificación, aplicáronse Regresións Lineais, Naive Bayes, Random Forest, Máquinas de Vectores de Soporte e Adaboost. Tras utilizar diversas técnicas e algoritmos de preprocesado, así como hiperparametrización para a optimización dos modelos, os resultados mostran, utilizando diferentes métricas tradicionais, que os clasificadores poden lograr unha alta taxa de detección en todos os experimentos. Os modelos baseados en árbores presentan os mellores resultados.

Capítulo 1

Introducción

Internet de las cosas (IoT) se ha convertido en la fuerza que está detrás de la automatización del hogar, las ciudades inteligentes, los sistemas de salud modernos y los procesos de fabricación avanzada [1]. Sin embargo, su implementación esta compuesta por un conjunto de muchos dispositivos restringidos interconectados, limitados en energía, ancho de banda, memoria, entre otros, que genera un gran aumento en las dimensiones de los sistemas y tráfico en las redes, y en consecuencia, un incremento considerable de amenazas y vulnerabilidades[2].

Un modo de aliviar la carga computacional y liberar recursos en la red IoT es mediante el uso de estándares especializados y protocolos de comunicaciones específicos que se adapten a la necesidades del entorno. En redes IoT, en la capa de transporte, *Transmission Control Protocol* (TCP) y *User Datagram Protocol* (UDP) son los protocolos dominantes para la mayoría de las aplicaciones, mientras que en la capa de aplicación encontramos en la actualidad diversos protocolos populares disponibles orientados a la comunicación máquina a máquina (M2M). Existe en IoT una necesidad crítica de un arquitectura de capas flexible, sin embargo, ante el numero cada vez mayor de arquitecturas propuestas, no se ha definido claramente un acuerdo común en la definición de protocolos o un modelo de referencia convergente[3].

Esta falta de estandarización en los sistemas IoT, su falta de madurez, relativa simplicidad y el gran número de dispositivos que se pueden encontrar en una misma infraestructura conlleva al manejo de una gran cantidad de información susceptible de modificación. Las redes IoT pueden presentar no solo los mismos problemas de seguridad que las redes de sensores y los de comunicación móvil e Internet, sino también algunos más particulares, lo que las convierte en un blanco de interés para usuarios remotos malintencionados,

hackers o ciber-atacantes. El desafío es entonces prevenir mitigar ó limitar el impacto de nuevos e ingeniosos modelos maliciosos que pueden alterar los datos y la integridad de la información. [4] [5] [6].

Una forma de manejar problemas de seguridad es a través de los Sistemas de Detección de Intrusiones (IDS), los cuales ayudan a descubrir, determinar e identificar el uso no autorizado, la duplicación, la alteración y la destrucción de los sistemas de información. Los Sistemas de Detección de Intrusiones se pueden clasificar en diferentes tipos según el sistema que monitorizan o en función de cómo se implementan. Los IDS basados en firmas identifican patrones de trafico o datos de aplicaciones presuntamente maliciosos, y funcionan bien ante ataques conocidos, pero tienen un bajo rendimiento ante los desconocidos. Los IDS basados en anomalías de tráfico centran su funcionamiento en estandarizar patrones en un conjunto de datos y localizar aquellos cuyo comportamiento es anormal o atípico, de tal manera que son capaces de detectar ataques nuevos o desconocidos, y pueden ayudar a detectar información importante y crítica en diversas aplicaciones. Las anomalías en una red no siempre se pueden categorizar como un ataque y no siempre son elementos dañinos. Aun así, pueden brindar peculiaridades sobre el comportamiento del tráfico y ayudar a identificar comportamientos cruciales en diversas aplicaciones [7].

Las técnicas de detección de anomalías basadas en aprendizaje automático (ML), han ganado credibilidad como una aproximación exitosa para la detección de anomalías en la red, incluidas las redes IoT. Los métodos cimentados en ML pueden capturar y examinar el tráfico de IoT benigno y anómalo de manera precisa, mediante algoritmos confiables y sólidos, brindando seguridad a los dispositivos y redes de IoT, e incluso pueden predecir ataques nuevos o de día cero [8].

En el aprendizaje supervisado, los algoritmos trabajan con datos "etiquetados" *(labeled data)*, intentado encontrar una función que, dadas las variables de entrada *(input data)*, les asigne la etiqueta de salida adecuada. El algoritmo se entrena con un "histórico" de datos y así "aprende" a asignar la etiqueta de salida adecuada a un nuevo valor, es decir, predice el valor de salida. El aprendizaje no supervisado tiene lugar cuando no se dispone de datos "etiquetados" para el entrenamiento. Sólo se conocen los datos de entrada, pero no existen datos de salida que correspondan a un determinado input. Por ello, tienen un carácter exploratorio. [9] En teoría, los métodos supervisados proporcionan una mejor tasa de detección que los métodos semisupervisados y no supervisados, ya que tienen acceso a más información. Los métodos supervisados requieren disponer de un conjunto de datos de tráfico representativo (*dataset*) para construir su modelo predictivo. Estos *datasets* deben encontrarse perfectamente etiquetados, considerar las tendencias reales de las vulnerabilidades de seguridad incluyendo varios tipos de anomalías, y permitir la validación y evaluación de la efectividad del sistema [10].

1.1. Motivación

En el año 2018, cuando se inicia esta investigación, se realiza una revisión de los IDS en IoT basados en anomalías haciendo uso de aprendizaje automático supervisado, empleando protocolos específicos, en donde se encuentra un desarrollo casi inexistente. Hasta ese momento los *datasets* publicados específicos en IoT eran limitados y los sistemas de detección eran desarrollados con *datasets* genéricos ampliamente difundidos, o mediante la mezcla de estos con trazas de trafico IoT que permitiera una representación del entorno.

Esta situación despierta el interés por generar sistemas de detección de intrusiones en entornos propios de IoT, con características particulares. Ante la carencia de *datasets* en las condiciones requeridas, nuestro primer propósito era generar *datasets* de IoT etiquetados que pudieran ser empleados para la clasificación de anomalías de tráfico mediante algoritmos de aprendizaje automático supervisado, con protocolos y condiciones específicas del entorno.

Por otra parte, el tráfico anómalo en una red IoT también puede originarse de una configuración incorrecta, una instalación irregular o un mal funcionamiento del hardware de los sensores. En el centro de procesamiento de datos (CPD) del Centro de Investigación en Tecnologías de la Información y las Comunicaciones (CITIC) de la Universidade da Coruña (UDC) [11], un fallo inicial de software, derivado de la configuración incorrecta de los sensores, provocó que los sensores de temperatura enviaran datos erróneos al sistema de refrigeración, afectando la temperatura de los dispositivos, provocando una falla de hardware que colapsó todo el sistema. Este hecho inspiró a adoptar en este trabajo una red de sensores IoT, con la estructura y funcionamiento de un CPD real, en donde las anomalías se presentan sobre los mensajes transmitidos en los protocolos de la capa de aplicación de la red IoT.

Para poder realizar las alteraciones sobre los datos de temperatura, requeríamos de datos de temperatura que se ajustaran a situaciones reales de la red de sensores, pero que pudieran ser generados dinámicamente en diferentes entornos. Por ello, se realizó un análisis minucioso de las condiciones del CPD y un estudio de diferentes técnicas de modelado de sensores que permitieran replicar la red de sensores real en dispositivos simulados.

La implementación relativamente sencilla y los resultados obtenidos en IDS que emplean algoritmos de aprendizaje automático han hecho de estas técnicas la opción elegida en esta tesis como instrumento de desarrollo. Una vez generados los *datasets*, el desafío era demostrar su aplicación para tareas de clasificación de detección de anomalías de tráfico empleando aprendizaje automático en entornos IoT.

1.2. Objetivos

El objetivo general de esta tesis es detectar anomalías de tráfico en IoT mediante la aplicación de algoritmos de aprendizaje automático e inteligencia colectiva. Los objetivos específicos se detallan a continuación:

- Desarrollar el modelado matemático de sensores de temperatura, que emulen el comportamiento de una redes de sensores reales para que puedan ser implementados en dispositivos IoT.
- Diseñar e implementar entornos IoT que hagan uso de protocolos específicos e introduzcan alteraciones en el *payload* de los mensajes.
- Crear *datasets* semi-sintéticos de IoT, anotados, fáciles de seguir y manipular, que posibiliten la aplicación de algoritmos de aprendizaje supervisado.
- Implementar y optimizar técnicas para manejo de datos desbalanceados, discretización, extracción de características y ajuste de hiperparámetros.
- Aplicar diferentes algoritmos de aprendizaje automático, evaluando su rendimiento, a través de diferentes métricas tradicionales, para validar los conjuntos de datos propuestos.

1.3. Desarrollo de la investigación

La investigación comienza haciendo una revisión de los IDS desarrollados específicamente para detección de anomalías en entornos IoT. Ante la carencia de estos, la intención fue reducir la brecha existente en sistemas de detección de anomalías de trafico para IoT con uso de protocolos específicos en la capa de aplicación.

La figura 1.1 presenta las etapas y fases llevadas a cabo durante el proceso de investigación.

El análisis de la estructura real del CPD, la localización de los sensores y la definición de los protocolos para el intercambio de mensajes, sirvió de base para el diseño del escenario propuesto. La construcción de un entorno simulad de redes de sensores en IoT, posibilitó gestionar los sistemas de comunicaciones, adaptando los requerimientos de cómputo, espacio de almacenamiento, ancho de banda y permitiendo actuar sobre el tráfico de la red, en el momento en que es generado. Así mismo, permitió crear los *datasets* en

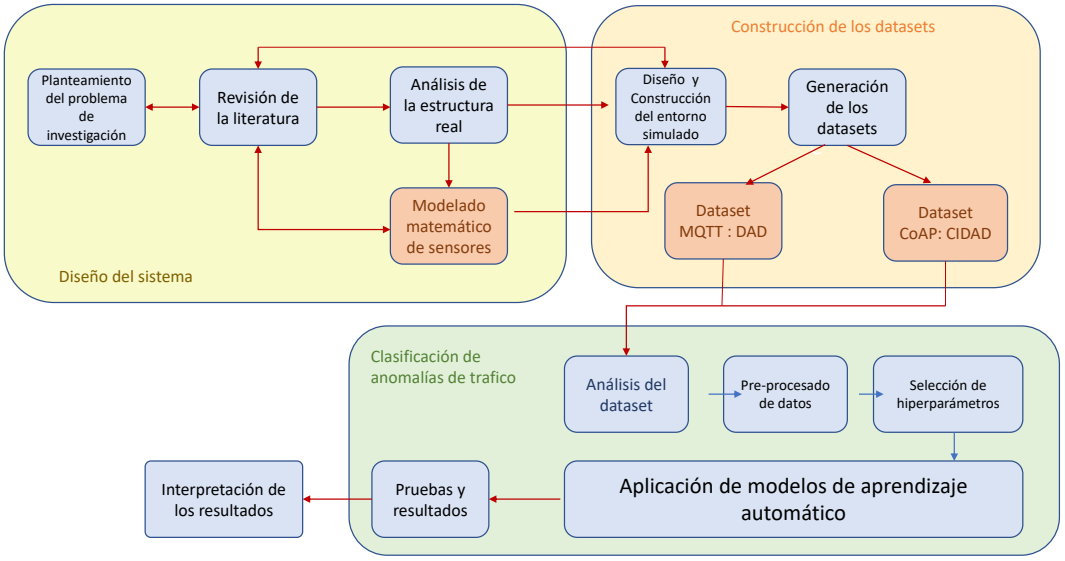

Figura 1.1: Fases y etapas del proceso de investigación

las condiciones necesarias, inyectando el tráfico anómalo en la carga útil de los mensajes de temperatura de manera controlada para el correcto etiquetado.

Para evitar el sesgo en los datos al replicar los valores de temperatura en los *datasets*, se realizó el modelado del comportamiento de los sensores de forma tal que se ajustaran a los valores de distribuciones reales, pero generando valores dinámicos representativos del sistema. El modelo resultante fue implementado en cada uno de los dispositivos que hacen el papel de sensores en el entorno virtual.

El siguiente paso fue el diseño y construcción del entorno simulado que permitió la generación de los *datasets*. El primero de los *datasets*, denominado *Annotated Dataset for Anomaly Detection in a Data Center with IoT Sensors* (DAD) [12], realiza el intercambio de mensajes haciendo uso del protocolo *MQ Telemetry Transport* (MQTT). El segundo, llamado CIDAD (CoAP-IoT Anomaly Detection Dataset) [13], consiste en un *dataset* que implementa el protocolo *Constrained Application Protocol* (CoAP) en la capa de aplicación.

El auge en el desarrollo de las redes IoT y la carencia de IDS específicos ocasionó que muchos investigadores despertaran un gran interés en generar sistemas y *datasets* para aprendizaje automatico supervisado. Para el año 2022 alrededor de 19 *datasets* de IoT para IDS usando ML habían sido publicados, demostrando la necesidad que existía en la producción de nuevos mecanismos para crear e implementar modelos que sean capaces de reconocer el comportamiento anormal de las redes IoT. Los IDS basados en dichos

datasets, presentan la aplicación de variedad de algoritmos de aprendizaje automático, que tras su exploración, permitió seleccionar para la validación de los *datasets* propuestos cinco de los modelos de aprendizaje automático más empleados: Regresiones Lineales (LR), Naive Bayes (NB), *Random Forest* (RF), Maquinas de vectores de soporte (SVM) y *Adaptive Boosting* (AdaBoost).

La generación del sistema de clasificación consistió en determinar el comportamiento de las redes IoT simuladas. El análisis de datos permitió lidiar con la variedad, volumen, y necesidad de procesamiento para la detección automática, haciendo los datos perfectamente entendibles para los algoritmos de ML, y poder evitar clasificaciones no deseadas.

Finalmente, el entrenamiento y aplicación de los modelos es llevado a cabo, demostrando la aplicación de algoritmos de aprendizaje automático para detección de anomalías en entornos IoT.

1.4. Aportes de la investigación

Las contribuciones al ámbito científico, tras la realización de esta tesis son:

- La presentación del modelado matemático de una red de sensores de temperatura, mediante técnicas estadísticas, para su implementación en dispositivos ligeros y de baja carga computacional que simulan el comportamiento de sensores reales.

- La publicación de dos *datasets* completos y perfectamente etiquetados, basados en la reproducción de un escenario real, con un número suficiente de muestras, extracción de características específicas y mezcla de anomalías, para la clasificación de anomalías de trafico mediante algoritmos de aprendizaje automático.

- La optimización de modelos de aprendizaje automático mediante la combinación de técnicas y algoritmos de pre-procesado de datos, generación y selección de características, discretización de los datos y eliminación de información redundante, junto con la hiperparametrización de los modelos y el manejo de datos desbalanceados.

- El análisis y validación de los *datasets* generados para la detección de anomalías empleando aprendizaje automático.

1.5. Estructura de la tesis

Esta tesis se encuentra dividida en 10 capítulos. El capítulo 2 presenta la revisión de la literatura a lo largo del desarrollo de la tesis. Inicialmente se hace una revisión de los Sistemas de Detección de Intrusiones, no específicos en redes IoT, pero que puede contener trazas de estas, planteados al comienzo de la investigación. Luego, se exponen los *datasets* en entornos IoT encontrados a lo largo del desarrollo de la investigación, con una cantidad representativa de muestras, variedad de escenarios, diferentes anomalías y perfectamente etiquetados, diseñados para ser empleados con algoritmos de ML supervisado. Finalmente, se presenta una revisión de modelados de sensores haciendo uso de técnicas estadísticas

El capítulo 3 comienza exponiendo la infraestructura física existente en el CPD, la red de sensores, su localización, operación y la relación entre ellos, como base para el diseño del entorno. Por último, se presenta el análisis estadístico de la muestra tomada de sensores reales, que permite realizar el modelado matemático de los sensores de temperatura.

Los capítulos 4 y 5 contextualizan los escenarios y los protocolos seleccionados para el intercambio de mensajes, exponiendo el diseño y la construcción de cada uno de los entornos simulados y los procesos para la generación de los *datasets*.

El análisis de los *datasets* es llevado a cabo en el capítulo 6, estudiando los comportamientos del tráfico tanto a nivel de paquete como de flujo, donde es posible.

Para que los datos estén en las condiciones perfectas para entrar en el clasificador es necesario realizar el pre-procesado de los datos que se muestra en el capítulo 7. Aquí se definen las técnicas y métodos a emplear para la discretización, la selección de características y el manejo de datos desbalanceados.

El proceso de entrenamiento de algoritmos de aprendizaje automático es aplicado a los datos en el capítulo 8, para luego presentar la aplicación de los modelos y los resultados obtenidos en el capítulo 9.

Finalmente, en el capítulo 10 se exponen las conclusiones y los trabajos futuros.

Capítulo 2

Revisión de la literatura

Este capítulo presenta la revisión de la literatura a lo largo del desarrollo de la tesis. El problema inicial de investigación nos lleva a buscar sistemas para detectar anomalías de tráfico en IoT mediante la aplicación de algoritmos de aprendizaje automático. La cantidad de IDS encontrados que cumplieran estas condiciones fue escaso, siendo mayoritariamente IDS no específicos utilizados para IoT, demostrando la brecha existente de dichos sistemas en este entorno. La sección 2.1 del capítulo presenta los IDS encontrados hasta el 2018, haciendo especial énfasis en los *datasets* utilizados en enfoques IoT y los modelos de ML aplicados para la detección.

Al principio de la tesis la cantidad de *datasets* específicos encontrados era escasa, sin embargo, durante su desarrollo muchos investigadores tornaron sus proyectos a la generación de *datasets* para IoT, generando varios de ellos pero ninguno en las condiciones requeridas en esta tesis. En consecuencia, la sección 2.2 del capítulo presenta todos los *datasets* en entornos propios de IoT, precisando los IDS que los emplean, así como los modelos de ML utilizados en la detección de anomalías de tráfico.

Las anomalías inyectadas en el *dataset* iban a encontrarse sobre los mensajes de los valores de los sensores de temperatura. Por ello, y para evitar datos estáticos y el sobreajuste en los modelos, se optó por la construcción de un modelo matemático que generará valores de temperatura, conservando las características y relaciones del sensor al que pertenece. Como etapa previa a este proceso, fue necesario realizar un estudio de técnicas estadísticas aplicadas para la representación de sensores, enfocados a sensores de temperatura, el cual se expone en este capítulo, en la sección 2.3.

2.1. IDS basados en algoritmos de ML

Dada la falta inicial de *datasets* públicos y específicos disponibles en entornos IoT, la aplicación de IDS comienza ajustándose a *datasets* genéricos ampliamente aceptados por la comunidad científica, con algunas trazas de redes IoT. Tres de los *datasets* más ampliamente difundidos para Sistemas de Detección de Intrusiones *son el KDD Cup 1999 Data*, el *NSL-KDD dataset*, y el *DARPA 2000*, los cuales brindan mejoras sobre sus versiones anteriores.

El KDD Cup 1999 [14] [15] [16] contiene un gran número de conexiones simples con 41 características, 24 tipos de ataques y es simulado bajo una típica red LAN usada por las fuerzas aéreas norteamericanas. En este *dataset* se realiza la adquisición de 9 semanas de datos TCP, aplicando denegación de servicio (DoS), *Root to Local* (R2L), *User to Root* (U2R) y ataques *probing*. Uno de los principales problemas de este *dataset* es el gran numero de registros duplicados, lo que puede llevar a los modelos a tasas de detección a favor de los registros frecuentes y puede no ser una perfecta representación de la red.

EL *dataset NSL-KDD* [17] pretende solucionar inicialmente los registros duplicados del KDD Cup 1999, pero manteniendo su vasto número de registros. Realiza modificaciones sobre los conjuntos de entrenamiento y validación, permitiendo realizar variados experimentos con gran número de algoritmos de manera más fiable [18] [19]. Pajoouh et al. [20] usaron este *dataset* para proponer un modelo que aplicaba un modulo de clasificación de dos niveles empleando NB y una versión de factor de certeza de un k vecinos más cercanos para identificar actividades maliciosas tales como ataques U2R y R2L.

El *dataset DARPA 2000* [21] simula dos escenarios de ataque de *Lincoln Laboratory DDoS* (LLdoS) 1.0 y 2.0, y genera variaciones en las redes para presentar diferentes escenarios de ataques. Por otra parte, McHugh [22] plantea una serie de deficiencias en los conjuntos de datos nombrados, donde enfatiza en la falta de evidencia estadística de similitud con tráfico de redes típicas, bajas tasas de tráfico, distribuciones relativamente uniformes de las cuatro categorías principales de ataques y distribución sesgada de los hosts atacados, entre otros.

El *dataset* más comúnmente adoptado para IoT ha sido el *UNSW-NB15 dataset* [23], el cual ha sido empleado por muchos investigadores en sus aplicaciones en ML. El UNSW-NB15 *dataset* cubre una gran colección de tráfico normal combinado con nueve tipos de ataques: DoS, *Fuzzers*, *Analysis*, *Backdoor*, *Shellcode*, *Worm*, *Exploits*, *Generic* y *Reconnaissance*. La comparación entre este dataset y el KDD es realizada en [24]. Así mismo, Koroniotis et al. [25] aplican cuatro técnicas de clasificación, Árboles de decisión (DT), *Association Rule Mining* (ARM), Redes Neuronales Artificiales (ANN), y NB a este *data-*

set, validando la capacidad de los algoritmos para el reconocimiento de ataques. Timcenko et al. [26] utiliza este dataset para generar un IDS haciendo uso de SVM y métodos de ensamble. El *AD-IoT* [27] selecciona una parte del *dataset UNSW-NB15* modificado y propone un sistema de detección de anomalías para detectar cibertaques en nodos fog en una smart city basado en métodos de ensamble, usando clasificación por RF y *Extra Tree* (ET) y técnicas de *bagging*. Moustafa et al. [1] utilizan el *UNSW-NB15* y el *dataset* NIMS [28] con tráfico *botnet*, junto con datos de sensores de luz, humedad y temperatura conectados a un hub IoT AWS para realizar un Sistemas de Detección de Intrusiones de Red (NIDS) basado en un algoritmo de aprendizaje de ensamblado AdaBoost, usando tres técnicas de aprendizaje máquina: DT, NB y ANN.

El proyecto *"Malware Capture Facility Project"* [29] es un proyecto en el cual los principales objetivos eran generar y capturar actividades anómalas, *botnets*, con características especificas. En este analizaron, monitorizaron y capturaron diversos tipos de tráfico de red durante algunos meses de operación con diversos tipos de tráfico e incluyendo archivos RRD con la historia de la forma del tráfico, flujos bidireccionales de Argus (tanto el archivo binario como el archivo de texto), registros web para todo el tráfico web y un informe de DNS, entre otros. Las etiquetas fueron generadas manualmente por un grupo de expertos en seguridad y se agregan tanto a los archivos de *Argus* como a los *weblogs*.

Bhuyan et Al. [10] generan tres diferentes *datasets* denominados *Tezpur University Intrusion Detection System* (TUIDS), uno para intrusión, uno para escaneo coordinado y el último un TUIDS DDoS tanto a nivel de paquete como de flujo, con servicios de tráfico web, email, samba, *Teletype Network* (Telnet), FTP y bases de datos, con tráfico normal de diferentes usuarios y 14 tipos de ataques distribuidos en 6 escenarios.

Para entornos reales, la Universidad de Brescia (UNIBS) [30] presenta un *dataset* para clasificar tráfico encriptado usando SSH. Este contiene tres días consecutivos de tráfico en un día de negocios, utilizando 20 *workstations* en una red de área local.

Además de este, el IDS dataset ISCX 2012 [31] es presentado implementando físicamente un *testbed* usando dispositivos reales y tráfico real (SSH, HTTP, SMTP, IMAP, POP3 y FTP) a través de perfiles, los cuales imitan el comportamiento de usuarios en diferentes escenarios de tráfico malicioso.

Otros *datasets* son creados o modificados de los anteriormente mencionados, tales como [32], [18], [33] [34] y [35]. Estos fueron creados para explotar la cantidad de recursos existentes en los *datasets* disponibles, juntando diferentes tipos de ataques en el mismo *dataset* y de esta manera tomar ventaja de la gran cantidad de información contenida. El uso de *datasets* compuestos o modificados reduce las deficiencias encontradas y mejora la extracción de características.

2.2. Datasets de IoT

El uso emergente de las redes de IoT y su relativa novedad, junto con sus amenazas de seguridad, ha hecho que la creación de *datasets* de IoT especializados sean un esfuerzo necesario y reciente. En esta sección se presentan los *datasets* disponibles más relevantes, ordenados por año de publicación, desarrollados específicamente en redes IoT, diseñados para la detección de anomalías de tráfico mediante aprendizaje automático supervisado.

1. **SCADA IIoT** [36]: Diseñado en un entorno de Internet Industrial de las Cosas (IIoT), Lemay et al. presentan un *dataset* sintético etiquetado para redes sistemas de supervisión adquisición de datos (SCADA) de comunicación Modbus/TCP de un escenario industrial ficticio. El *dataset* contiene ataques de penetración que imitan el comportamiento de tiempo y la tasa de paquetes por unidad de tiempo, lo cual es un buen factor distintivo para los ataques. El número exacto de Unidad Terminal Maestra (MTU) y Unidad Terminal Remota (RTU) varía para ilustrar el impacto de las diferentes implementaciones en el tráfico real. El *dataset* incluye capturas de paquetes de cinco tipos de actividades maliciosas. La aplicación para detectar anomalías de tráfico fue realizada por Duque et al. en [37], donde se utilizaron cuatro algoritmos de aprendizaje automático diferentes para detectar anomalías, SVM, RF, kNN y *k-means clustering*, logrando el mejor rendimiento con el modelo SVM.

2. **Doshi et al.** [38]: Desarrollan algunas técnicas para detectar automáticamente el tráfico de ataques de IoT, lo que demuestra que el uso de comportamientos de red específicos de IoT para informar la selección de características puede resultar en una detección Denegación de servicio distribuido (DDoS) de alta precisión. El *pipeline* se ensambla para operar en las *middleboxes* de la red (por ejemplo, enrutadores, *firewalls* o conmutadores de red) y los dispositivos correspondientes que pueden ser parte de una red de *bots* en curso. Para la detección de ataques, se implementa una variedad de clasificadores, incluido el algoritmo KDTree de k vecinos más cercanos (kNN), Maquinas de vectores de soporte lineal (LSVM), DT, RF y Redes Neuronales (NN).

3. **N-BaIoT** [39]: Este *dataset* construye y recolecta tráfico desde dos redes: la primera consiste en una red de vídeo vigilancia con cámaras *Internet Protocol* (IP) con ocho diferentes tipos de ataques que afectan la disponibilidad e integridad de las conexiones de vídeo. La segunda red corresponde a una red IoT con tres PCs y nueve dispositivos IoT, uno de ellos infectado con botnet malware Mirai. El *dataset* fue

utilizado para detectar ataques de botnet en IoT extrayendo capturas instantáneas del comportamiento de la red y utilizando *deep autocoders*. Utilizando el *dataset* N-BaIoT, Kitsune [40] realiza un Sistemas de Detección de Intrusiones, el cual aprende cómo detectar ataques online en redes locales sin supervisión utilizando *autoencoders*. Abbasi [41] utiliza este *dataset* para tareas de clasificación utilizando LR y ANN.

4. **All eyes on you** [42]: Pahl et al. desarrollaron detección de anomalías de microservicios de IoT donde crearon y publicaron dos *datasets* separados. Estos datasets monitorizan las conexiones entre siete tipos diferentes de capa de estado virtual (VSL) de controladores de luz, sensores de movimiento, termostatos, baterías solares, lavadoras, cerraduras de puertas y teléfonos inteligentes de usuarios. Basando su trabajo en este *dataset*, Hasan et al. [2] proponen sistemas utilizando algoritmos de aprendizaje máquina, LR, SVM, DT, RF, y ANN. La infraestructura basada en IoT explota varios clasificadores, los cuales pueden detectar y proteger el sistema cuando este se encuentra en un estado anormal, obteniendo la mejor precisión con RF y ANN tanto en etapa de entrenamiento como de validación. El gran fallo de este *dataset* es que presenta únicamente un día de captura.

5. **Anthi et al.** [43]: Presentan un *dataset* basado en ocho dispositivos IoT comercialmente populares como ejemplo de un hogar inteligente. Este contiene la recopilación de tres semanas de datos benignos y dos semanas de datos maliciosos en un banco de pruebas en IoT en cinco escenarios con seis categorías de ataques. Además, proponen un IDS de tres capas considerando NB, *Bayesian Network*, J48, *Zero R*, *OneR*, *Simple Logistic*, SVM, Multi-layer Perceptron (MLP) y RF. Una deficiencia del conjunto de datos es que este no emplea protocolos específicos y solo contiene dos días de capturas de tráfico anómalo.

6. **IIoT MODBUS** [44]: Frazao et al. desarrollaron otro enfoque en IIoT. Aquí emplean un pequeño banco de pruebas de automatización usando MODBUS/TCP. El banco de pruebas emula un proceso controlado por un sistema SCADA que utiliza los protocolos MODBUS y TCP. Para fines de análisis, implementan un subconjunto de los posibles ataques en el banco de pruebas, inundación de ping, inundación de TCP SYN e inundación de consultas MODBUS. La inundación de consultas MODBUS consiste en registros de retención de lectura, que tienen como objetivo alcanzar el controlador lógico programable (PLC). Para validar el *dataset* se implementaron cuatro clasificadores, kNN, SVM, DT y RF, siendo el clasificador de Árboles de decisión el que mejores resultados presenta.

7. **UNSW** [45]: Un grupo de investigadores de la Universidad de New South Wales Sydney (UNSW) analizó el tráfico de red de los dispositivos IoT para conocer su comportamiento al implementar el *Internet Engineering Task Force* (IETF) *Manufacturer Usage Description* (MUD). Ellos presentan un *dataset* etiquetado con una variedad de ataques volumétricos, como DoS, inundaciones reflectantes de TCP/UDP/*Internet Control Message Protocol* (ICMP) y suplantación de identidad *Address Resolution Protocol* (ARP) para dispositivos IoT. También desarrollaron un sistema de aprendizaje automático que emplea dos técnicas de detección de valores atípicos, detección de límites y cadena de Markov, para determinar si un dispositivo IoT está involucrado en un ataque volumétrico o no y el flujo que contribuye al ataque. La monitorización de IoT se realizó para cada dispositivo a través de una combinación de telemetría *Software-defined networking* (SDN) de grano grueso (por dispositivo) y de grano fino (por flujo). El proceso se realizó en varias escalas de tiempo para detectar la desviación de los patrones de tráfico esperados en los flujos definidos por el perfil MUD del dispositivo.

8. **BoT-IoT** [46]: El *Cyber Range Lab del Australian Centre for Cyber Security* (ACCS), diseñó un *dataset* a través de un banco de pruebas de sensores de IoT simulado, con una combinación de sensores normales y tráfico de redes de *bots*. El *dataset* comprende una red simulada de servicios IoT, que contiene algunos sensores de temperatura, presión y humedad que se conectan mediante la suscripción y publicación de servicios del protocolo MQTT. El *dataset* contiene ataques DoS, DDoS, exfiltración de datos y ataques de registro. Para generar el Sistemas de Detección de Intrusiones, aplican algoritmos de SVM, Redes Neuronales Recurrentes (RNN) y *Long-Short Term Memory Recurrent Neural Network* (LSTM-RNN). Adicionalmente, otros autores explotan el BoT-IoT para desarrollar sus experimentos: Susilo et al. [47] construyó un sistema de detección de anomalías aplicando RF, SVM, MLP y Redes Neuronales Convolucionales (CNN), mientras que Alsamirimi et al. [48] implementó NB, RF, MLP, *Iterative Dichotomiser 3* (ID3), AdaBoost, Análisis Discriminante Cuadrático (QDA) y kNN.

9. **ToN-IoT** [49]: Los laboratorios Cyber Range Labs de UNSW Canberra construyeron un conjunto de datos MQTT, llamado ToN-IoT, que incluye fuentes de datos heterogéneas recopiladas de una red realista y a gran escala. En este se utilizaron más de 10 sensores IoT e IIoT, como sensores meteorológicos y de control industrial, para capturar datos de telemetría. El conjunto de dispositivos y sensores de IoT, como actuadores de IoT industrial, se conectaron a puertas de enlace MQTT para publicar y suscribirse a diversos *topics*, como medir la temperatura y la humedad.

Los conjuntos de datos se recopilaron en procesamiento paralelo para incluir varios eventos normales y de ataques cibernéticos de las redes de IoT. Sarhan et al. [50] buscaron estandarizar las técnicas para que pudieran aplicarse a cualquier conjunto de datos. Utilizaron seis modelos de aprendizaje automático: *Deep Feed Forward* (DFF), CNN, RNN, DT, LR y NB, y también exploraron tres algoritmos para la extracción de características, Análisis de Componentes Principales (PCA), Análisis Discriminante Lineal (LDA) y codificador automático.

10. **IoT-NID** [51]: El *IoT Network Intrusion Dataset* (IoT-NID) fue desarrollado por el Hacking and Countermeasure Research Lab. El *dataset* consta de 42 archivos de paquetes sin procesar en diferentes puntos de tiempo, utilizando dispositivos domésticos inteligentes típicos, combinando algunas computadoras portátiles y teléfonos inteligentes. El *dataset* incluye una buena cantidad de tráfico de red, y una de sus ventajas es la cantidad de ataques que presenta, *Man-in-the-Middle* (MiTM) (suplantación de identidad ARP), DoS (inundación SYN), exploración (exploración de host y puertos), inundación UDP/*acknowledgement* (ACK)/HTTP y *Mirai*.

11. **IoT-23** [52]: El dataset Aposemat IoT-23 publicado por el grupo de ciberseguridad del laboratorio Stratosphere Laboratory, el Centro de Inteligencia Artificial y la Facultad de Ingeniería Electrica de la Universidad Técnica Checa de Praga, ofrece un gran conjunto de datos que consta de veintitrés capturas de diferentes tráficos de red IoT. Estos escenarios se dividen en veinte capturas de red de dispositivos IoT infectados y tres capturas de red de tráfico de red de dispositivos IoT reales, a través de los protocolos HTTP, DNS, DHCP, Telnet, SSL e IRC. Las capturas de malware se ejecutan durante largos períodos, realizando diversos escenarios de ataque, incluidos *Mirai*, *Torii* y *Gagfyt*. El uso de este *dataset* para detectar anomalías de tráfico también fue empleado por Thamaraiselvi et al. [53], aplicando RF, NB, SVM y DT. Además, Aversano et al. [54] evalúa el tráfico IoT anómalo con un enfoque de *deep learning* capaz de identificar tanto el tráfico IoT normal como el malicioso, así como diferentes tipos de anomalías. Integran cinco conjuntos de datos diferentes, el BoT-IoT [46], el presentado por Sivanathan et al. [55] para el tráfico normal proveniente de varios dispositivos, el conjunto de datos de intrusión de red de IoT [51], una parte de IoT23 [52], que involucra principalmente tráfico malicioso, y un pequeño banco de pruebas de automatización [44] para sistemas de control industrial.

12. **IoTID20** [56]: Este dataset es una combinación de dispositivos IoT e interconexión de estructuras en un hogar inteligente. La red incluye computadoras portátiles,

tabletas, teléfonos inteligentes y una cámara . Los ataques son realizados a dos dispositivos que hacen de víctimas. El *dataset* contiene 8 tipos de ataques pertenecientes a DoS, *Mirai*, MiTM y *scan*. Los modelos de aprendizaje automático empleados para detección de anomalías son SVM, GaussianNB, LDA,LR, DT, RF y clasificadores ensemble. Las mejores tasas de detección alcanzadas fueron con RF y clasificadores ensemble.

13. **MedBIoT** [57]: Es otro dataset semisintético en IoT enfocado en la detección de ataques de botnets como *Mirai*, *BashLite* y *Torii*. Desarrollado en el Departamento de Ciencias del Software de la Universidad Tecnológica de Tallin, este conjunto de datos se obtiene de una red con más de 80 dispositivos, incluyendo tráfico normal y malicioso. Se implementaron los modelos de clasificación kNN, SVM, DT y RF, verificando la aplicabilidad del dataset propuesto.

14. **MQTTset** [58]: Vaccari et al. presentaron el MQTTset, un dataset centrado en el protocolo MQTT compuesto por dispositivos IoT de diferente naturaleza (por ejemplo, sensores de temperatura, humedad, movimiento, etc.), para simular un entorno de hogar/oficina/edificio inteligente. MQTTset combina el tráfico legítimo con diferentes tráficos maliciosos/de ataque, como ataques de denegación de servicio de inundación, inundación de publicación deMQTT, *SlowITe*, datos mal formados y autenticación de fuerza bruta, para replicar diferentes contextos, como domótica, monitorización de infraestructuras críticas o contextos industriales. En base a este escenario, realizan un Sistemas de Detección de Intrusiones implementando DT, RF, MLP, NN y Gaussian NB para identificar comportamientos maliciosos, con el fin de proteger el sistema de ataques.

15. **CCD-INID-V1** [59]: Liu et al. crean un dataset disponible públicamente usando sensores inteligentes en una red IoT. Los datos fueron recopilados en entornos de laboratorios y hogares inteligentes usando tableros de sensores Rainbow HAT instalados en Raspberry Pis. El dataset contiene cinco tipos de cyberataques: envenenamiento de tablas ARP, denegación de servicio ARP, UDP *floodattack*, ataques de fuerza bruta Hydra con protocolos *Asterisk* y *SlowLoris*. Para la clasificación de ataques, proponen un método híbrido que combina un modelo embebido de selección de características y CNN. Como método de detección de intrusiones implementan dos modelos: (a) Random Forest con CNN (RCNN) y (b) eXtreme Gradient Boosting con CNN (XCNN). Para comparar la efectividad de las técnicas propuestas con algoritmos tradicionales de ML, implementan kNN, NB,LR, y SVM. Una validación adicional es realizada, comparando los resultados obtenidos con el N_BaIoT

dataset [39] y el *dataset* CIRA-CIC-DoHBrw-2020 [60], y de esta forma examinar la efectividad de los modelos propuestos.

16. **IoT-Flock** [61]: Ghazanfar et al. crearon un *dataset* basado en hogares inteligentes, a partir de una herramienta generadora de tráfico IoT usando una sola máquina física. El conjunto de datos dado contiene cuatro tipos de sensores de monitorización ambiental, sensor de temperatura, humedad, luz y movimiento. El intercambio de paquetes se realiza a través del protocolo MQTT dentro de un entorno de red de área local inalámbrica (WLAN). Para demostrar cómo IoT-Flock puede ayudar a generar una solución de seguridad de IoT, el conjunto de datos se evaluó utilizando tres modelos de ML, NB, RF y kNN, obteniendo altas tasas de detección.

17. **IoT-RPL** [62]: Dhifallah et al. proponen un nuevo dataset IoT-RPL, desarrollado en el Laboratorio Hatem Bettaher (IRESCOMATH). El dataset presenta protocolos de comunicación inalámbrica como Internet Protocol v6 (IPV6), Internet Control Message Protocol v6 (ICMPV6), *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN), IEEE 802.15.4, CoAP y UDP. Fue construido inyectando cuatro tipos de ataques (*hello food attack*, *blackhole attack*, *Decreased rank attack* y *version number attack*). Para la detección de intrusiones se compararon cinco modelos de aprendizaje automático, SVM, DT, RF, kNN y MLP, consiguiendo tasas de precisión del 99 % en la mayoría de los casos.

18. **X-IIoTID** [63]: Este *dataset* es implementado usando como base el banco de pruebas desarrollado en la Universidad de New South Wales Sydney, el Brown-IIoTbed [64]. Este consta de una variedad de protocolos de conectividad M2M, máquina a humano (M2H) y humano a máquina (H2M), sensores, actuadores, varios dispositivos móviles y de TI, medios de acceso, interfaz de programación de aplicaciones (API), y estados que se implementan en los tres niveles de un sistema IIoT. Contiene una gran variedad de ataques en diferentes capas del sistema. Los datos incluyen tráfico normal y anómalo, capturado varias horas del día durante cuatro meses no continuos. Para validación del dataset se emplean algoritmos de DT, NB, kNN, SVM, LR, *Deep Neural Network* (DNN), *Gated Recurrent Unit* (GRU). Los mejores resultados son alcanzados con DT en todos los casos, en donde alcanza para clasificación binaria una Exactitud de 99.54 %.

19. **CIC IoT Dataset 2022** [65]: Este *dataset* fue creado a partir de una red IoT de 60 dispositivos aislados, simulando la actividad de un hogar inteligente con diferentes protocolos, tales como *IEEE 802.11*, *Zigbee* y *Z-Wave*. En sus experimentos crean seis distintos escenarios creando interacciones entre ellos. El *dataset* contiene

dos tipos de ataques diferentes, de inundación y fuerza bruta *Real Time Streaming Protocol* (RTSP). El de inuncdación se hizo usando *Low Orbit Ion Cannon* (LOIC) para realizar ataques DoS mediante protocolos HTTP, UDP, y TCP. El de fuerza bruta RTSP se hizo usando Hydra y Nmap, para encontrar las *Uniform Resource Locator* (URL) RTSP de las cámaras mediante fuerza bruta. Las URL descubiertas se usaron para ver el vídeo de la cámara en tiempo real. Los IDS fueron desarrollados utilizando 12 diversos clasificadores gaussian NB, DT, LDA, AdaBoost, *Ridge*, *Perceptron*, *Passive-Agressive*, eXtreme Gradient Boosting (XGBoost), kNN, RF, *Linear Support Vector Classification* (LSVC) y SGD , obteniendo los mejores resultados con XGBoost con una exactitud del 98.6 % y los clasificadores basados en árboles, DT y RF con 98.5 % de exactitud.

Para referencia, la tabla 2.1 lista los *datasets* de IoT encontrados, organizados por su año de publicación, mostrando el tipo de red, anomalía que presenta y su uso en algoritmos de aprendizaje automático. Todos los *datasets* mencionados aquí tienen una cantidad suficiente de muestras, en escenarios livianos, con una variedad de protocolos y diferentes tipos de anomalías. Se mencionan las principales aplicaciones que utilizan inteligencia artificial, lo que significa que todas están etiquetadas y en condiciones para la operación en IDS usando ML. La mayoría de los conjuntos de datos de IoT encontrados se basaron en entornos domésticos inteligentes o cámaras de videovigilancia, en entornos no específicos. Tres conjuntos de datos manejan entornos o rastros en plataformas IIoT y tres de ellos usan protocolos dedicados. Algunos de ellos introducen redes de sensores inalámbricos, sistemas de medición de temperatura, humedad, movimiento, etc., pero ninguno de los *datasets* mencionados proporciona redes de sensores de temperatura livianos de un CPD, con una cantidad aceptable de trazas en IoT, con una mezcla de anomalías.

Tabla 2.1: Tabla resumen de los *datasets* de IoT para ML-IDS

Nº	**Año**	**Dataset**	**Tipo de red**	**Tipos de anomalías**	**AplicaciónML**
1	2016	SCADA IIoT	capturas de tráfico MODBUS	*Metasploit, fingerprinting, unauthorized packets*	SVM, RF, kNN, k-means
2	2018	Doshi et al.	smartphones y portátiles	DDoS, *Mirai*	kNN, LSVM, DT, RF, NN
3	2018	N-BaIoT	cámaras IP, PCs y dispositivos IoT .	*Mirai botnet* y *BASHLITE malware*	*Deep autoencoders, encoders,* LR, ANN, RCNN, XCNN
4	2018	All eyes on you	hogar inteligente y smartphone	*timestamps* anómalos, DoS	LR, SVM, DT, RF, ANN
5	2019	Anthi	hogar inteligente	DoS	NB, *Bayesian Network, J48, Zero R, OneR, Simple Logistic,* SVM, MLP, RF.
6	2019	IIoT MODBUS	banco de pruebas usando MODBUS/TCP.	inundación ping TCP SYN y peticiones MODBUS	kNN, SVM, DT, RF y *autoencoder*
7	2019	UNSW	hogar inteligente	DoS, inundación TCP/UDP/ICMP, y ARP spoofing	Detección de límites y cadenas de Markov
8	2019	BoT-IoT	hogar inteligente MQTT	DoS, DDoS, *data exfiltration*, *keylogging*	SVM, RNN y LSTM-RNN, *autoencoder*, RF, SVM, CNN, NB, RF, MLP, ID3, AdaBoost, QDA, kNN
9	2019	ToN-IoT	sensores industriales y de temperatura MQTT	ciberataques desde redes IoT	DFF, CNN, RNN, DT, LR, NB y autoencoder
10	2019	IoT-NID	hogar inteligente, portátiles y smartphones	*Mirai*, MiTM, DoS, *Scanning*, etc.	*autoencoder*
11	2020	IoT-23	hogar inteligente (dispositivos IoT reales)	*Mirai, Torii y Gagfyt*	RF, NB, SVM, DT, *autoencoder*
12	2020	IoTID20	hogar inteligente	inundacion Syn, HTTP y UDP ARP *Spoofing*, fuerza bruta	SVM, Gaussian NB, LDA, LR, DT, RF y clasificadores ensemble
13	2020	MedBIoT	IoT (por ejemplo, ventiladores, luces, etc.)	*Mirai, BashLite, Torii*	kNN, SVM, DT y RF.
14	2020	MQTTset	sensores de movimiento, temperatura y humedad (MQTT)	inundación DoS, inundación de publicación MQTT , *SlowITe*, *malformed data*, y autenticación con fuerza bruta	DT, RF, MLP, NN y Gaussian NB
15	2021	CCD-INID-V1	sensores inteligentes en una red IoT	envenenamiento ARP, ARPDoS,inundación UDP, fuerza bruta con Hydra con protocolo Asterisk, *SlowLoris*	RCNN, XCNN, kNN, NB, LR , SVM
16	2020	IoT-Flock	hogar inteligente (MQTT/CoAP)	MQTT Packet Crafting Attack, inundación de publicación MQTT, filtración CoAP y segmentación CoAP	NB, RF y kNN
17	2021	IoT-RPL	nodos simulados	*Hello food, blackhole, decreased rank y version number*	SVM, DT, RF, kNN y MLP
18	2022	X-IIoTID	IIoT	*Weaponization, exploitation, lateral movement, exfiltration, tampering, crypto-Ransomware, ransom denial of service*	DT, NB, kNN, SVM, LR, DNN, GRU
19	2022	CIC IoT Dataset 2022	redes de sensores inalámbricos	inundación y fuerza bruta RTSP	NB, DT, LDA, AdaBoost, *Ridge, Perceptron, Passive-Agressive*, XGBoost, kNN,RF, LSVC y SGD

2.3. Modelado matemático de la temperatura

El modelado matemático es una de las formas más eficientes de conocer y comprender el funcionamiento de los sistemas, y predecir su comportamiento. Suele ser útil cuando contiene muchos componentes y numerosas interacciones como puede ocurrir si se trata de conjuntos bastante complejos y de gran tamaño. En este caso concreto, la idea era representar un conjunto de datos real con cierto grado de precisión y en la forma más completa posible. Esta sección presenta la recopilación de la información de técnicas empleadas para modelado de sensores de temperatura en entornos similares.

Nikolova et. al. [66] generan curvas de respuesta a sensores analógicos y proponen una metodología para ajustar datos de sensores a modelos específicos mediante el uso de librerías de MATLAB. Los datos de voltaje recibidos de los sensores se ajustan a datos exponenciales, series de Fourier, series de potencia, gausianas, polinomiales, suma de sinusoidales y distribuciones Weibull, usando bondad de ajuste. Dependiendo de la forma y especificidad del sensor verifican el uso de lo modelos estadísticos, para generar una serie de ecuaciones para termopares, detectores de temperatura resistivos y termistores.

Analizando otros dominios, en agricultura, los métodos estadísticos usados en el análisis de datos de oxígeno y temperatura en compostaje, han sido basados en matemáticas lineales, test de *t-student* y análisis de varianza con ANOVA [67]. Shouhai et al. [68] realizan un análisis estadístico del modelado de la temperatura en compostaje utilizando series de tiempo con modelos matemáticos no lineales. Ellos presentan un modelo de Gompertz modificado y realizan una descripción matemática de funciones tales como función logística, de Richart y Weibull, para describir series de tiempo en compostaje. La bondad de ajuste del modelo fue evaluado usando *R-squared* (R2) *statistic*.

Otro método muy empleado para predicción y análisis de series de tiempo es el Modelo autorregresivo de media móvil (ARMA). Este es empleado por Flores et al. para el diseño de modelos de predicción de fenómenos naturales para plantas energéticas [69].

Una aproximación relacionada con el análisis de datos de sensores de temperatura, es el realizado por Bhandari et al. [70]. Ellos modelan el fenómeno de la temperatura como un proceso estocástico y lo analizan haciendo uso de un marco de modelado de series de tiempo usando Modelo autorregresivo integrado de media móvil (ARIMA), el cual determina cómo la predicción a corto plazo de temperatura futura se ve afectada por intervalos de muestreo y técnicas de extrapolación. Los modelos ARIMA son usados para pronosticar los días restantes de datos. Para cada tasa de muestreo, el modelo ARIMA se entrena en tres días de datos y luego es capaz de predecir hasta dos horas en adelante desde ese punto.

El método de descomposición con tendencia estacional utilizando el método LOESS (STL) es un importante enfoque aplicado al análisis y predicción de series de tiempo [71]. Xing Huo et al.[72] proponen un sistema de predicción de temperatura en ciudades de China usando descomposición con STL. Este método también es sido aplicado a campos de datos de teledetección, entre otros [73] [74].

Capítulo 3

Sensores de temperatura

Para la construcción del escenario virtual, se tomó como modelo la arquitectura de redes de sensores de temperatura de un CPD real. La primera sección de este capítulo presenta una visión general de los sensores que lo componen, estructura de la red, interacción entre los sensores, distribución y localización.

El entorno simulado necesitaba igualmente una red de sensores que enviará temperaturas ajustadas al comportamiento real de la red, pero generadas con cierto grado de aleatoriedad. Del escenario real se obtuvo un conjunto de datos con 14 días de captura de los valores de temperatura de los sensores, los cuales fueron analizados y ajustados para determinar el modelo a aplicar en el entorno virtual. El desarrollo del estudio realizado del conjunto de datos y la definición del modelo se expone en la sección 3.2.

3.1. Infraestructura física del CPD

El CPD es una instalación física centralizada que aloja equipos de tecnologías de la información (TI), en donde el promedio apropiado de temperatura para una sala con servidores oscila entre los 18°C y los 23°C, aproximadamente. El incremento de estas temperaturas puede generar mal funcionamiento de los dispositivos o incluso romper el sistema. El CPD localizado en el Centro de Investigación en Tecnologías de la Información y las Comunicaciones de la UDC [11], incorpora un sistema de refrigeración que consiste en tres elementos con sensores: los Racks, las regletas *Power Distributor Unit* (PDU), y las máquinas de refrigeración (InRow).

La posición especifica y descripción de cada sensor es presentada en la figura 3.1. La

función de cada sensor es determinada por su localización.

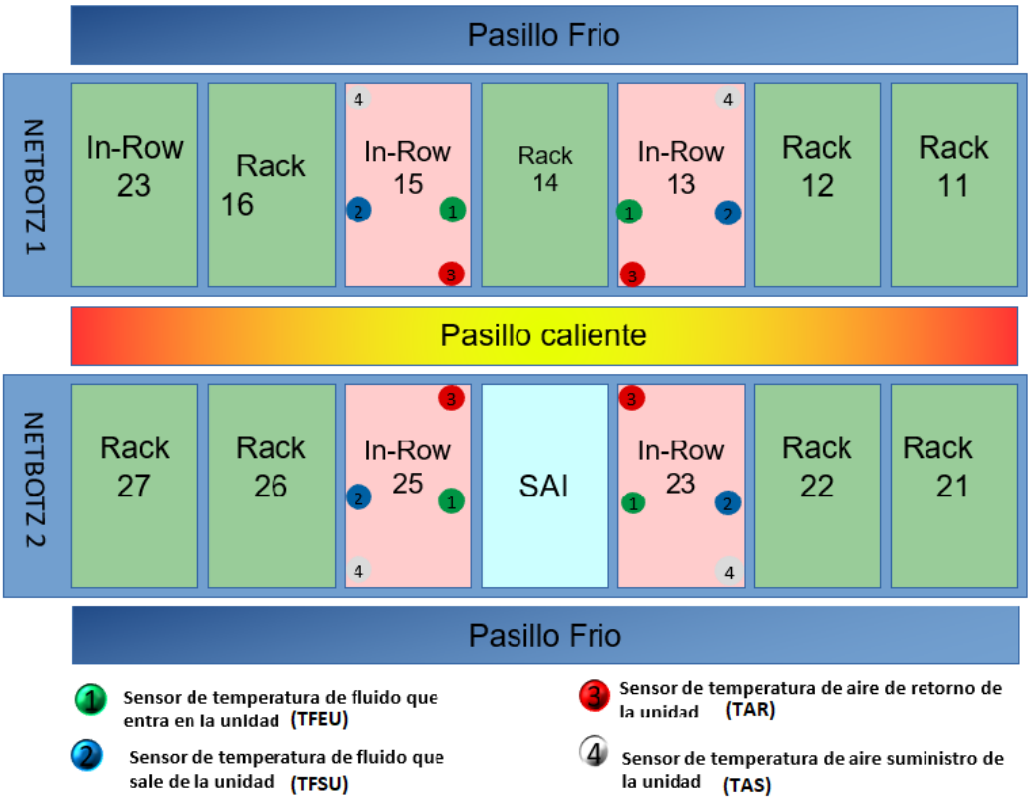

Figura 3.1: Arquitectura fisica del CPD

El NETBOTZ, es un producto de monitorización desarrollado por *American Power Conversion* (APC) por Schneider Electric[75]. Es un dispositivo que supervisa las variables ambientales críticas del CPD como temperatura, humedad, apertura de puertas, detección de fluidos, detección de corrientes de aire, detección de ruidos, de movimiento y grabación de vídeo. Contiene un módulo de cámara y otro de sensores integrados que permiten detectar anomalías en el ambiente. En el CPD existen dos dispositivos NETBOTZ, el NETBOTZ 1, con dirección IP 10.6.56.58, tiene 5 Racks (11, 12, 14, 16 y 17) y el NETBOTZ 2, con dirección IP 10.6.56.49, tiene 4 Racks (21, 22, 26 y 27). Cada Rack a su vez tiene dos sensores, uno en el respaldo, el cual es localizado en el lado del pasillo caliente y en otro en el frente, localizado en el pasillo frío. Los elementos más importantes del sistema son las unidades de frío o InRows (13, 15, 23 y 25), dispositivos responsables de medir la temperatura del aire de los pasillos y la del líquido del circuito de refrigeración. En el escenario propuesto, solamente los sensores de las InRows serán considerados, dado que los demás sensores mantienen valores estáticos y, por lo tanto, no son significativos.

Cada InRow tiene control sobre 4 tipos de sensores, sensor de Temperatura de Fluido que Entra en la Unidad (TFEU), sensor de Temperatura de Fluido que Sale de la Unidad

(TFSU), sensor de Temperatura de Aire de Retorno (TAR) y sensor de Temperatura de Aire de Suministro (TAS). El propósito de cada uno ellos se explica a continuación:

- TFEU: Estos sensores miden la temperatura del líquido que entra al circuito de refrigeración.
- TFSU: Estos sensores miden la temperatura del líquido que abandona el circuito de refrigeración.
- TAR: Estos dispositivos toman los valores de la temperatura del aire que proviene del pasillo caliente. Las medidas dadas por este sensor son equivalentes a las entregadas por los sensores traseros de los Racks.
- TAS: Estos sensores miden la temperatura que proviene del pasillo frío. Normalmente las temperaturas medidas de estos sensores son relativos a las medidas obtenidas por los sensores frontales de los Racks.

3.2. Modelado matemático de los sensores

Para poder diseñar el modelo matemático que simule el funcionamiento real de redes de sensores de temperatura, se tomo una muestra correspondiente a 14 días de actividad de todos los sensores de las InRows presentes en el CPD. Los sensores envían datos de temperatura cada 5 minutos. Las señales obtenidas de los sensores se presentan en la figura 3.2. El comportamiento de las señales depende de la carga de trabajo de los procesadores. A mayor carga de trabajo mayor será la temperatura alcanzada. Algunos cambios externos, tales como abrir una puerta del pasillo, pueden alterar los valores de temperatura medidos.

La tabla 3.1 presenta un resumen estadístico de los valores obtenidos por los sensores de la InRow. En esta se observa cómo el promedio de TAS es de 20ºC, 27ºC para el sensor de TAR, 11ºC para el sensor de TFEU y 15ºC para el sensor de TFSU. La mayor desviación estándar se presenta en el sensor de TFEU con un valor alrededor de 1ºC.

Posteriormente, se lleva a cabo un análisis de correlación entre los sensores. El comportamiento encontrado para todas las InRows es semejante, por lo que para la exposición se tomará una InRow seleccionada aleatoriamente. La figura 3.3 presenta los resultados obtenidos para los sensores de la InRow 25. El triángulo superior muestra los valores del coeficiente de correlación de Pearson. Los valores de distribución se presentan en la diagonal, mientras el triángulo inferior presenta los diagramas de dispersión entre los sensores.

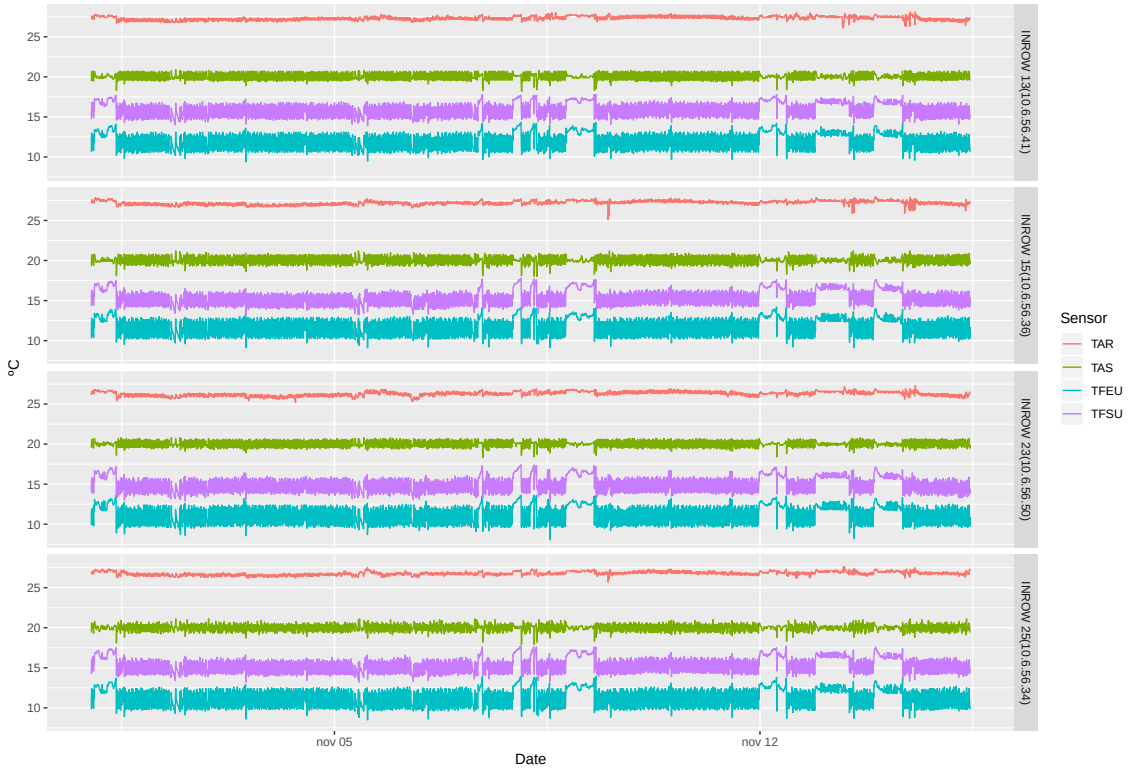

Figura 3.2: Señales de los sensores en las InRows.

Tabla 3.1: Resumen estadístico para los valores de temperatura.

Dispositivo	Sensor	media	SD	min	max	mediana
INROW 13 (10.6.56.41)	TAS	20.00	0.38	18.20	20.90	19.90
	TAR	27.33	0.24	26.10	28.10	27.30
	TFEU	11.94	0.93	9.40	14.30	11.80
	TFSU	15.89	0.79	13.90	17.80	15.80
INROW 15 (10.6.56.36)	TAS	20.00	0.42	18.00	21.20	19.90
	TAR	27.20	0.25	25.10	27.90	27.20
	TFEU	11.69	0.98	9.10	14.20	11.60
	TFSU	15.37	0.89	13.20	17.80	15.20
INROW 23 (10.6.56.50)	TAS	20.00	0.34	18.30	20.80	20.00
	TAR	26.29	0.27	25.20	27.30	26.30
	TFEU	11.14	1.00	8.10	13.60	11.00
	TFSU	14.95	0.88	12.80	17.40	14.80
INROW 25 (10.6.56.34)	TAS	20.00	0.37	17.90	21.10	20.10
	TAR	26.79	0.22	25.70	27.60	26.80
	TFEU	11.24	1.00	8.50	13.90	11.10
	TFSU	15.34	0.88	13.20	17.70	15.20

Como era de esperar, debido al funcionamiento del circuito de refrigeración, la correlación entre los sensores de fluido es siempre alta. Se presenta una pequeña correlación entre TAS y TFSU. Sin embargo, la TAR está débilmente relacionada con los demás sensores presentando cierto grado de independencia.

Los escenarios a diseñar requerían de datos de temperatura que se ajustaran a los suministrados por las InRows. Las limitaciones en la implementación presentadas por redes IoT, nos retaba a generar modelos matemáticos sencillos pero precisos con cargas computaciones muy bajas. Por este motivo se realizó la estimación del comportamiento

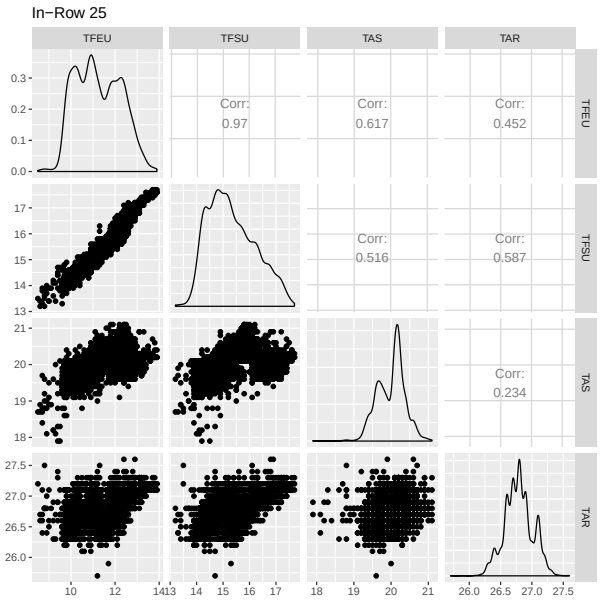

Figura 3.3: Función de distribución y correlación entre sensores en la InRow 25

cuantitativo del sistema, empleando diferentes modelos paramétricos y no-paramétricos. Estos análisis buscaban encontrar un modelo unificado que permitiera abarcar el comportamiento de todos los sensores al mismo modelo y pudiera ser implementado en cada uno de los dispositivos correspondientes a los sensores en el entorno virtualizado.

Se considera inicialmente encontrar la función densidad de probabilidad que se ajuste a los sensores de temperatura, usando pruebas de bondad de ajuste de Kolmogorov-Smirnov y de Anderson-Darling, para luego intentar ajustar los datos, vistos como series de tiempo, a métodos de regresiones simples. Ante la falta de convergencia en los resultados, se aplican métodos ARMA y ARIMA, que tienen que ser descartados debido a la complejidad de su implementación en redes IoT. Finalmente, se toman los modelos de descomposición tendencia estacional utilizando el método LOESS (STL) como una aproximación exitosa.

3.2.1. Bondad de ajuste

La prueba de bondad de ajuste consiste en encontrar la distribución de probabilidad que mejor se ajusta a una serie de datos. Se basa en la comparación de la función de distribución teórica propuesta por el modelo cuyo ajuste estamos evaluando con la función de distribución empírica de los datos. Si tenemos X_1,....,X_N una muestra de una variable aleatoria X, y una función densidad de probabilidad F(x) y se denota a SN(x) a la función de distribución empírica asociada a la muestra, se calcula a partir de la diferencia entre las funciones de distribución acumuladas teórica y observada. Existen diferentes tests para

ejecutar estas comparativas. Sin embargo, en este estudio se tomaron la prueba de bondad de ajuste de Kolmogorov-Smirnov y la prueba de bondad de ajuste de Anderson-Darling.

La prueba de Kolmogorov-Smirnov compara la función de distribución teórica con la empírica y calcula un valor de discrepancia máxima entre ambas distribuciones, proporcionando un p-valor asociado a la probabilidad de obtener una distribución que discrepe tanto como la observada si verdaderamente se hubiera obtenido una muestra aleatoria, de tamaño n, de una función de distribución.

La prueba de Anderson-Darling se basa en la diferencia de cuadrados entre las distribuciones otorgando una mayor relevancia a los datos existentes en las colas de la distribución. Se considera una prueba muy potente estadísticamente cuando se alude a pruebas basadas en las funciones de distribución empíricas [76]. Conviene tener en cuenta que la prueba Kolmogorov-Smirnov es más sensible a los valores cercanos a la mediana que a los extremos de la distribución, y la prueba de Anderson-Darling proporciona igual sensibilidad con valores extremos.

Ahora bien, los test de bondad de ajuste comparan los datos a distribuciones empíricas, en donde por lo general las poblaciones numéricas tienen distribuciones que pueden ser representadas muy fielmente por una curva normal apropiada. Aun cuando las variables individuales no estén normalmente distribuidas, las sumas y promedios de las variables en condiciones adecuadas tendrán de manera aproximada una distribución normal. Por esta razón la primera aproximación es usar los test con distribuciones normales. Sin embargo, el costo a pagar cuando se usa una distribución normal para caracterizar datos que no son normales es alto, ya que los resultados de las estimaciones obtenidas pueden ser engañosas.

Existen muchas situaciones prácticas en las cuales la variable de interés podría tener una distribución asimétrica, lo que se refleja en la inclinación a valores muy positivos o bien muy negativos. Si los datos tienen un sesgo positivo una opción común es ajustarlos a una distribución lognormal. Se dice que una variable aleatoria no negativa X tiene una distribución lognormal si la variable aleatoria $Y = \ln(X)$ tiene una distribución normal. La función de densidad de probabilidad resultante de una variable aleatoria es lognormal cuando el $\ln(X)$ está normalmente distribuido con parámetros μ y α. los parámetros μ y α no son la media y la desviación estándar de X sino de $\ln(X)$. Las distribuciones Gamma y Weibull se encuentran muy relacionadas con la distribución lognormal, dado que el cambio o modificación en los parámetros σ y μ generan cambios en la forma de la distribución haciéndola más o menos sesgada.

A priori, podemos asumir que las señales de temperatura siguen distribuciones normales. Sin embargo, hemos aplicado diferentes pruebas de bondad de ajuste para determinar qué distribución se ajusta mejor a los datos. Existen diferentes métodos para determinar la bondad de ajuste. En esta tesis se tomaron como primera aproximación distribuciones

Normal, Weibull, lognormal y Gamma, en los test de pruebas de Kolmogorov-Smirnov y Anderson-Darling para todos los sensores. Para estas pruebas se plantea la hipótesis nula (H_0), como que la muestra sigue la distribución definida, con un nivel de significancia de 0.05. Los resultados de los tests se presentan en la tabla 3.2 mostrando el p-valor asociado a la prueba y el estadístico de la misma (estadístico de Kolmogorov-Smirnov) representado por la letra D y (Anderson-Darling) representado por la An.

Tabla 3.2: Estadísticas test de bondad de ajuste

		Normal		**Lnorm**		**Gamma**		**Weibull**	
Test	**Sensor**	**D/An**	**p-valor**	**D/An**	**p-valor**	**D/An**	**p-valor**	**D/An**	**p-valor**
Kolmogorov-Smirnov	TAS	0.15595	$< 2{,}2 \times 10^{-16}$	0.15884	$< 2{,}2 \times 10^{-16}$	0.1579	$< 2{,}2 \times 10^{-16}$	0.12464	$< 2{,}2 \times 10^{-16}$
	TAR	0.11654	$< 2{,}2 \times 10^{-16}$	0.1149	$< 2{,}2 \times 10^{-16}$	0.11544	$< 2{,}2 \times 10^{-16}$	0.16636	$< 2{,}2 \times 10^{-16}$
	TFEU	0.082724	$< 2{,}2 \times 10^{-16}$	0.075811	$< 2{,}2 \times 10^{-16}$	0.075815	$< 2{,}2 \times 10^{-16}$	0.10402	$< 2{,}2 \times 10^{-16}$
	TFSU	0.0888	$< 2{,}2 \times 10^{-16}$	0.08	$< 2{,}2 \times 10^{-16}$	0.0821	$< 2{,}2 \times 10^{-16}$	0.1165	$< 2{,}2 \times 10^{-16}$
Anderson-Darling	TAR	45,533	$1{,}47 \times 10^{-7}$	45,228	$1{,}47 \times 10^{-7}$	45,322	$1{,}47 \times 10^{-7}$	94,882	$1{,}47 \times 10^{-7}$
	TAS	47,257	$1{,}47 \times 10^{-7}$	49,244	$1{,}47 \times 10^{-7}$	48,562	$1{,}47 \times 10^{-7}$	51,504	$1{,}47 \times 10^{-7}$
	TFEU	38,602	$1{,}47 \times 10^{-7}$	35,593	$1{,}47 \times 10^{-7}$	36,201	$1{,}47 \times 10^{-7}$	53,252	$1{,}47 \times 10^{-7}$
	TFSU	41,351	$1{,}47 \times 10^{-7}$	33,063	$1{,}47 \times 10^{-7}$	35,627	$1{,}47 \times 10^{-7}$	92,955	$1{,}47 \times 10^{-7}$

Debido a los p-valores obtenidos, la decisión es rechazar la hipótesis nula, por el nivel de significancia. En otras palabras, los datos no se ajustan a las distribuciones consideradas, y se ha decidido buscar el ajuste mediante series de tiempo.

3.2.2. Series de tiempo

Cualquier señal que es examinada secuencialmente sobre intervalos regulares de tiempo es considerada una serie temporal. El objetivo es estimar cómo la secuencia de observaciones continuará en el futuro, encontrando el método que haga un buen ajuste a los datos seleccionados. La finalidad es entonces modelar las características en series de tiempo considerando capturar la tendencia y el comportamiento estacional observado, manejar la varianza no constante (heterocedasticidad) y encontrar un modelo estadístico que sea capaz de reproducir esa "inercia" o autocorrelación que tienen muchas variables temporales.

Los valores presentes en series de tiempo pueden exhibir una gran variedad de patrones, y es a veces útil dividir una serie de tiempo en varios componentes que representen cada uno de ellos, una categoría de patrón subyacente. Una serie temporal denotada como Y_t, se supone puede descomponerse de modo aditivo como:

$$Y_t = S_t + T_t + \epsilon_t \tag{3.1}$$

Explicado de otra manera, para cada punto (Y_t) en un instante de tiempo t, una serie

de tiempo puede ser expresada como una suma o un producto de 3 componentes llamados estacionalidad (S_t), tendencia (T_t) y error (ϵ_t), también conocido como ruido blanco.

La tendencia estima el movimiento a largo plazo de una serie, independientemente de otros componentes irregulares, la componente estacional son oscilaciones a corto plazo que se repiten sucesivamente, comportamiento periódico de la serie, y el error recoge los movimientos transitorios e irregulares de esta.

3.2.2.1. Regresiones

La regresión es una técnica de modelado estadístico que se emplea para describir una variable de respuesta continua como una función de una o varias variables predictoras. Las técnicas de regresión lineal permiten crear un modelo lineal. Este modelo describe la relación entre una variable dependiente y (también conocida como la respuesta) como una función de una o varias variables independientes (denominadas predictores).

En el caso más simple, el modelo de regresión permite generar una relación lineal entre la variable de predicción y y una única variable predictora x. La intención es ajustar los datos en términos lineales, cuadráticos, cúbicos y exponenciales. Los resultados obtenidos del proceso se muestran en la tabla 3.3.

Tabla 3.3: Resultados de las regresiones.

Sensor	**Estadístico**	**Lineal**	**Cuadrática**	**Cúbica**	**Exponencial**
TAS	R2-ajustado	-0.0002	-0.0004	-0.0007	-0.0002
	ECM	0.1392	0.1392	0.1392	0.00035
	estadísticas F	0.000054	0.0034	0.0071	0.0017
	valor-p	0.9814	0.9966	0.9992	0.9663
TAR	R2-ajustado	0.1516	0.1515	0.2462	0.1521
	ECM	0.041	0.041	0.03698	5.8e-5
	estadísticas F	742.8	371.6	453.1	745.5
	valor-p	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$
TFEU	R2-ajustado	0.0168	0.01823	0.0541	0.01627
	ECM	0.99	0.99	0.95	0.0078
	estadisticas F	71.93	39.54	80.28	69.68
	valor-p	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$
TFSU	R2-ajustado	0.039	0.04154	0.11	0.04
	ECM	0.75	0.748	0.6945	0.0031
	estadisticas F	171.5	90.97	172.9	171.8
	valor-p	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$	$2{,}2 \times 10^{-16}$

3.2.2.2. Modelos ARIMA

Si el objetivo es predecir a corto plazo, se adopta otra aproximación que consiste inicialmente en transformar los datos para introducir estacionariedad en media y/o varianza para poder encontrar qué estructura de autocorrelación (modelo estocástico) explica

mejor esa serie ya transformada. Para ello, se definen los modelos ARIMA (familia de modelos estadísticos que son capaces de reproducir una gran variedad de series estacionarias). Defínase por estacionariedad a la propiedad estadística de una serie de permanecer constante a lo largo del tiempo. Los modelos ARIMA proporcionan otro enfoque para la predicción de series de tiempo y proporciona perspectivas complementarias al problema. Los modelos ARIMA tienen como objetivo describir las autocorrelaciones en los datos, además de observar la gráfica de tiempo de los datos. A partir de las autocovarianzas, se calcula lo que se denomina Función de autocorrelación (ACF) de un modelo, cuyos valores no son más que coeficientes simples de autocorrelación de distinto orden. Además de la ACF, se suelen calcular las autocorrelaciones parciales o valores de la Función de autocorrelación parcial (PACF) del modelo. Una autocorrelación parcial con un retraso k es una correlación que resulta después de eliminar los efectos de cualquier correlación debida a los términos a cortos retrasos. La diferenciación es un método que consiste en transformar una serie de tiempo no estacionaria en una estacionaria. Este es un paso importante en la preparación de los datos cuando se usan modelos ARIMA.

El diagrama de autocorrelación, que se muestra en la figura 3.4, nos permite saber cómo se correlaciona la serie de tiempo dada consigo misma. Para una serie de tiempo estacionaria, el ACF caerá a cero relativamente rápido, como lo hace el sensor TAS.

Como se observa en la figura 3.4 (b), la autocorrelación de residuos para el sensor TAS muestra que es una serie estacionaria. De esta forma el modelo ARMA puede ser usado. Esto quiere decir que para los otros sensores es necesario diferenciar las series usando ARIMA.

Los resultados obtenidos de los cálculos de modelos ARIMA son mostrados en la tabla 3.4,presentando los resultados de σ^2 , *log likelihood* , Criterio de información de Akaike (AIC) y Criterio de información Bayesiano (BIC).

Tabla 3.4: Resultados a modelos ARIMA

	TAR	**TAS**	**TFEU**	**TFSU**
ARIMA Model	(3,1,5)	(5,0,2) with non-zero mean	(5,1,2)	(5,1,1)
σ^2 estimado	0.01076	0.07068	0.3367	0.3367
log likelihood	3528.83	-388.69	-3522.97	-3522.97
AIC	-7039.65	795.38	4981.6	7059.94
BIC	-6982.65	852.39	5032.24	7104.05

Al ser diseñado para ser implementado en redes IoT es necesario que el sistema cuente con modelos de bajo consumo energético y baja complejidad computacional. A pesar de hacer uso del principio de la parsimonia, en donde el modelo ajusta la información disponible sin usar coeficientes innecesarios (es decir, siempre se opta por el modelo con

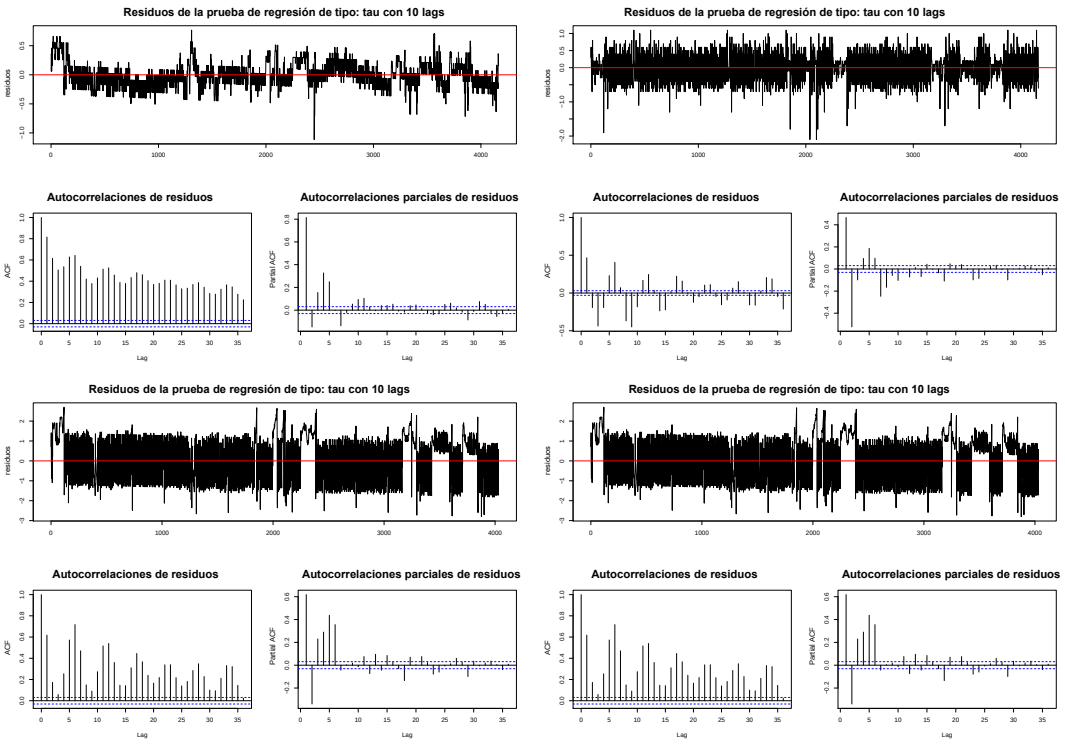

Figura 3.4: Señal, autocorrelaciones y autocorrelaciones parciales de residuos: (**a**) Sensor TAR. (**b**)sensor TAS.(**c**) sensor TFEU. (**d**) sensor TFSU.

el menor número de coeficientes), el gran número de parámetros obtenidos con auto ARIMA requiere una carga computacional excesivamente alta para ser implementada en dispositivos IoT ligeros y por esta razón se descarta este modelo.

3.2.2.3. Descomposición STL

STL es un método versátil y robusto para descomponer series de tiempo. Es un acrónimo de "tendencia estacional utilizando el método LOESS", mientras que Loess es un método para estimar relaciones no lineales. El método de descomposición con STL fue desarrollado por Cleveland et al. [71].

La descomposición STL tiene varias ventajas sobre otros métodos clásicos de descomposición [77] y maneja cualquier tipo de estacionalidad. La componente estacional puede cambiar en el tiempo, siendo la tasa de cambio y la suavidad del ciclo de tendencia controlado por el usuario. Este puede ser robusto con los outliers (ej. el usuario puede especificar una descomposición robusta) de esta forma observaciones inusuales ocasionales no afectarán a las estimaciones de los componentes estacionales y de tendencia, pero si afectarán los componentes remanentes.

En cada uno de los sensores tres vectores de datos son extraídos, uno que corresponde

a la tendencia, uno a la estacionalidad y el último a los valores aleatorios. A modo de ejemplo, la figura 3.5 presenta la descomposición STL de la señal del sensor TAS, con sus componentes tendencia, estacional y aleatoria. En caso de querer un mayor número de muestras a las que se tienen, se hacen predicciones sobre las líneas de tendencia, manteniendo la estacionalidad y generando nuevos valores aleatorios, lo que garantiza un comportamiento acorde al del sensor real.

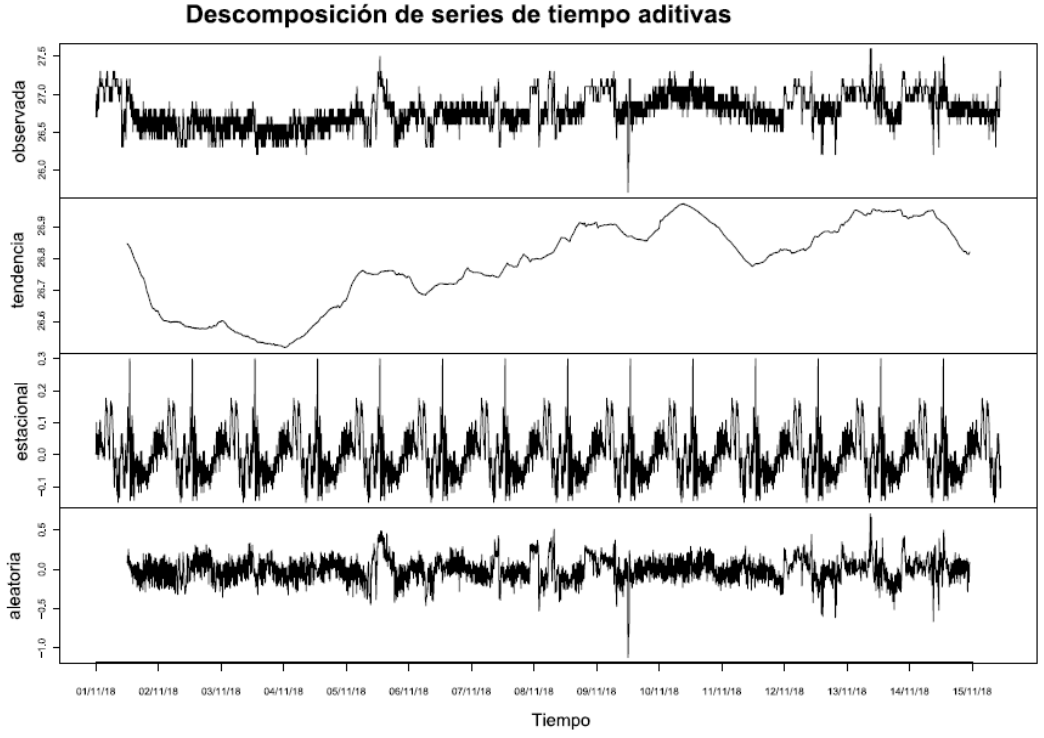

Figura 3.5: Descomposición STL para el sensor TAS.

Tras este exhaustivo análisis, y debido a las características antes mencionadas, el entorno simulado implementa el uso del modelo de descomposición STL para generar los datos de temperatura. Al igual que en el CPD real, los sensores virtuales envían datos de temperatura cada 5 minutos, tomando una frecuencia de muestreo de 288 muestras al día.

Capítulo 4

Construcción del dataset DAD

Aunque se necesitan buenos datasets para validar y evaluar los IDS, generar dichos conjuntos de datos es una tarea que requiere mucho tiempo [10]. Para la construcción de los entornos se seleccionaron dos de los protocolos más empleados y que presentan mayores prestaciones y comparaciones en la literatura con arquitecturas distintas [3].

Este capítulo presenta el diseño, construcción y generación del dataset denominado **DAD** *Annotated Dataset for Anomaly Detection in a Data Center with IoT Sensors* [12], el cual implementa MQTT como protocolo IoT en la capa de aplicación.

El entorno simulado, basado en el entorno real descrito en el capítulo 3, consiste en una red de cuatro InRows, con cuatro sensores en cada uno de ellos. La InRow 13 es comprometida, inyectando tráfico anómalo sobre el *payload* de los mensajes de los protocolos IoT, en cinco de los siete días, en donde el tráfico anormal es estadísticamente diferente del tráfico normal.

Para el dato de la temperatura a enviar, se ha codificado el algoritmo que implementa la serie temporal descrita en la sección 3.2, ecuación 3.1. Cada mensaje de temperatura es calculado a partir de la suma del valor del patrón estacional de ese nodo, más el valor del patrón de tendencia de la temperatura y un valor aleatorio que simula el funcionamiento del entorno real.

Se presentan tres diferentes tipos de anomalías en los entornos, en donde el comportamiento del nodo puede verse afectado de la siguiente manera:

- **Interceptación**: borrado aleatorio de algunos paquetes enviados.
- **Modificación**: alteración al dato de temperatura, modificando el comportamiento normal del sensor.

- **Duplicación**: envío de más mensajes de los previstos inicialmente.

El etiquetado de tráfico se realiza a nivel de flujo. Esto quiere decir que todos los paquetes pertenecientes a un flujo anormal son marcados como anomalía.

A continuación, se hará una breve descripción del protocolo MQTT, para en la sección 4.2 explicar la arquitectura del escenario virtual empleado y posteriormente la generación del dataset.

4.1. Protocolo MQTT

Fue diseñado por Andy Stanford-Clark *International Business Machines Corporation* (IBM) y Arlen Nipper en 1999 para conectar sistemas de telemetría de oleoductos por satélite. Aunque comenzó como un protocolo propietario, se lanzó libre en 2010 y se convirtió en un estándar OASIS en 2014. MQTT significa *MQ Telemetry Transport*, pero anteriormente se conocía como Message Queuing Telemetry Transport. Hay dos variantes diferentes de MQTT, con su versionado MQTT v3.1.0, MQTT v3.1.1 (de uso común), la última versión de MQTT (v5) aprobada en enero de 2018 (actualmente de uso limitado) y MQTT para redes de sensores (MQTT-SN) (poco difundida) [78]. Esta última, realiza un mapeo UDP de MQTT, que agrega soporte de intermediario para indexar nombres de *topics*, y pretende reducir las limitaciones de MQTT existentes para IoT [3].

MQTT Es un protocolo asíncrono que tiene como objetivo comunicaciones M2M de publicación/suscripción que se ejecuta en la parte superior de la pila TCP. Consiste en tres componentes: suscriptor, publicador y un servidor central que se denomina *broker*. Cada cliente puede ser un publicador que envía información al *broker* en un *topic* específico o/y un suscriptor que recibe mensajes automáticos cada vez que hay una nueva actualización en el *topic* al que está suscrito. Un dispositivo interesado se registraría como suscriptor de *topics* específicos para que el *broker* le informe cuando los publicadores publican *topics* de interés. El publicador actúa como generador de datos de interés. El publicador envía al *broker* que retransmite la información a las entidades interesadas (suscriptores).

Uno de los componentes más importantes del protocolo MQTT es la definición y tipología de los mensajes. Cada mensaje consta de 3 partes, como se aprecia en la figura 4.1:

- Cabecera fija. Es un conjunto de campos obligatorios y ocupa de 2 a 5 bytes. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje.

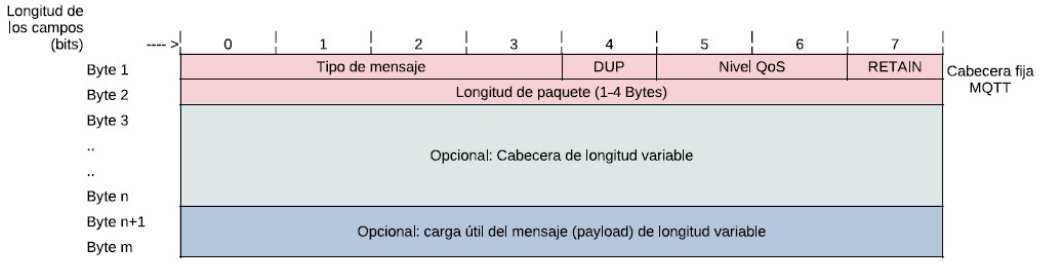

Figura 4.1: Formato del mensaje MQTT.

- Cabecera variable. Es opcional, de longitud variable y contiene información adicional necesaria en ciertos mensajes o situaciones.
- Contenido (*payload*). De longitud variable, corresponde a la carga útil del mensaje. Puede tener un máximo de 256 MB, aunque en implementaciones reales el máximo es de 2 a 4 kB.

En este formato, el valor del campo Tipo de mensaje indica una variedad de mensajes que incluyen *CONNECT* (1), *CONNACK* (2), *PUBLISH* (3), *SUSCRIBE (8)*, etc. El flag DUP indica que el mensaje está duplicado y que el receptor puede haberlo recibido antes. El campo Nivel de Calidad de servicio (QoS) identifica tres niveles de QoS para garantizar la entrega de mensajes de publicación. El campo *Retain* informa al servidor que retenga el último mensaje de publicación recibido y lo envíe a los nuevos suscriptores como primer mensaje. El campo Longitud restante muestra la longitud restante del mensaje, es decir, la longitud de las partes opcionales.

MQTT garantiza la confiabilidad al brindar la opción de tres niveles de QoS, entendido como la forma de gestionar la robustez del envío de mensajes al cliente ante fallos (por ejemplo, de conectividad) [79]:

1. *Fire and forget*: un mensaje se envía una vez y no se requiere acuse de recibo.
2. *Delivered at least once*: un mensaje se envía al menos una vez y es requerido un acuse de recibo.
3. *Delivered exactly once*: se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Aunque MQTT se ejecuta sobre TCP, está diseñado para tener una sobrecarga baja en comparación a otros protocolos de capa de aplicación basados en TCP. Además, la arquitectura publicación/suscripción proporciona flexibilidad de transición y simplicidad de implementación, lo que permite conectar dispositivos y redes integrados con aplicaciones y middleware. MQTT permite posibilidades de comunicación de uno a uno, de uno

a muchos y de muchos a muchos. El protocolo MQTT está diseñado para proporcionar enrutamiento para dispositivos pequeños y económicos en redes vulnerables, y usar el ancho de banda y el uso de la batería con moderación, lo que lo hace un protocolo de mensajería ideal para las comunicaciones IoT y M2M.

4.2. Descripción del escenario virtual

La infraestructura virtual para el entorno MQTT se construyó usando cinco nodos virtuales VMware ESXi 6.5 configurados con Ubuntu 18.04. Presenta nodos interconectados mediante los vSwitch, pero todos los nodos están aislados permitiendo realizar capturas del tráfico únicamente de la red IoT.

En este escenario, en una de las máquinas se ha instalado el *broker* mosquitto [80]. Este se encarga de albergar y desplegar el código que implementa la generación de comunicaciones MQTT con los clientes que corresponden a las InRows del CPD. El proceso inicializa los clientes, distribuye el código asociado a cada nodo que implementa la serie temporal asignada, inicializa cada nodo y controla su ejecución hasta que finaliza el envío de paquetes al *broker*. En cada uno de los cuatro nodos cliente lanza un proceso que simula cada uno de los cuatro sensores que componen las unidades frigoríficas. Estos procesos reciben un identificador correlacionado con el identificador de la unidad de frío (InRow). Por ejemplo, la unidad de frío 13 tiene los procesos 131, 132, 133 y 134, los cuales simulan el funcionamiento de la TAS, TAR, TFEU y TFSU, respectivamente. Las muestras generadas por los sensores se envían cada cinco minutos al *broker* mediante un mensaje MQTT que contiene su identificador de nodo *clientId*, una vez conectado el sensor para la publicación del tópico correspondiente.

En el *broker*, a través de tcpdump [81], se realiza una captura del tráfico intercambiado con los nodos clientes, para luego anotarlo mediante la información que tienen los clientes sobre el envío de los tokens en la conexión, permitiendo marcar aquellos tokens que pertenecen a situaciones anómalas o de ataque.

4.3. Generación del Dataset

En este dataset se han configurado cuatro nodos cliente y un nodo *broker* central. Los nodos cliente con cuatro procesos cada uno, simulan el funcionamiento de cada sensor enviando muestras de temperatura al *broker*. Para cada uno de estos sensores se ha

implementado un código que permite la publicación de mensajes MQTT.

A continuación, se muestra el pseudo código para estos procesos:

Algorithm 1 Pseudo código del envío de mensajes

Inicializar Cliente (*clientId*, *topic*)
Configurar el número de *tokens*, el número de segundos de espera y el identificador de patrón.
Crear el cliente MQTT (*brokerAddress*, *clientId*, $PERSISTENCE$)
Establecer conexiones con el Broker (*communication options*)
for número de tokens **do**
    Calcular la temperatura a enviar de acuerdo al patrón.
    Post mensaje MQTT con *topic*
    Esperar por la confirmación de envío del token
end for
Desconectar cliente MQTT
Destruir cliente MQTT

Mediante un proceso distribuido y sincronizado con un servidor *Network Time Protocol* (NTP), se crea una red de sensores donde se recoge la publicación de muestras en el *broker*. La captura de los paquetes se realiza en las interfaces de red del nodo en el que se ejecuta el *broker*.

Este proceso permite realizar las tareas de:

- Despliegue y configuración de sensores en los nodos.
- Verificación del estado de ejecución de la configuración de sensores en los nodos.
- Trazabilidad del funcionamiento del sensor.
- Inicio de la ejecución de diferentes procesos.
- Parametrización del número de tokens a enviar y la periodicidad de los envíos.
- Parametrización del identificador a enviar como clientId.
- Generación de la muestra que corresponde al patrón del día de la semana.
- Cancelación y reinicio de la simulación en cualquier momento durante la operación.
- Programación asíncrona de su funcionamiento.

Finalmente, una vez el tráfico ha sido capturado en el *broker*, con la información que ha sido generada previamente en los nodos cliente, se realiza el proceso de etiquetado del *dataset* a nivel de paquete indicando el tipo de token (normal o anómalo). DAD compromete algunos tokens con esas modificaciones y otros sin ninguna de ellas. Posteriormente

se identifican los paquetes que forman parte de flujos con un comportamiento anormal, para ser igualmente etiquetados. El proceso de anotación se realiza utilizando la herramienta Scapy [82]. El pseudo código que describe este proceso se especifica a continuación:

Algorithm 2 Pseudo código del proceso de etiquetado de mensajes

```
Crear cabeceras XML y CSV
for cada paquete en archivo PCAP do
    if tiene capa TCP then
        Obtener dirección IP y puerto fuente TCP
    end if
    if tiene capa MQTTPublish then
        Obtener msgid, clientId y topic
        if clientId en sensor de anomalía then
            Anotar paquete como anomaly
        end if
        Construir linea con frameNumber, ip_src, tcp_srcport, clientID, msgid, anomaly
    end if
    Grabar información en formato CSV y XML
end for
```

Seguidamente, se presenta un ejemplo de anotación en el archivo Valores separados por comas (CSV):

```
frame.number;ip.src;tcp.srcport;mqtt.clientid;mqtt.msgid;label

1;10.6.56.34;38378;131;1;1
```

Capítulo 5

Construcción del dataset CIDAD

Este capítulo presenta el diseño, construcción y generación del *CoAP-IoT Anomaly Detection Dataset* (CIDAD). Este dataset realiza el intercambio de mensajes haciendo uso del protocolo CoAP en la capa de aplicación. La primera sección presenta una breve descripción del protocolo CoAP, para posteriormente presentar el escenario virtual implementado, y finalmente mostrar el proceso de generación del *dataset*.

La arquitectura propuesta para la generación del *dataset* CIDAD es la mismo que se propone para el *dataset* DAD, es decir, está basada en el entorno real, consta de cuatro InRows, con cuatro sensores en cada uno de ellos, en donde cada sensor envía datos cada 5 minutos. Nuevamente una de las InRows es comprometida inyectando tráfico anómalo sobre el *payload* de los mensajes de los protocolos IoT, en cinco de los siete días. El *dataset* se genera realizando capturas para cada uno de los entornos durante siete días.

El dato de la temperatura a enviar, corresponde al realizado en el modelado matemático ,sección 3.2, ecuación 3.1, presentando igualmente los tres diferentes tipos de anomalías: *Interceptación*, *Modificación* y *Duplicación*.

5.1. Protocolo CoAP

El *Constrained Application Protocol* (CoAP) es un protocolo de transferencia web especializado, diseñado por el Internet *Internet Engineering Task Force* (IETF) para ser usado en nodos y redes restringidas. Está diseñado para interactuar fácilmente con HTTP para la integración con la web mientras cumple con los requisitos especializados, como soporte de multidifusión, sobrecarga muy baja y simplicidad para entornos restringidos,

y aplicaciones M2M, como energía inteligente y automatización de edificios [83].

El modelo de interacción de CoAP es similar al modelo cliente/servidor de HTTP. Sin embargo, las interacciones de máquina a máquina generalmente dan como resultado una implementación de CoAP que actúa tanto en roles de cliente como de servidor. CoAP consume servicios de una manera más sencilla utilizando URI como sustantivos y métodos HTTP *GET*, *POST*, *PUT* y *DELETE* como verbos. A diferencia de transferencia de estado representacional (REST), CoAP está vinculado a UDP (no TCP) de forma predeterminada, lo que lo hace más adecuado para aplicaciones IoT. Además, CoAP modifica algunas funcionalidades de HTTP para cumplir con los requisitos de IoT, como el bajo consumo de energía y operación en caso de conexiones ruidosas y con pérdidas. CoAP se puede dividir en dos subcapas: la subcapa de mensajería y la subcapa de solicitud/respuesta. La subcapa de mensajería detecta duplicaciones y proporciona comunicación confiable sobre la capa de transporte UDP usando *exponencial backoff*, ya que UDP no tiene un mecanismo integrado de recuperación de errores. Por otro lado, la subcapa de solicitud/respuesta maneja las comunicaciones REST. La razón para diseñar un sistema basado en el protocolo UDP para administrar los recursos es eliminar la sobrecarga de TCP y reducir los requisitos de ancho de banda. Además, CoAP admite unidifusión y multidifusión, a diferencia deTCP, que por su naturaleza no está orientado a la multidifusión.

CoAP define cuatro tipos de mensajes:

- *Confirmable*: un mensaje de solicitud que requiere un acuse de recibo (ACK). La respuesta se puede enviar de forma síncrona (dentro del ACK) o si necesita más tiempo computacional, se puede enviar de forma asíncrona con un mensaje separado.
- *Non-Confirmable*: un mensaje que no necesita ser reconocido.
- *Acknowledgment*: confirma la recepción de un mensaje confirmable.
- *Reset*: confirma la recepción de un mensaje que no pudo ser procesado.

CoAP utiliza un formato simple y pequeño para codificar mensajes. La primera parte, fija, de cada mensaje son cuatro bytes de encabezado. Luego un valor de token puede aparecer cuya longitud tiene un rango de entre cero y ocho bytes. El valor del token se utiliza para correlacionar solicitudes y respuestas. Las opciones y la carga útil son los siguientes campos opcionales. Un mensaje típico de CoAP puede estar entre 10 a 20 bytes, tal y como se muestra en la figura 5.1. Ejecutándose sobre UDP (protocolo no fiable), CoAP integró sus propios mecanismos para lograr fiabilidad. Dos bits en el encabezado de cada paquete indican el tipo de mensaje y el QoS requerido. Los campos del encabezado son los siguientes:

- *Version*: es la versión de CoAP.
- *Type*: es el tipo de transacción.
- Longitud token: indica el tamaño del token.
- *Code*: representa el método de solicitud (1–10) o el código de respuesta (40–255).

Figura 5.1: Formato de mensaje CoAP.

Algunas de las funciones importantes proporcionadas por CoAP incluyen [84]:

- Observación de recursos: permite suscripciones bajo demanda. Esto le posibilita monitorizar recursos de interés usando el mecanismo publicar/suscribir.
- Transporte de recursos por bloques: significa la capacidad de intercambiar datos del transceptor entre el cliente y el servidor sin necesidad de actualizar todos los datos para reducir la sobrecarga de comunicación.
- Descubrimiento de recursos: el servidor utiliza rutas de URI conocidas basadas en los campos de enlace web en formato de enlace CoRE para proporcionar descubrimiento de recursos para el cliente.
- Interactuar con HTTP: Flexibilidad de comunicación con variedad de dispositivos, debido a que la arquitectura REST común permite que CoAP interactúe fácilmente con HTTP a través de un proxy.
- Seguridad: CoAP es un protocolo seguro, ya que se basa en la seguridad de la capa de transporte de datagramas Datagram Transport Layer Security (DTLS) para garantizar la integridad y confidencialidad de los mensajes intercambiados.

Constrained Application Protocol (CoAP) implementa también un mecanismo simple de retransmisión *Stop-and-Wait* para mensajes confirmables y un campo de encabezado de 16 bits en cada paquete *Constrained Application Protocol* (CoAP) llamado ID de mensaje, que es único y se utiliza para detectar duplicados. *CoAPCHTTP Mapping* permite a los clientes de *Constrained Application Protocol* (CoAP) acceder a recursos en servidores HTTP a través de un proxy inverso que traduce los códigos de estado HTTP a los códigos de respuesta de CoAP [79].

5.2. Descripción del escenario virtual

Para el entorno CoAP, la infraestructura virtual es construida usando cinco nodos virtuales VMware ESXi 6.5 configurados con Ubuntu 20.04, usando vSwitch para nodos interconectados, aislados de la red externa. Nuevamente, el dataset se genera monitorizando la red durante siete días consecutivos en el entorno simulado, en donde el etiquetado de tráfico se realiza a nivel de paquete. Por la forma en que CoAP realiza el intercambio de mensajes, en donde cada flujo estaría compuesto de un único paquete, no es posible realizar un etiquetado por flujos.

Este entorno está constituido por el nodo que corresponde al cliente y cuatro servidores que simulan el funcionamiento de las InRows, que envían los datos de temperatura correspondientes. Según cómo funciona el protocolo CoAP, un nodo cliente puede solicitar al servidor el envío de un paquete CoAP. Cada sensor se identifica mediante una dirección URI (por ejemplo, www.inrow15.gal/1), donde el recurso indica la InRow y el número que sigue a la barra significa el sensor correspondiente: (1) TAS, (2) TAR, (3) TFEU, y (4) TFSU. El cliente CoAP envía mensajes a cada uno de los cuatro sensores de cada InRow, actuando como monitor del sistema, consultando cada InRow periódicamente.

El mensaje del cliente CoAP contiene un paquete CoAP creado con la herramienta Scapy de Python [82] y el valor del token se genera aleatoriamente. Finalmente, el paquete CoAP se envía de forma asíncrona utilizando el método de envío de la biblioteca Scapy. Cada uno de los servidores escucha los mensajes del cliente, utilizando el método sniff de la librería Scapy y realizando un proceso de filtrado. Por cada mensaje recibido se envía una respuesta que incluye el valor de temperatura de cada nodo.

Las muestras generadas por los sensores se envían cada cinco minutos al cliente a través de un mensaje CoAP que contiene su identificador de nodo como clientId (usando valores URI-Host y URI-Path). La captura del intercambio de tráfico entre los nodos se ejecuta en el cliente. La información de conexión se almacena para ser etiquetada posteriormente en el conjunto de datos a través de Scapy. La mayoría de los paquetes anormales se marcan en el servidor, excepto aquellos que pertenecen a la anomalía de interceptación.

5.3. Generación del dataset CIDAD

CoAP está diseñado para interactuar fácilmente con HTTP y se ejecuta sobre UDP y los clientes y servidores se comunican a través de datagramas sin conexión [85]. El puerto UDP es el 5683. A continuación, se describe la estructura utilizada en este escenario para

el intercambio de mensajes en cada uno de los roles que utiliza cada host:

- Cliente: envía mensajes de solicitud de CoAP a las InRows (servidores). Estos mensajes son de tipo Non-Confirmable y no requieren un mensaje ACK de confirmación (la red estará menos sobrecargada y los dispositivos consumirán menos recursos al procesar y enviar estos mensajes). La solicitud enviada por el cliente tiene la siguiente estructura:

  - *Version*: el valor de este campo será 1 (01).
  - *Type*: dado que los mensajes serán de tipo NON, el valor será 1 (01).
  - *Code*: los mensajes que se envíen desde el cliente solicitando un recurso del servidor serán del método GET. Por tanto, el valor del campo de código será 1 (0,01).
  - *Message-ID*: se calcula según el mensaje enviado. El id-mensaje aumenta de acuerdo con cada mensaje y sigue un orden cronológico.
  - *Token*: cada token es un campo único y generado aleatoriamente del mensaje. El tamaño de este token es de 4 bytes, el tamaño mínimo recomendado.
  - *Options*: las opciones utilizadas son *URI-Host* y *URI-Path*, para poder identificar correctamente el recurso en el servidor correspondiente. También agregaremos la opción *Accept* con valor 0, que indica que el formato de representación de recursos que acepta el cliente es texto plano (text/plain).

- Servidor: las InRows serán las que actuarán como servidores devolviendo respuestas al cliente. El escenario incluye cuatro: InRow *www.inrow13.gal*, *www.inrow15.gal*, *www.inrow23.gal* y *www.inrow25.gal*. Cada sensor de la InRow tiene un identificador de recurso o URI-Path designados como 1, 2, 3 y 4. Las respuestas enviadas por el servidor tienen la siguiente estructura:

  - *Version*: igual que el mensaje original (01).
  - *Type*: mismo tipo que el mensaje original (tipo NON, 01).
  - *Code*: el código de respuesta es 2.05 Contenido, que incluye la representación del recurso solicitado.
  - *Message-ID*: esto se genera aleatoriamente para cada mensaje de respuesta.
  - *Token*: lo mismo que el mensaje de solicitud.

- *Options*: las opciones *URI-Host* y *URI-Path* son las mismas que las de la solicitud. También se agrega la opción *Content-Format*, que especifica el formato de representación del objeto incluido en el campo payload del mensaje (en nuestro caso, el valor será 0, text/plain).
- *Payload*: este campo incluye la representación del objeto, que es el valor de temperatura correspondiente al sensor en este escenario específico.

El código 3 muestra cómo el cliente construye el mensaje para cada uno de los nodos de las InRows.

Algorithm 3 Pseudo código del proceso de construcción del mensaje del cliente

for $server, ip$ in $self.servers$ **do**
  for $sensor$ in $self.sensors$ **do**
    $coapPackage \leftarrow CoAP($ $ver = 1$, $type = 1$, $code = 1$,
      $msg\_id = self.create\_msgID$, $token = self.create\_token$,
      options =[("$UriHost$", $server$), "$Uripath$", sensor)
      ("$Accept$", b"$x00$")], paymark = b"$0xff$")
    $packet \leftarrow (IP(dst \leftarrow ip)UDP(sport \leftarrow 5683, dport \leftarrow 5683)coapPackage)$
  end for
end for

A continuación, el código 4 muestra la respuesta del servidor al cliente. El cliente tiene una dirección IP "192.168.0.1":

Algorithm 4 Pseudo código del proceso de construcción del mensaje del servidor

if req.haslayer(CoAP) and req.getlayer(IP).dst == "192.168.0.1" **then**
  $respCoap \leftarrow (CoAP($
    $ver = coap.ver$, $type = coap.type$, $code = 69$,
    $msg\_id = self.create\_msgID()$,
    $token = coap.token$,
    $options = [coap.options[0], coap.options[1]$,
    ("$ContentFormat$", b"$x00$")],
    $paymark = coap.paymark$)
    $Raw(load = self.calculate\_replyText(coap)))$
  $respIP = (IP(dst = ip.src)/UDP(sport = udp.dport, dport = udp.sport)/respCoap)$
end if

Para la inyección de anomalías en este entorno se usó la herramienta Ettercap utilizando una máquina con Kali Linux 2020.1 como proxy entre una de las InRows y el cliente. Tras un escaneo de la red se selecciona la InRow a ser suplantada y ahora todas

las comunicaciones que tengan lugar entre las dos máquinas pasarán por la máquina intermedia. En el caso de los paquetes de eliminación se genera un filtro que descarta los paquetes de respuesta al cliente. Por lo tanto, los paquetes de petición son realizados, pero la respuesta nunca llega a destino y dichos paquetes desde el cliente pueden ser marcados como anomalías. Los mensajes son etiquetados con la herramienta de scapy e importados en un CSV [86].

Capítulo 6

Análisis de los datasets

Para el desarrollo de esta etapa de la tesis, se aplicó una adaptación de la metodología para minería de datos *Cross Industry Standard Process for Data Mining* (CRISP-DM)[87]. La metodología CRISP-DM se conceptualiza en 6 fases:

- *Entendimiento del negocio*: Esta fase inicial se enfoca en entender los objetivos y requerimientos del proyecto, para convertir este conocimiento en una definición del problema y un diseño preliminar para alcanzar los objetivos.
- *Entendimiento de los datos*: La fase de entendimiento comienza con la recolección inicial de los datos, hasta familiarizarse con ellos y encontrar problemas en la calidad de los mismos. La idea es descubrir las primeras perspectivas de los datos o detectar interesantes subconjuntos para formular hipótesis de informaciones ocultas.
- *Preparación de los datos*: La fase de preparación cubre todas las actividades para construir el conjunto de datos finales (datos que alimentarán el modelo de aprendizaje) a partir de los datos en crudo. Este puede llevarse a cabo múltiples veces y en ningún orden especifico. Las tareas incluyen selección de tablas, registros y atributos, así como transformación y limpieza de datos para herramientas de modelado.
- *Modelado*: En esta fase, se seleccionan y aplican varias técnicas de modelado y sus parámetros se calibran a valores óptimos. Algunas técnicas tienen requisitos específicos sobre la forma de los datos. Por lo tanto, a menudo es necesario retroceder a la fase de preparación de datos.
- *Evaluación*: En la etapa previa se creó un modelo (o modelos) que parece tener una alta calidad desde la perspectiva del análisis de datos. Antes de proceder al

despliegue final del modelo, es importante evaluarlo más a fondo y revisar los pasos ejecutados para construirlo para asegurarse de que logre correctamente los objetivos. Al final de esta fase, se debe llegar a una decisión sobre el uso de los resultados.

- *Despliegue*: La creación del modelo generalmente no es el final del proyecto. Implica la aplicación de modelos dentro de los procesos. La fase de despliegue puede ser tan simple como generar un informe o tan compleja como la realización periódica y quizás automatizada de un proceso de análisis de datos.

Partiendo de "entender el negocio" como la generación de un sistema de clasificación de anomalías haciendo uso de algoritmos de aprendizaje automático y tomando los *datasets* generados como elementos a clasificar por los modelos de ML seleccionados, la aplicación óptima de dichas técnicas requiere de datos acondicionados para ser perfectamente interpretados por el clasificador. Para realizar el preprocesado de los datos, es necesario determinar las características intrínsecas de estos mediante un análisis de los *datasets*. Por consiguiente, este capítulo presenta las fase de preparación de los datos para cada uno de los *datasets* presentados. La primera sección corresponde al *dataset* DAD, para posteriormente analizar el *dataset* CIDAD.

Las anomalías de duplicación, interceptación y modificación, para ambos *datasets*, fueron inyectadas en la InRow 13, durante franjas horarias especificas en diferentes combinaciones a lo largo de la semana. La presencia de la anomalía y su tipo se presenta en la tabla 6.1 .

Tabla 6.1: Inyección de anomalías en días de la semana.

Día	**Modificación**	**Interceptación**	**Duplicación**
Lunes			
Martes		✓	
Miércoles			✓
Jueves	✓		
Viernes	✓	✓	✓
Sábado	✓	✓	✓
Domingo			

A través de tshark [88], una componente de Wireshark que permite filtrar una diversa cantidad de características sobre los *pcaps* en las distintas capas de la red, se seleccionan diferentes características iniciales para cada uno de los *datasets*.

6.1. Descripción general del dataset DAD

Para el entorno MQTT, las características iniciales seleccionadas se muestran en la tabla 6.2. A partir de estas, se crean algunas nuevas características, tales como día de la semana, hora, tipo de protocolo en capa de aplicación, retraso entre paquetes, identificador de flujo, número de paquetes por flujo y duración del flujo, entre otras, que permitan identificar comportamientos específicos con más facilidad.

Tabla 6.2: Características iniciales en DAD.

Nº	**Nombre del campo**	**Descripción**	**Tipo**
1	frame.len	Longitud de la trama	Entero sin signo (4 bytes)
2	frame.number	Número de la trama para identificación	Entero sin signo (4 bytes)
4	frame.time	Fecha y hora del llegada de la trama	DateTime
5	frame.time_epoch	Segundos desde el 01/01/1970 hasta el evento	Time offset
6	frame.protocols	Protocolos de la trama	Cadena de caracteres
7	ip.dst	Dirección IP de destino	Dirección IPv4
8	ip.hdr_len	Longitud de la cabecera IP	Entero sin signo (1 byte)
9	ip.id	Identificador IP	Entero sin signo (2 bytes)
10	ip.len	Longitud total de IP	Entero sin signo (2 bytes)
11	ip.proto	Código de protocolo en IP	Entero sin signo (1 byte)
12	ip.src	Dirección IP de fuente	Dirección IPv4
13	tcp.srcport	Puerto TCP fuente	Entero sin signo (2 bytes)
14	tcp.dstport	Puerto TCP destino	Entero sin signo (2 bytes)
15	tcp.options.timestamp.tsval	Valor del Timestamp	Entero sin signo (4 bytes)
16	tcp.flags.str	Flags TCP	Cadena de caracteres
17	tcp.flags	Flags TCP	Lógico
18	tcp.flags.ack	Flag ACK	Lógico
19	tcp.flags.fin	Flag FIn	Lógico
20	tcp.flags.res	Flag Reserved	Lógico
21	tcp.flags.reset	Flag Reset	Lógico
22	tcp.flags.syn	Flag Syn	Lógico
23	mqtt.clientid	clientId	Cadena de caracteres
24	mqtt.protoname	Nombre de protocolo MQTT	Cadena de caracteres
24	mqtt.msg	Carga útil msg MQTT	Secuencia de Bytes
26	mqtt.topic	Tópico	Cadena de caracteres

En el *dataset* DAD cada nodo cliente envía un mensaje al *broker* con el identificador correlacionado *mqtt.clientid* de la unidad de frío (InRow). Por ejemplo, la InRow 13 tiene registrado 131 para TAS, 132 para TAR, 133 para TFEU y 134 para TFSU. Todos los sensores de la InRow tienen la misma dirección IP, pero los puertos TCP que utilizan para la transmisión de mensajes son diferentes cada día, utilizando un total de 112 puertos TCP durante la semana. Al iniciar la conexión el mensaje con información de clientId, dirección IP de fuente y el puerto TCP es enviado. Para la correcta identificación del sensor fue necesario realizar una equivalencia entre el puerto empleado y su clientId. La equivalencia de puertos con relación al clientId se presenta en la tabla 6.3.

Tabla 6.3: Equivalencia de puertos TCP a clientId.

ip.src 10.6.56.41		**ip.src 10.6.56.36**		**ip.src 10.6.56.50**		**ip.src 10.6.56.34**	
tcp.srcport	**clientId**	**tcp.srcport**	**clientId**	**tcp.srcport**	**clientId**	**tcp.srcport**	**clientId**
49134		47588		44488		40818	
49146		47600		44500		40824	
49150		47604		44502		40834	
49158	131	47620	151	44516	231	40840	251
49166		47628		44518		40850	
49174		47630		44532		40856	
49182		47638		44540		40868	
49140		47590		44486		40820	
49144		47602		44494		40826	
49154		47610		44506		40838	
49160	132	47616	152	44514	232	40842	252
49168		47624		44520		40852	
49180		47632		44528		40858	
49186		47644		44538		40866	
49136		47592		44492		40816	
49142		47598		44496		40828	
49142		47606		44508		40832	
49162	133	47614	153	44510	233	40844	253
49170		47622		44522		40848	
49176		47634		44536		40860	
49184		47640		44530		40864	
49138		47642		44534		40822	
49148		47594		44490		40830	
49156		47596		44498		40836	
49164	134	47612	154	44504	234	40846	254
49172		47618		44512		40854	
49178		47626		44524		40862	
49188		47636		44526		40870	

Los sensores envían datos de temperatura al *broker* cada cinco minutos, es decir, en condiciones normales enviarían 288 muestras al día, y 2017 paquetes a la semana. DAD tiene un total de 101.583 paquetes, presentando 96,9 % de tráfico TCP y 3,4 % de tráfico UDP en la capa de transporte. El 63,3 % de los paquetes totales pertenecen a tráfico MQTT y el 1,6 % de estos están marcados como anomalías. La cantidad de bytes de origen, bytes de destino, paquetes de origen, paquetes de destino, paquetes TCP, paquetes UDP y paquetes MQTT es uniforme a lo largo de los días de la semana, presentando un promedio de 14.020 paquetes TCP y 9.267 paquetes MQTT por día. El tráfico anómalo se realiza desde la InRow 13 sobre los paquetes MQTT, con dirección IP fuente 10.6.56.41.

La información con el número de bytes enviados y recibidos, la cantidad de paquetes enviados y recibidos, la cantidad de paquetes en los protocolos UDP, TCP y MQTT, y la cantidad de paquetes anómalos y normales del *dataset* se presenta en la tabla 6.4.

Tabla 6.4: Tabla resumen del dataset DAD.

Día	Src Bytes	Dst Bytes	Src Paq	Dst Paq	TCP	UDP	MQTT	Paq normal	Paq anómalos
Lunes	699,527	348,336	9,526	4,905	13,936	495	9,248	14,431	0
Martes	696,658	347,846	9,491	4,907	13,908	490	9,184	14,398	0
Miércoles	702,714	350,844	9,574	4,950	14,032	492	9,264	14,316	208
Jueves	703,431	351,018	9,585	4,953	14,048	490	9,268	14,426	112
Viernes	704,084	351,124	9,592	4,952	14,054	490	9,292	14,160	384
Sábado	707,946	353,310	9,645	4,985	14,138	492	9,339	14,246	384
Domingo	702,744	350,682	9,571	4,947	14,026	492	9,277	14,518	0
Total	**4,917,104**	**2,453,160**	**66,984**	**34,599**	**98,142**	**3,441**	**64,872**	**100,495**	**1,088**

Una de las características que brinda mucha información en el análisis de tráfico es la cantidad de paquetes enviados desde cada dirección IP fuente. Esta información permite obtener una idea de la distribución de los paquetes en la red, la forma en la que se interconectan los nodos y vislumbra el comportamiento general del sistema.

La cantidad de paquetes enviados desde cada IP de origen a lo largo de los días se muestra en la figura 6.1. Cada dirección IP representa una InRow, es decir, los paquetes presentados son la suma de los paquetes enviados por los cuatro sensores. Por la propia estructura y las cualidades de transporte de mensajes de publicación/suscripción del protocolo MQTT, la red de sensores IoT envía todos los paquetes al *broker* y los sensores no crean conexiones entre sí. Por lo tanto, la mayor parte del envío de paquetes será realizada del *broker* a los sensores y las conexiones de los sensores con este deberán presentar una distribución homogénea en caso de normalidad.

La tabla 6.5 presenta toda la composición de protocolos en el *dataset*. Las InRows realizan diariamente una conexión de sincronización con el *broker* mediante el protocolo NTP enviando un promedio de 82 paquetes diarios y su respectiva respuesta. El nodo con dirección IP 10.6.56.34 tiene una segunda interfaz de red a través de la cual se sincroniza con el servidor NTP. Por esta razón, y como la captura del trafico es realizada en el *broker*, en la figura 6.1 el tráfico NTP proveniente del nodo 10.6.56.34 no aparece, y hay ausencia de paquetes en este nodo.

Las anomalías de interceptación, al no ser recibidas en el *broker* no pueden ser marcadas, por lo tanto no se reflejan en las estadísticas generales de anomalías. En otras palabras, los porcentajes de tráfico anómalo no tienen en cuenta las anomalías de interceptación, y es necesario generar otros mecanismos para su identificación. Un indicio de esta anomalía puede ser visto mediante la ausencia de paquetes en la dirección IP 10.6.56.41 en la figura 6.1 el día martes. Sin embargo, la detección de la anomalía de interceptación por ausencia de paquetes, puede verse difícil de reconocer ante la presencia de anomalías de duplicación, dado que la combinación de estas dos alteraciones presenta equilibrio en la distribución. Es importante también resaltar que los cierres de conexión TCP se realizan siempre al día siguiente de establecida la conexión, por lo tanto, la ausen-

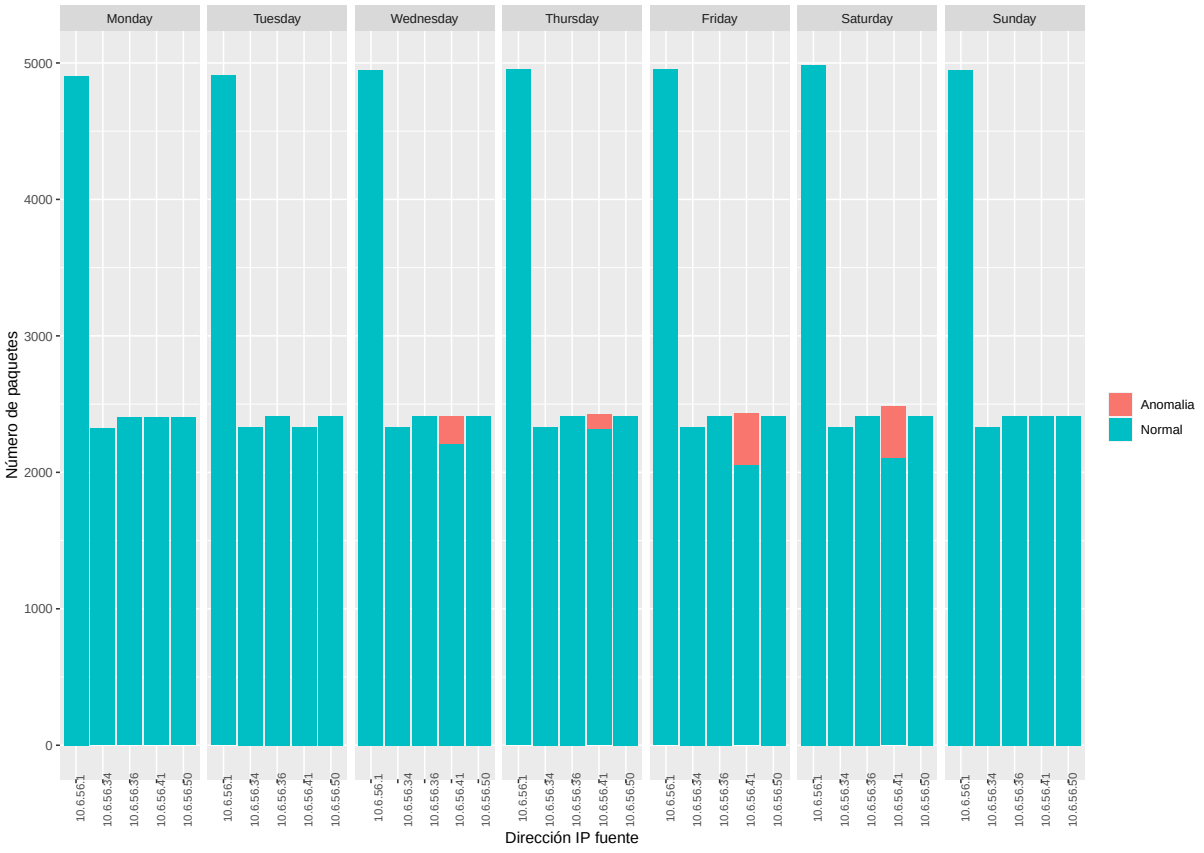

Figura 6.1: Número de paquetes por IP fuente por etiqueta por día en dataset DAD.

cia de paquetes presentados el día domingo, son los paquetes pertenecientes a los cierres de conexión TCP.

Tabla 6.5: Número de paquetes por protocolo en el dataset DAD.

Protocolo		**Lunes**	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**	**Domingo**
TCP	MQTT	9,248	9,184	9,264	9,268	9,292	9,339	9,277
		4,688	4,724	4,768	4,780	4,762	4,799	4,749
UDP	NBNS	3	0	0	0	0	0	0
	NTP	492	490	492	490	490	492	492

Como se puede observar en el análisis general, y debido a las condiciones de configuración del entorno, la mayoría de las características presentan homogeneidad en su comportamiento. Esta situación nos lleva a buscar características de IoT que permitan establecer rasgos específicos en el comportamiento de la red. Todas las anomalías se presentan en los paquetes MQTT sobre TCP.

6.1.1. Descripción TCP/MQTT

Retomando, sabemos que el *dataset* contiene 98.142 paquetes TCP, de los cuales 64.872 son MQTT y 1088 corresponden a anomalías marcadas. En la carga útil de los mensajes se tienen temperaturas promedio de 18,43ºC, donde la mínima es de 8,67ºC y la máxima de 29,40ºC.

Cada uno de los sensores establece una conexión TCP con el *broker* diariamente, lo que corresponde al flujo TCP. La conexión siempre se cierra al día siguiente, por lo que se presentan algunos paquetes correspondientes a un flujo de un día al día siguiente. El datagrama TCP contiene banderas o *flags* TCP que controlan la transmisión de datos a través de una conexión. En el caso del *dataset* DAD, los *flags* empleados en la conexión son *acknowledgement* (ACK), *synchronization* (SYN), *finished* (FIN) y *reset* (RST). El *dataset* tiene un total de 224 flujos de conexión TCP, de los cuales 208 representan flujos normales y 16 flujos anómalos. Para comprender el estado de la conexión fue conveniente realizar el análisis los *flags*, para determinar cómo el *broker* establecía y cerraba las conexiones. Las conexiones realizadas se muestran en la tabla 6.6. El comportamiento de conexiones particulares se debe a que a nivel de aplicación envía el mensaje de desconexión pero luego no espera el *flag* ACK.

Tabla 6.6: Análisis de flags TCP.

		Broker	
		S	SF
Cliente	S	16	0
	SF	0	4
	SFR	7	0
	SFRR	0	85

La configuración por defecto del envío de mensajes desde los sensores es de 5 minutos, lo que permite establecer un tiempo de inactividad de 30 segundos (los valores típicos de tiempo de espera oscilan entre 15 segundos y 5 minutos [89]). Para la generación de flujos MQTT, se puede decir que transcurridos más de 300 segundos entre el envío de un paquete y el siguiente, cada paquete corresponde a un flujo diferente. Como consecuencia, una conexión TCP se puede dividir en diferentes flujos MQTT. Los flujos se establecen como unidireccionales. Así mismo, si un paquete pertenece a un flujo donde al menos un paquete está etiquetado como anomalía, esta alteración afecta a todo el flujo y en consecuencia todos los paquetes del flujo son etiquetados como anormales. El etiquetado de los flujos se realizó a posteriori.

El número de flujos TCP/MQTT por día se muestran en la tabla 6.7. Todas las

anomalías se presentan sobre MQTT. DAD contiene 64.376 flujos MQTT, de los cuales 544 corresponden a flujos anómalos. El número de flujos a lo largo de los días es homogéneo, presentando un número menor de flujos el domingo debido a los cierres de conexión que se realizan al día siguiente. Se tiene una media de 1,73 paquetes por flujo.

Tabla 6.7: Número de flujos TCP/MQTT por anomalía.

Día	**Anómalos**	**Normal**	**Total**
Lunes	0	9,216	9,216
Martes	0	9,168	9,168
Miércoles	104	9,152	9,256
Jueves	56	9,200	9,256
Viernes	192	9,080	9,272
Sábado	192	9,136	9,328
Domingo	0	8,880	8,880
Total	**544**	**63,832**	**64,376**

Otra característica relevante a analizar es la duración de los flujos MQTT. La figura 6.2 (a) presenta el comportamiento del *broker* con la dirección IP 10.6.51.1. La figura 6.2 (b) corresponde a la InRow 23 con la dirección IP 10.6.56.50 como exponente de un nodo con comportamiento normal y la figura 6.2 (c) corresponde a la InRow 13, nodo anómalo, con la dirección IP 10.6.51.41.

Las anomalías de interceptación consisten en evitar la recepción de algunos paquetes. Su falta de etiquetado en el *dataset* hace de su reconocimiento una tarea difícil. Un mecanismo eficaz para la identificación de este tipo de anomalía es mediante el análisis del *timelag*, o retraso entre paquetes. En la configuración inicial del sistema, se establece un desfase temporal en la recepción entre un mensaje y otro de 5 minutos. Una anomalía de interceptación puede ser identificada cuando el tiempo de recepción entre paquetes es superior al estipulado. La alteración del *timelag* junto con la disminución de paquetes enviados al día es una muestra clara de presencia de anomalía de interceptación. La aparición de una anomalía de duplicación, junto con la de interceptación, puede dificultar la detección de la falta de paquetes. Sin embargo, dado que las anomalías de duplicación están marcadas en el conjunto de datos, la presencia de anomalías de interceptación debe presentarse entre flujos normales. Para ejemplarizar este comportamiento la figura 6.3 muestra una representación logarítmica lineal de la duración del retraso entre flujos, donde se aprecia el comportamiento del *broker* con la dirección IP 10.6.51.1, la InRow con la dirección IP 10.6.56.50, que corresponde a un nodo que no ha sufrido alteraciones, y la InRow 13 con la dirección IP 10.6.51.41, nodo que ha sido interceptado.

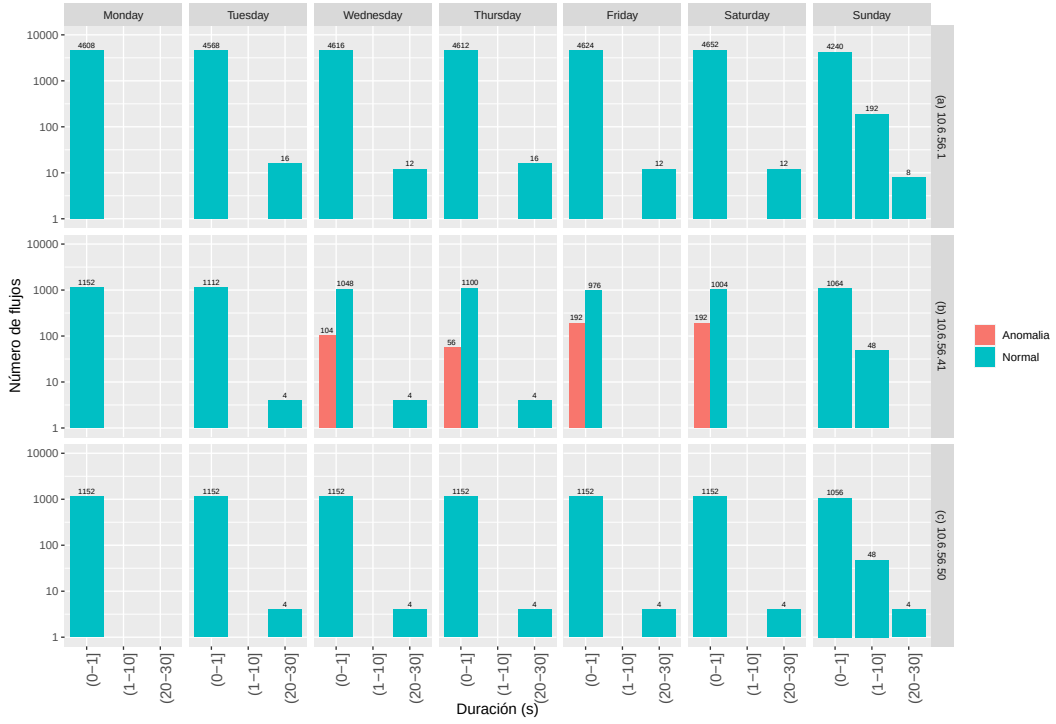

Figura 6.2: Duración de flujo en escala log-lineal: (**a**) Broker, (**b**) Nodo anómalo, (**c**) Nodo normal.

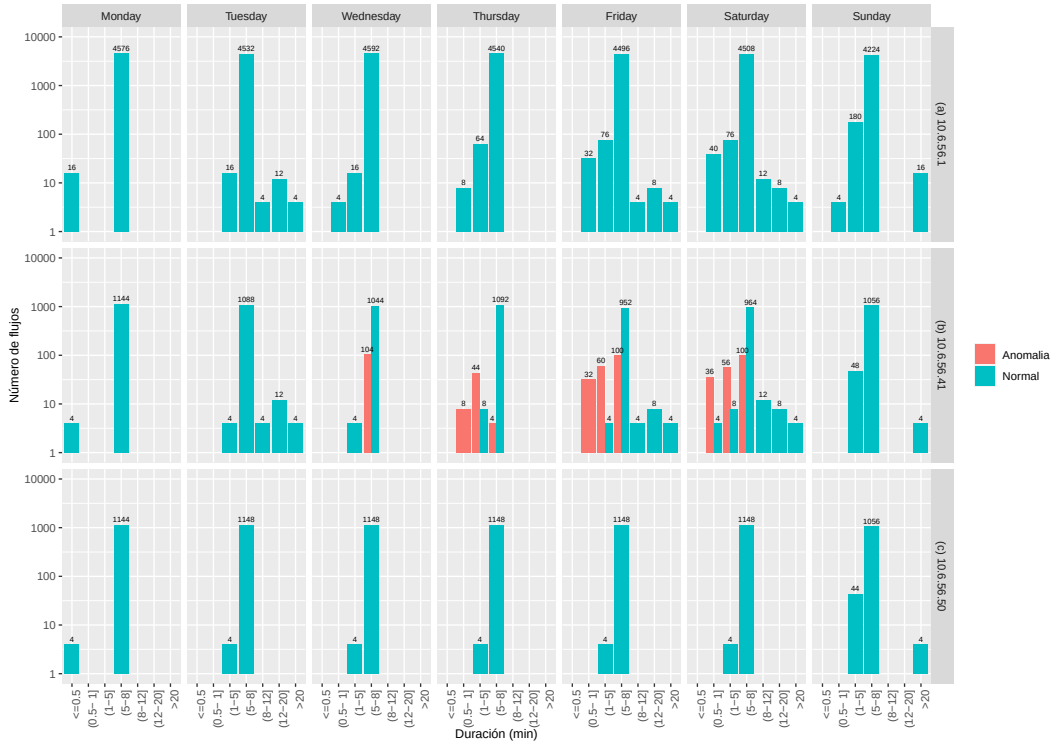

Figura 6.3: Duración del retardo entre flujos en escala log-lineal: (**a**) Broker, (**b**) Nodo anómalo, (**c**) Nodo normal.

6.2. Descripción general del dataset CIDAD

Para el entorno CoAP existe un nodo cliente, que será el encargado de realizar las peticiones a los sensores. Desde este nodo se obtienen las capturas. Para este *dataset* se filtran las características iniciales a considerar, algunas propias para tramas UDP y protocolo CoAP, tal y como se presenta en la tabla 6.8.

Tabla 6.8: Características iniciales del dataset CIDAD.

Nº	**Nombre del campo**	**Descripción**	**Tipo**
1	frame.len	Longitud de la trama	Entero sin signo (4 bytes)
2	frame.number	Número de la trama para identificación	Entero sin signo (4 bytes)
3	frame.protocols	Protocolos de la trama	Cadena de caracteres
4	frame.time	Fecha y hora del llegada de la trama	
5	frame.time_epoch	Segundos desde el 01/01/1970 hasta el evento	Time offset
6	ip.dst	Dirección IP de destino	Dirección IPv4
7	ip.hdr_len	Longitud de la cabecera IP	Entero sin signo (1 byte)
8	ip.id	Identificador IP	Entero sin signo (2 bytes)
9	ip.len	Longitud total de IP	Entero sin signo (2 bytes)
10	ip.proto	Código de protocolo en IP	Entero sin signo (1 byte)
11	ip.src	Dirección IP de fuente	Dirección IPv4
12	ipv6.dst	Dirección IPv6 de destino	Dirección IPv6
13	ipv6.hopopts	Opciones Hop-By-Hop	Etiqueta
14	ipv6.src	Dirección IPv6 de fuente	Dirección IPv6
15	ipv6.flow	Etiqueta de flujo	Entero sin signo (4 bytes)
16	udp.length	Longitud de la trama UDP	Entero sin signo (2 bytes)
17	udp.dstport	Puerto UDP destino	Entero sin signo (2 bytes)
18	udp.srcport	Puerto UDP fuente	Entero sin signo (2 bytes)
19	udp.time_delta	Tiempo desde la trama anterior	Time offset
20	udp.time_relative	Tiempo desde la primera trama	Time offset
21	udp.checksum	Suma de control UDP	Entero sin signo (2 bytes)
22	coap.code	Código CoAP	Entero sin signo (1 byte)
23	coap.type	Tipo CoAP	Entero sin signo (1 byte)
24	coap.mid	Identificador de mensaje CoAP	Entero sin signo (2 bytes)
25	coap.token	token CoAP	Secuencia de Byte
26	coap.opt.uri_host	Uri-Host	Cadena de caracteres
27	coap.opt.uri_path	Uri-Path	Cadena de caracteres
28	coap.opt.accept	Aceptar	Cadena de caracteres
29	coap.opt.ctype	Content-type	Cadena de caracteres
30	coap.payload	Carga útil	Cadena de caracteres

Al igual que para DAD, se crean algunas nuevas características provenientes de características existentes tales como día de la semana, hora, sensor y tipo de protocolo en capa de aplicación, entre otras. Igualmente, en condiciones normales, debe realizarse el envío de 288 paquetes diarios con los valores de temperatura, pero en CoAP se envían a través de línea de texto de datos (*CoAP data text line*). El tráfico anómalo se realiza desde la InRow 13 con dirección IP fuente 192.168.0.2 en los días específicos de la semana

presentados en la tabla 6.1. Para el caso de CIDAD, la identificación del nodo que envía el dato de temperatura se obtiene mediante los campos *coap.opt.uri_host*, que determina la InRow, y el campo *coap.opt.uri_path*, que determina el numero de sensor al que pertenece, siendo (1) TAS, (2) TAR, (3) TFEU, y (4) TFSU.

El CIDAD tiene un total de 88.238 paquetes. La información de las estadísticas generales del *dataset* puede observarse en la tabla 6.9. De esta tabla se infiere que el 73 % de todo el tráfico corresponde a paquetes UDP, el número de paquetes enviados en IPV6 representa el 0.95 % del total de paquetes, y el 26 % el tráfico ARP. Todos los paquetes IPV6 corresponden a paquetes ICMPV6, mientras que los paquetes IPv4 corresponden a datagramas UDP e ICMP. Todas las anomalías se presentan en el protocolo CoAP/UDP. El 71,62 % de los paquetes UDP son normales, mientras que solo el 1,2 % de todo el tráfico es etiquetado como anomalía.

Tabla 6.9: Tabla resumen del dataset CIDAD.

Día	Src Bytes	Dst Bytes	Paq src	Paq dst	ARP	IP	IPv6	UDP	Paq normal	Paq anómalos
Lunes	490,344	331,776	7,040	4,608	2,312	9,216	120	9,216	11,648	0
Martes	529,942	331,818	7,737	4,609	3,105	9,123	118	9,120	12,250	96
Miércoles	537,736	331,818	7,836	4,609	3,105	9,219	121	9,216	12,349	96
Jueves	516,302	330,666	7,464	4,593	2,702	9,235	120	9,232	11,961	96
Viernes	633,196	331,902	9,442	4,611	4,707	9,225	121	9,216	13,669	384
Sábado	633,056	331,902	9,440	4,611	4,707	9,225	119	9,216	13,667	384
Domingo	489,796	331,776	7,030	4,608	2,304	9,216	118	9,216	11,638	0
Total	**3,830,372**	**2,321,658**	**55,989**	**32,249**	**22,942**	**64,459**	**837**	**64,432**	**87,182**	**1,056**

CIDAD presenta todas las anomalías sobre los paquetes de mensajes de temperatura. Debido al intercambio de mensajes solicitud/respuesta realizada por CoAP, 32,249 paquetes corresponden a peticiones realizadas por el cliente. Durante la anomalía de interceptación, el cliente realiza la petición de la temperatura, pero los paquetes de línea de texto de datos de los sensores de la InRow 13 son interceptados y no se envían al cliente. A diferencia de DAD, la anomalía de interceptación sí esta marcada y en el *dataset*, y son los paquetes sin respuesta los que se encuentran marcados como anomalías. En la anomalía de duplicación se envían paquetes adicionales, mientras que en la anomalía de modificación todas los valores de temperatura se ajustan a 11,56 ^{o}C, independientemente del sensor al que pertenecen. La tabla 6.10 presenta toda la composición de protocolos en el *dataset*.

Tabla 6.10: Número de paquetes por protocolo en el dataset CIDAD.

Protocolo			Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
IP	UDP	CoAP	4608	4608	4608	4592	4608	4608	4608
		COAP_DTxL	4608	4512	4608	4512	4608	4608	4608
	ICMP		0	3	3	3	9	9	0
ICMPv6			120	118	121	120	121	119	118
ARP			2312	3105	3105	2702	4707	4707	2304

Como consecuencia de la sencillez de la estructura que maneja el protocolo CoAP, la red presenta un intercambio de paquetes bajo, uniforme y homogéneo. La primera evidencia de paquetes eliminados o añadidos en este tipo de entorno liviano podría ser la falta de uniformidad en el tráfico. La figura 6.4 presenta la cantidad de paquetes diarios por dirección IP de fuente. Los paquetes marcados los días martes, viernes y sábado en el cliente con dirección IP 192.168.0.1 corresponden a los paquetes de petición que no recibieron respuesta y, por lo tanto, pertenecen a la anomalía de interceptación. Las anomalías presentadas sobre el nodo de la InRow con IP 192.168.0.2 corresponden a las anomalías de duplicación y modificación.

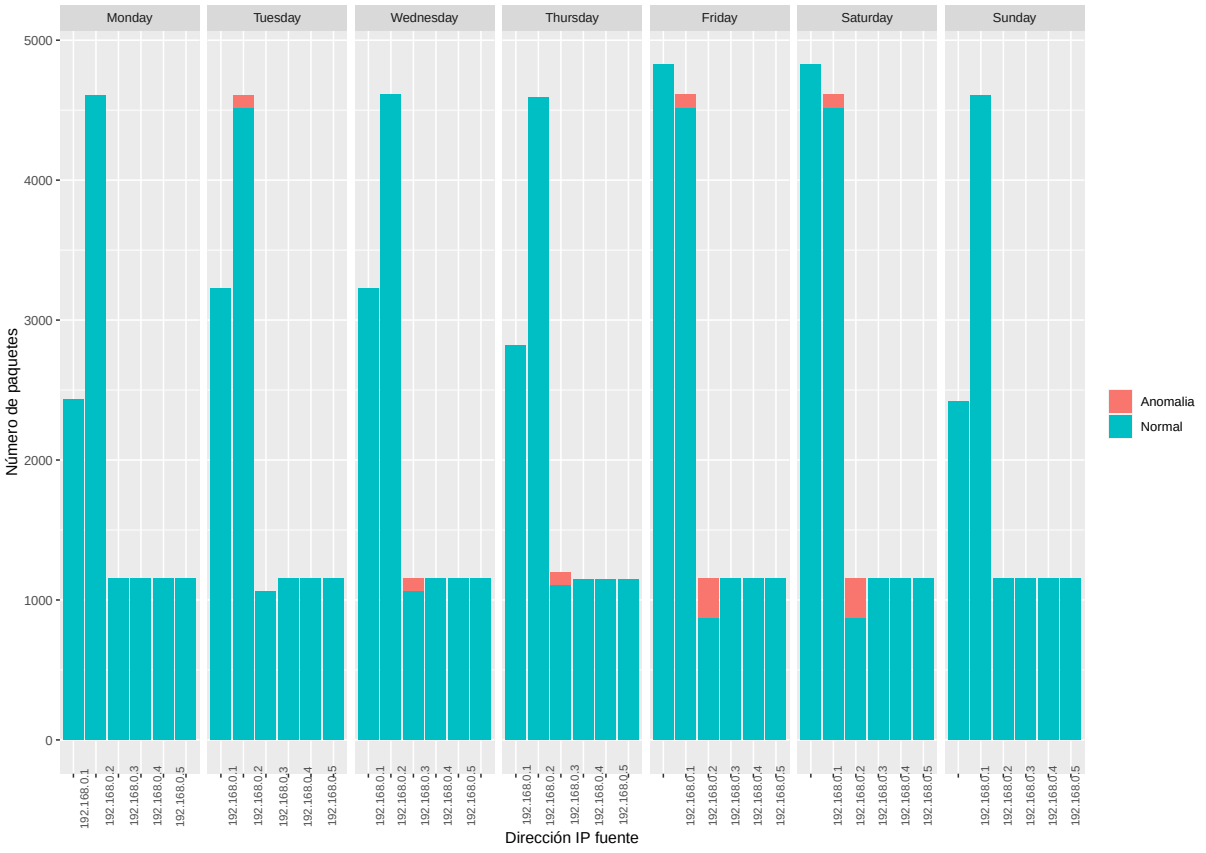

Figura 6.4: Número de paquetes por IP fuente por etiqueta por día en dataset CIDAD.

Al igual que para MQTT, en casos normales debe haber homogeneidad en la cantidad de paquetes enviados desde cada dirección IP fuente, así como en la cantidad de bytes enviados. Las alteraciones en la uniformidad en los paquetes enviados por los sensores representa un comportamiento anormal de la red. En la InRow con dirección IP 192.168.0.2 puede distinguirse, sin estar marcados, la anomalía de interceptación por falta de paquetes el día martes.

6.2.1. Descripción UDP/CoAP

CIDAD presenta el protocolo UDP en la capa de transporte y CoAP en la capa de aplicación. CoAP emplea un modelo solicitud/respuesta utilizando un formato de encabezado simple basado en binario. En la implementación realizada, CoAP utiliza dos tipos de mensajes marcados en el *dataset*, en donde los CoAP representan las solicitudes realizadas por el cliente y la *CoAP data text line* son las respuestas enviadas por cada uno de los sensores. Por lo tanto, los mensajes con los datos de temperatura de los sensores se alojarán en la carga útil de los mensajes de la línea de texto de datos CoAP.

La tabla 6.11 muestra la cantidad de paquetes en cada tipo de mensaje CoAP por anomalías. Todos los paquetes de anomalías están sobre mensajes UDP. Los paquetes etiquetados como anomalías en el tipo CoAP los martes, viernes y sábados corresponden a paquetes enviados por el cliente que no han recibido respuesta y corresponden a una anomalía de interceptación. Esta anomalía también se refleja en la ausencia de la línea de texto de datos CoAP. Las anomalías de modificación y duplicación están etiquetadas sobre la línea de texto de datos de CoAP. En el caso del miércoles (anomalía de modificación) el número de paquetes permanece invariable, mientras que el jueves (anomalía de duplicación) se produce un aumento considerable del número de paquetes. En los casos en que exista una mezcla de anomalías (viernes y sábado), el número de paquetes y su distribución no es un indicador preciso de irregularidad.

Tabla 6.11: Número de paquetes en protocolo UDP por anomalía.

	CoAP		**CoAP data text line**		**Total**
Día	**Normal**	**Anormal**	**Normal**	**Anormal**	
Lunes	4,608	0	4,608	0	9,216
Martes	4,512	96	4,512	0	9,120
Miércoles	4,608	0	4,512	96	9,216
Jueves	4,592	0	4,544	96	9,232
Viernes	4,512	96	4,320	288	9,216
Sábado	4512	96	4,320	288	9,216
Domingo	4,608	0	4,608	0	9,216
Total	**31,952**	**288**	**31,424**	**768**	**64,432**

En CIDAD al hacer uso de un protocolo no orientado a la conexión no se realiza ningún tipo de conexión previa. En consecuencia los flujos UDP corresponden al envío de un único paquete.

Capítulo 7

Preparación de los datos

Uno de los aportes de esta tesis consiste en la selección, combinación y aplicación adecuada de técnicas y algoritmos para el manejo de los datos y características. La fase de preparación de los datos implica acondicionar los datos de manera óptima para que sean perfectamente interpretados por los distintos modelos de clasificación de aprendizaje automático.

7.1. Definición de las técnicas

El rendimiento del algoritmo puede verse afectado negativamente cuando se utilizan datos sin procesar. Algunas características pueden determinar el comportamiento general de la muestra, mientras que otras simplemente no proporcionan información adicional. Esta sección presenta las técnicas adoptadas para ambos *datasets*, para limpiar, eliminar el ruido y reducir su dimensionalidad.

7.1.1. Manejo de variables horarias

Algunos datos son cíclicos por defecto. Este es el caso de las horas, los minutos y los segundos. Si tomamos la hora como una variable lineal, el clasificador no entiende que la hora 23 precede a la hora 0. Este problema puede ser resuelto si se le entregan al clasificador datos cíclicos en vez de lineales. Esta conversión es llevada a cabo reemplazando el atributo existente por dos nuevas características utilizando su representación sinusoidal. Para un

reloj de 24 horas, esto se logra con transformadas de seno y coseno, respectivamente, como en la Ecuación 7.1.

$$\begin{aligned} x &= sin(2 * \pi * hour/24) \\ y &= cos(2 * \pi * hour/24) \end{aligned} \tag{7.1}$$

El hecho de que la derivada del seno o del coseno cambia en función del tiempo, donde la posición (x,y) varía suavemente a medida que viaja alrededor del círculo unitario, hace necesario contar con ambas variables. La distancia entre dos puntos corresponde a la diferencia de tiempo en un ciclo de 24 horas.

7.1.2. Algoritmo de discretización

La mejor manera de representar los datos depende no solo de la semántica de los datos, sino también del tipo de modelo que estamos usando. Los modelos lineales y los modelos basados en árboles, las dos grandes familias aquí empleadas, tienen propiedades muy diferentes en lo que respecta a cómo funcionan con diferentes representaciones de características.

Los modelos lineales solo pueden modelar relaciones rectilíneas, que son lineas en el caso de poseer una sola característica. Una forma de hacer que los modelos lineales sean más potentes en datos continuos es utilizar la discretización (también conocida como *binning)*, que consiste en dividir una característica en varias de ellas. En este proceso una única característica continua de entrada del *dataset* es transformada en características categóricas que codifican en qué categoría nominal o bin se encuentra un punto de datos [90]. En este caso, las características categóricas del conjunto de datos deben convertirse en una representación numérica. Este proceso se realiza mediante la codificación binaria habitual, donde cada variable categórica que tiene posibles valores m se reemplaza con variables ficticias $m - 1$. Aquí una variable ficticia tiene valor uno para una categoría específica y cero para todas las categorías restantes [91].

7.1.3. Manejo de datos desbalanceados

En el entorno de intrusiones de red, es común encontrar que una de las clases constituye una minoría muy pequeña de los datos, donde esa minoría representa los casos importantes a detectar. Cuando un algoritmo aprende datos extremadamente desequilibrados, existe

una probabilidad significativa de que una muestra seleccionada contenga pocos o incluso ninguno de la clase minoritaria, lo que da como resultado un algoritmo con un rendimiento deficiente para predecir la clase minoritaria [92], haciendo fácil realizar clasificaciones incorrectas en algoritmos de aprendizaje automático.

La validacion cruzada (cv) es un método estadístico para evaluar el rendimiento de generalización, más estable y completo que usar únicamente la división en un conjunto de entrenamiento y otro de prueba.

En la validación cruzada, los datos son divididos repetidamente y se entrenan múltiples modelos. La validación cruzada de k-iteraciones estratificadas es una variación de cv k-iteraciones en donde las iteraciones se realizan conservando el porcentaje de muestras para cada clase, manteniendo la relación de equilibrio de datos entre clases. Por lo general, es una buena idea utilizar la validación cruzada estratificada en lugar de la validación cruzada sin estratificar para evaluar un clasificador, ya que da como resultado estimaciones más confiables del rendimiento de generalización [90].

Otro aproximación que permite minimizar el problema del desequilibrio de datos es el uso de técnicas de muestreo. *Synthetic Minority Over-sampling Technique* (SMOTE) es una técnica que consiste en realizar un sub-muestreo aleatorio a la clase mayoritaria, además de sobre-muestrear con reemplazo a la clase minoritaria, creando ejemplos de clases minoritarias sintéticas para aumentar la clase minoritaria [92] [93].

7.1.4. Reducción de la dimensionalidad

Hasta ahora el acondicionamiento de los datos ha incluido técnicas que adicionan características. No obstante, menos características pueden hacer que los algoritmos de aprendizaje automático se ejecuten de manera más eficiente (menos complejidad de espacio o tiempo) y sean más efectivos. Algunos modelos pueden ser engañados por características de entrada irrelevantes, lo que resulta en un detrimento del rendimiento predictivo.

La función Eliminación recursiva de caracteristicas (RFE) por sus siglas en inglés (*Recursive Feature Elimination*), implementa la selección hacia atrás de predictores en función de la clasificación de importancia de los predictores. Los predictores se clasifican y los menos importantes se eliminan secuencialmente antes del modelado. El objetivo es encontrar un subconjunto de predictores que puedan usarse para producir un modelo preciso [94]. Después de que el algoritmo selecciona los predictores más importantes, realiza una estimación de la precisión obtenida con diferentes números de predictores utilizados. El modelo seleccionado como estimador RFE fue un clasificador de árbol de decisión con validación cruzada estratificada de 5 iteraciones.

7.2. Implementación de las técnicas

La tarea de elevar la calidad de los datos, instancias y atributos al nivel requerido, considerando transformaciones de los datos y construcciones a partir de uno o más atributos existentes, genera un gran impacto en los resultados de predicción. Tras obtener una visión clara del *dataset*, es necesario aplicar las técnicas de preparación de datos anteriormente presentadas para cada uno de los entornos diseñados. Este capítulo parte de las características extraídas del análisis en cada uno de los entornos que se exponen en el capítulo 6.

7.2.1. Entorno MQTT

Conociendo la naturaleza y funcionamiento del clasificador, 14 atributos fueron considerados inicialmente relevantes y seleccionados para su procesamiento.

Tabla 7.1: Características de pre-procesado para DAD.

Características			
ip.src.	tcp.srcport	frame.len	frame.time
ip.dst	tcp.dstport	tcp.flags	tcp.flags.ack
tcp.flags.fin	tcp.flags.res	tcp.flags.reset	tcp.flags.syn
frame.protocols	label		

Revisando el entorno, sabemos que las anomalías pueden estar distribuidas a lo largo de los días, que pueden darse en los mensajes de temperatura de los sensores y cada cada sensor se puede identificar según la dirección IP y el puerto TCP de fuente. Se crean entonces nuevos atributos para una mejor estimación del modelo:

- *Weekday*: el día de la semana extraído de frame.time.
- *hora_sen*: corresponde al seno del valor hora.
- *hora_cos*: corresponde al coseno del valor hora.
- *Protocol_Type*: determina el tipo de protocolo que usa. Se estima a partir del frame.protocols.
- *clientId*: es el atributo que permite identificar a qué sensor del InRow pertenece, en base a los atributos dirección IP fuente (ip.src) y puerto TCP fuente (tcp.srcport), los cuales se muestran en la tabla 6.3.

Una vez identificado el cliente, la discretización de los puertos implicará la construcción de 112 atributos con información redundante implícita en el clientId. Tras esto, se descartan las características de puerto TCP fuente (tcp.srcport) y destino (tcp.dstport).

Para generar las etiquetas, a la clase normal se le asigna a un valor de 0 y la clase de anomalía se asigna a un valor numérico de 1. El atributo de $frame\_time$ se convierte en una variable de formato fecha para poder extraer otra tipo de información relevante de ella. La característica d_flow representa la duración del flujo y $frame.len\_mean$ corresponde a la media aritmética de la longitud de la trama (frame length) de todos los paquetes que pertenecen al flujo.

Mediante el proceso de discretización, se crean 48 variables booleanas, las cuales deben ser filtradas para evitar sobrecarga en el clasificador. La eliminación de atributos correlacionados se hace en la implementación del algoritmo de selección de características. Al aplicar RFE, como se puede observar en la figura 7.1, en donde cada color corresponde a una iteración, la mejor precisión se logra utilizando 12 características, las cuales se muestran en la tabla 7.2 y serán las utilizadas para entrenar el clasificador. Después de asignar atributos simbólicos a valores numéricos, si hay una variación significativa, se requiere escalar los valores de las características, que se logra a través de la normalización media.

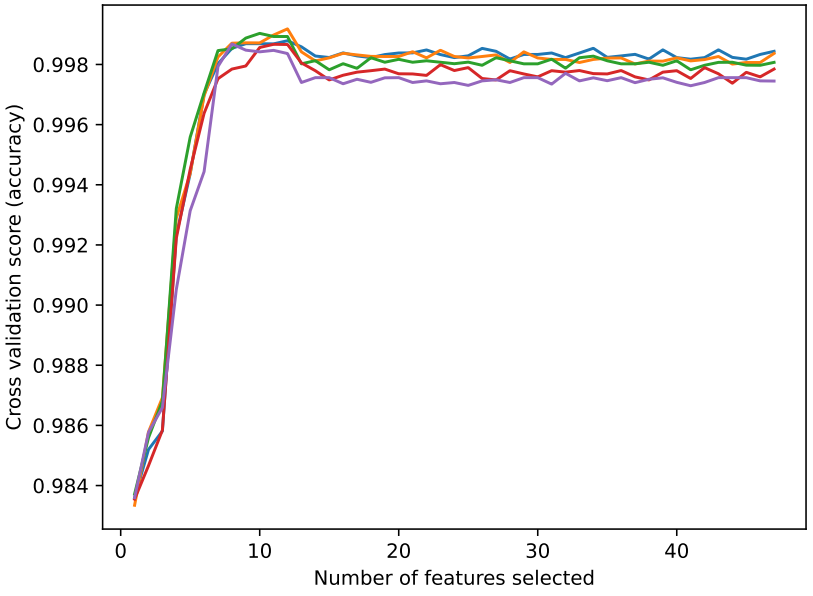

Figura 7.1: Estimación de características por RFE para DAD.

Tabla 7.2: Características seleccionadas en DAD.

Características			
hora_cos	*hora_sin*	*d_flujo*	*ip.src*_10,6,56,41
*clientId*_131	*clientId*_132	*clientId*_133	*clientId*_134
weekday_Friday	*weekday_Saturday*	*weekday_Thursday*	*weekday_Wednesday*

En el dataset DAD, encontramos 1.088 paquetes anómalos y 96.894 normales. Eso significa una proporción de 1:89, lo que muestra una situación de datos evidentemente desbalanceada. El uso de una única técnica no garantiza tomar muestras de ambas clases. Para garantizar muestras de ambas clases, en la experimentación se aplicó SMOTE dentro de un bucle anidado de validación cruzada k-iteraciones estratificado con k=5. De forma predeterminada, SMOTE utiliza cinco vecinos. Inicialmente, la clase minoritaria está sobremuestreada en una proporción de aproximadamente 0.4, y la clase mayoritaria se submuestrea a una proporción de 0.5, por lo que cada iteración de la división de cv tendrá la misma distribución de clase configurada en SMOTE.

7.2.2. Entorno CoAP

Tras un análisis de los datos, en el dataset CIDAD, como atributos para preprocesar se consideraron las características presentadas en la tabla 7.3

Tabla 7.3: Características de pre-procesado para CIDAD.

Características			
frame.len	frame.number	frame.protocols	frame.time_epoch
ip.dst	ip.id	ip.len	ip.src
ipv6.dst	ipv6.src	ipv6.flow	udp.length
udp.time_delta	udp.time_relative	udp.checksum	coap.payload
coap.opt.uri_host	coap.opt.uri_path	coap.code	coap_host
coap_sensor	label		

Se generan nuevas variables a partir de las existentes:

- *weekday*: día de la semana extraído del frame.time_epoch.
- *hora_sen*: corresponde al seno del valor hora.
- *hora_cos*: corresponde al coseno del valor hora.
- *protocolo_IoT*: determina el tipo de protocolo que usa. Se estima a partir del frame.protocols.

- *Temperatura*: valor numérico de la temperatura correspondiente al coap.payload.

- *coap_sensor*: se identifica el sensor al que pertenece mediante los campos coap.opt.uri_host y coap.opt.uri_path.

Se eliminan todas aquellas variables correlacionadas. Las variables categóricas se transforman en numéricas utilizando la técnica de discretización, convirtiendo una columna de clases en una matriz de datos. Para variaciones significativas en la transformaciones se escalan los valores de las características a través de la normalización media.

Por la forma como está construido el dataset, en este caso, tras la discretización, se generan un total de 78 atributos. La dimensionalidad del conjunto de datos de entrada es alta y por consiguiente lo será la complejidad de cada algoritmo de aprendizaje automático relacionado. Para la estimación del modelo RFE, se escogió un clasificador de árbol de decisión con cv estratificada de 5 iteraciones. La figura 7.1 presenta los cálculos de importancia obtenidos de la estimación por RFE.

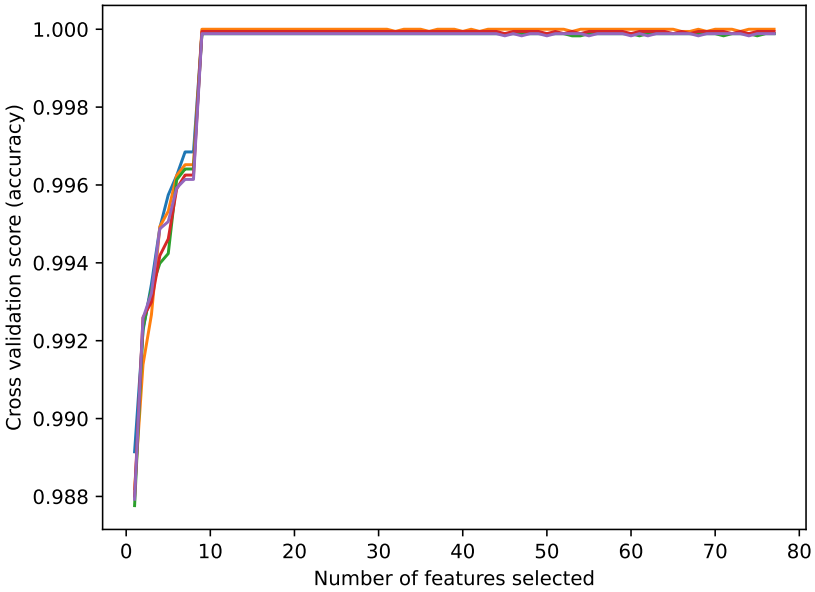

Figura 7.2: Estimación de características por RFE para CIDAD.

La mejor precisión se logra adoptando 9 características, las cuales se presentan en la tabla 7.4. Estas características son las seleccionadas para entrenar el clasificador.

Tabla 7.4: Características seleccionadas en CIDAD.

Características			
Temperatura	hora_cos	hora_sin	ip.dst_192.168.0.2
ip.src_192.168.0.2	weekday_Friday	weekday_Saturday	weekday,_Thursday
weekday_Tuesday			

En CIDAD los paquetes anormales representan 1,2 % del total de paquetes. La configuración de los algoritmos para manejo de datos desbalanceados se ha realizado de la misma manera que la utilizada con el *dataset* DAD, es decir, se ha empleado SMOTE dentro de una validación cruzada de k-iteraciones estratificadas anidada con un bucle externo con un k = 5 y un bucle interno con también k = 5. Por defecto, el SMOTE usa cinco vecinos. La clase minoritaria está sobremuestreada en una proporción de 0.4 y la clase mayoritaria submuestreada a una proporción de 0.5.

Capítulo 8

Algoritmos de aprendizaje automático

Luego de realizar el acondicionamiento de los datos y una vez seleccionadas las características, es necesario realizar el ajuste de hiperparámetros según el modelo, con el fin de mejorar la precisión del clasificador. Se seleccionan cinco modelos superficiales de aprendizaje máquina (Shallow Machine Learning) empleados para la detección de anomalías de trafico, dos de ellos parte de la familia de arboles de decisión, Random Forest y Adaboost, y tres de tipo lineal, que son Regresión Logística, Naive Bayes y Máquinas de vectores de soporte, para clasificación.

8.1. Definición de modelos e hiperparámetros

El enfoque supervisado consiste en entrenar un algoritmo usando un conjunto de datos para el cual sabemos el resultado, y luego aplicar este algoritmo para hacer una predicción a entradas de las cuales no sabemos su resultado [95]. Es extraer conocimiento de los datos enfocándose en el uso de algoritmos que imiten la forma en la que los que los humanos aprenden, mejorando su precisión gradualmente. De los muchos métodos de aprendizaje automático, algunos son menos flexibles o más restrictivos, siendo los modelos superficiales más simples en el sentido de que pueden producir, relativamente, solo un pequeño rango de formas de estimar.

En general, para construir un modelo de clasificación de alta precisión, es muy importante elegir un potente algoritmo de aprendizaje automático y ajustar sus parámetros. Los parámetros de ajuste, también llamados hiperparámetros, seleccionados de manera inadecuada conducen a resultados de clasificación deficientes. La optimización de hiperparámetros puede consumir mucho tiempo si se realiza manualmente, especialmente cuando el algoritmo de aprendizaje tiene muchos parámetros.

El proceso de optimización o ajuste de hiperparámetros es realizado antes de la fase de entrenamiento y consiste en probar varios modelos con diferentes combinaciones de valores y luego comparar el rendimiento del modelo para elegir el mejor según una métrica predefinida en un conjunto de validación cruzada. El algoritmo de búsqueda de cuadrícula (*grid-search*) es un algoritmo de optimización que basa su funcionamiento en una búsqueda exhaustiva de un subconjunto definido del espacio de hiperparámetros, mediante la identificación de una correcta combinación de parámetros. Esta optimización de parámetros se realiza empleando una técnica cv, lo que permite al clasificador predecir datos desconocidos con una alta precisión y prevenir el problema de sobreajuste [96][97].

8.1.1. Regresión Logística

El primer modelo de aprendizaje automático considerado es la Regresión Logística. Este es un método estadístico que permite analizar y predecir un resultado binario, siendo un caso específico de un conjunto de modelos lineales generalizados. El objetivo de la Regresión Logística es crear el mejor modelo de ajuste para establecer una relación de dependencia entre la variable de clase y las características [98]. Este modelo realmente no tiene hiperparámetros críticos para ajustar. Aun así, los parámetros L1, L2 y C se pueden ajustar como técnicas de regularización para abordar el sobreajuste. La técnica de regularización L1 maneja un modelo de Regresión Lasso, mientras que L2 opera sobre un modelo de Regresión Ridge. En resumen, la regularización L1 intenta estimar la mediana de los datos, mientras que la regularización L2 intenta estimar la media de los datos. El parámetro C, es una variable de control que conserva las cualidades de modificación de la regularización, al corresponder al inverso del parámetro de regularización Lambda. Los valores más altos de C corresponden a una menor regularización para evitar el sobreajuste del modelo.

8.1.2. Naive Bayes

El modelo Naive Bayes es un modelo de probabilidad bayesiano muy simplificado que opera con una fuerte suposición de independencia. Esto significa que la probabilidad de un atributo no afecta la probabilidad del otro. Dada una serie de n atributos, el clasificador Naive Bayes hace $2n!$ suposiciones independientes, con resultados a menudo correctos [99]. Los clasificadores Naive Bayes son una familia de clasificadores bastante similares al de Regresión Logística y al Clasificador de Vectores de Soporte Lineal (SVC), pero aún más rápidos en el entrenamiento. El precio que se paga por esta eficiencia es a menudo una ligera disminución del rendimiento de generalización (*generalization performance*) comparado con el de los clasificadores lineales. La razón por la que los modelos Naive Bayes son tan eficientes es que aprenden parámetros buscando cada característica individualmente y recopilan estadísticas simples por clase de cada característica. Para la clasificación de datos binarios, se emplea el Bernoulli Naive Bayes (BNB), que asume datos binarios, contando con qué frecuencia cada característica de cada clase es diferente de cero. BNB tiene un único parámetro, α, que controla la complejidad del modelo. La forma en que funciona α es agregando al dato α muchos puntos de datos virtuales que tienen valores positivos para todas las características, proporcionando como resultado un "suavizado" de las estadísticas. Un α grande significa más suavizado, lo que implica modelos menos complejos. El rendimiento del algoritmo es relativamente sólido para la configuración de α, lo que significa que la configuración de α no es crítica para un buen rendimiento [90].

8.1.3. Random Forest

Los modelos basados en árboles han venido proporcionando un rendimiento notable en sus aplicaciones en el ámbito de los Sistemas de Detección de Intrusiones. RFconstruye múltiples árboles de decisión, introduciendo *bootstrapping* en las muestras del conjunto de entrenamiento, usando una selección de características aleatorias. El *bootstrapping* es una técnica estadística de re-muestreo que involucra muestro aleatorio con reemplazo,

La predicción se realiza por el cálculo del voto de los árboles del bosque, ponderado por sus estimaciones de probabilidad. Es decir, la clase pronosticada es la que tiene la estimación de probabilidad media más alta en todos los árboles. Random Forest generalmente exhibe una mejora sustancial en el rendimiento sobre el árbol único y proporciona una tasa de error baja con una resistencia al ruido excepcional [100]. El parámetro *n_estimators* indica el número de árboles que crecen en el modelo RF. Una mayor cantidad de árboles produce modelos más estables, pero requiere más memoria y un mayor tiempo de ejecu-

ción. El parámetro *max_features* se refiere al número de características a considerar cuando se busca la mejor división realizada (*split*). Para la configuración de este parámetro se han considerado *sqrt* (es decir, la raíz cuadrada del número de características) y *log*2 (es decir, el logatirmo en base 2 de número de características)) como valores posibles.

8.1.4. AdaBoost

El AdaBoost es un algoritmo estereotipo de *boosting*, cuya idea básica es seleccionar y combinar un grupo de clasificadores débiles para formar un clasificador fuerte. El algoritmo de *boosting* se adopta como un enfoque repetitivo para producir un clasificador robusto, logrando un pequeño error de entrenamiento aleatorio, a partir de clasificadores débiles ensamblados solo a partir de elección aleatoria. El conjunto se emplea para dibujar registros de entrenamiento actualizando repetidamente la distribución de muestra de los datos que han sido entrenados [101]. El estimador base utilizado en esta implementación es un clasificador de árbol de decisión. Para el ajuste de hiperparámetros, el parámetro *n_estimators* realiza el número máximo de estimadores en los que finaliza el refuerzo. En el caso de un ajuste perfecto, el procedimiento de aprendizaje se detiene antes de tiempo. El parámetro *learning_rate* ó tasa de aprendizaje, se emplea para reducir la contribución de cada clasificador, es decir, el peso aplicado a cada clasificador en cada iteración de *boosting*. Una mayor tasa de aprendizaje aumenta la contribución de cada clasificador. Existe una compensación entre los parámetros *learning_rate* y *n_estimators*.

8.1.5. Máquinas de Vectores de Soporte

La última técnica supervisada en escena son las máquinas de vectores de soporte. Una máquina de vector de soporte clasifica los datos encontrando el mejor hiperplano que separa todos los puntos de datos de una clase de los de la otra clase. Las SVM giran en torno a la noción de un "margen" a cada lado de un hiperplano que separa dos clases de datos. Margen significa el ancho máximo de la losa paralela al hiperplano que no tiene puntos de datos interiores. Se ha demostrado que maximizar el margen y, por lo tanto, crear la mayor distancia posible entre el hiperplano de separación y las instancias de cualquier lado del mismo reduce un límite máximo en el error de generalización esperado [102]. El Kernel controla cómo serán proyectadas las variables de entrada. Existen muchas funciones Kernel, sin embargo las más comunes son lineales, polinómicas y de base radial. Cuando un SVM emplea un núcleo lineal para la clasificación, se puede definir como un SVC. Los modelos lineales pueden ser bastante limitantes en espacios de baja dimensión,

ya que las líneas y los hiperplanos tienen una flexibilidad limitada.

Una forma de hacer que un modelo lineal sea más flexible es agregar más características. Un parámetro crítico es el parámetro de regularización de penalización C, que puede tomar un rango de valores y tiene un efecto dramático en la forma de las regiones resultantes para cada clase. Una escala logarítmica podría ser un buen punto de partida, dado que la regularización es inversamente proporcional a C, por lo que debe ser estrictamente positiva. Al igual que con los modelos lineales, un C pequeño significa un modelo muy restringido, donde cada punto de datos solo puede tener una influencia muy limitada [90].

8.2. Entrenamiento de los modelos

Para los experimentos, todos los modelos se evaluaron dentro de la validación cruzada de k-iteraciones estratificado anidado con un bucle externo con k = 5 y un bucle interno con también k = 5, con SMOTE aplicado en el bucle interno. Como buena práctica, los datasets fueron divididos en conjuntos de entrenamiento y prueba en un porcentaje de 80 % y 20 % respectivamente.

En el bucle interno de las iteraciones el ajuste de hiperparámetros se realiza a través del barrido por cada uno de los parámetros de configuración seleccionados. Para cada configuración, del bucle interno se obtienen cinco valores del área bajo la curva característica operativa del receptor (ROC AUC) los cuales son promediados y numerados de 1 a 5 dependiendo de la iteración externa a la que pertenecen. Los mejores promedios determinan los parámetros que deben ser utilizados en el clasificador. Esta metodología se aplicó para todos los modelos de ML.

En esta sección, se exponen los resultados obtenidos de la implementación de las técnicas de ajuste de aprendizaje automático sobre los datos de entrenamiento, calculando el ROC AUC a partir de las medias de predicción.

8.2.1. Dataset DAD

Para la Regresión Logística se empleó el solucionador liblineal. Se consideró una penalización de $L1$ y $L2$ y se ajustó el parámetro C a valores de 20, 30, 40 y 50. Por la naturaleza estocástica del algoritmo, los resultados obtenidos varían dependiendo de las iteraciones realizadas. La media global de todas las iteraciones es de 0.9756 con una desviación típica de $2{,}061 \times 10^{-3}$. En general, valores grandes de C dan más libertad al modelo, por lo que se espera un mejor desempeño a valores altos de C. Sin embargo, como

se puede ver en la figura 8.1, la variación del parámetro C no mejora significativamente las prestaciones del modelo.

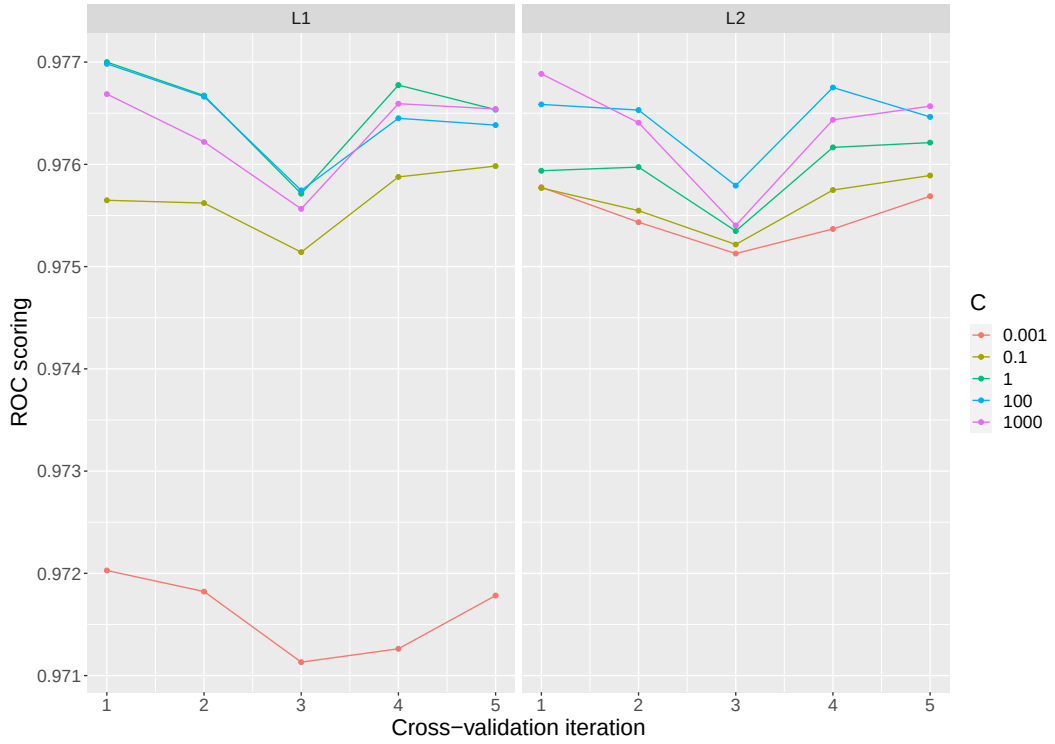

Figura 8.1: Medias de las puntuaciones de Regresión logística por iteración en DAD.

Los resultados presentados corresponden al valor promedio del iteración interna de la validación cruzada, en cada uno de los entornos para cada uno de los modelos. La tabla 8.1 presenta los mejores parámetros seleccionados para la mejor puntuación media en cada uno de las cv internas, así como la desviación típica (SD) obtenida. Los parámetros ajustados que exhibieron los mejores resultados, en la mayoría de los casos, son C igual a 1 y una penalización L1. Por lo tanto, estos mejores parámetros serán los utilizados para la evaluación del modelo en la etapa de prueba.

Tabla 8.1: Mejores modelos de Regresión Logística por iteración en DAD.

cv	**C**	**penalidad**	**media**	**Desviación típica**
1	1	l1	0.9770	$1{,}71 \times 10^{-3}$
2	1	l1	0.9767	$1{,}20 \times 10^{-3}$
3	100	l2	0.9758	$0{,}92 \times 10^{-3}$
4	1	l1	0.9768	$1{,}24 \times 10^{-3}$
5	1000	l2	0.9766	$1{,}87 \times 10^{-3}$

Para encontrar una combinación óptima de hiperparámetros que minimice una función de pérdida predefinida utilizando BNB, los resultados se calcularon variando α entre

0.01, 0.1, 0.5, 1 y 10. El valor medio del ROC AUC presentado en todas las iteraciones es de 0.9654 con una desviación típica de $3{,}229 \times 10^{-3}$. La figura 8.2 muestra la media por iteración de los resultados obtenidos en el ajuste de hiperparámetros. Como Regresión Logística, BNB no mejora el rendimiento proporcionalmente a los valores α en cada iteración.

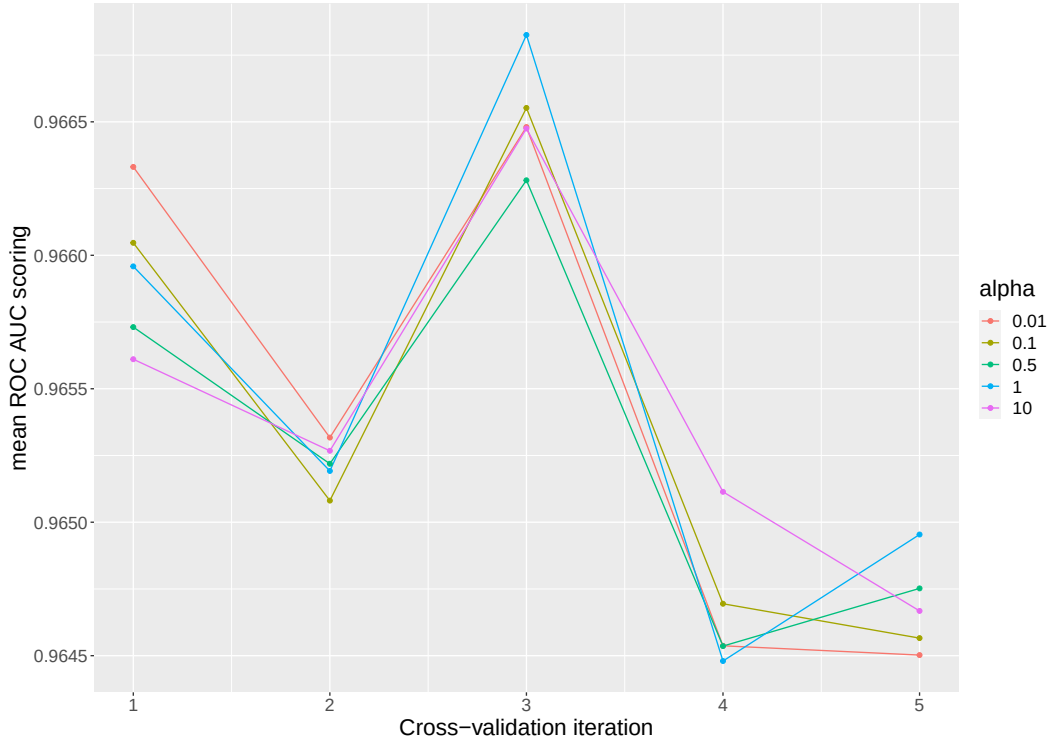

Figura 8.2: Medias de las puntuaciones de Naive Bayes por iteración en DAD.

La tabla 8.2 presenta los parámetros que ofrecen los mejores promedios en cada iteración interna. En donde las configuraciones de α predominantes son 0.01 y 1.

Tabla 8.2: Mejores modelos Naive Bayes por iteración en DAD.

cv	α	**media**	**desviación típica**
1	0.01	0.9663	$3{,}45 \times 10^{-3}$
2	0.01	0.9653	$2{,}69 \times 10^{-3}$
3	1.0	0.9668	$1{,}41 \times 10^{-3}$
4	10.0	0.9651	$3{,}23 \times 10^{-3}$
5	1.0	0.9650	$3{,}86 \times 10^{-3}$

Para el ajuste de hiperparámetros en Random Forest, las configuraciones más importantes son la cantidad de árboles en el bosque (*n_estimators*) y la cantidad de características consideradas para dividir en cada nodo hoja (*max_features*). Durante la ejecución, se probó la evaluación con 100, 200 y 300 árboles, y *sqrt* y *log2* como *max_features*. La media

global obtenida fue de 0.99 con una desviación típica de $5{,}5659 \times 10^{-6}$, alcanzando un ROC AUC de 1 en alguna de las iteraciones. La figura 8.3 presenta los resultados promedio de ROC AUC por iteración, siendo Random Forest el clasificador que presenta el mejor desempeño en la etapa de entrenamiento.

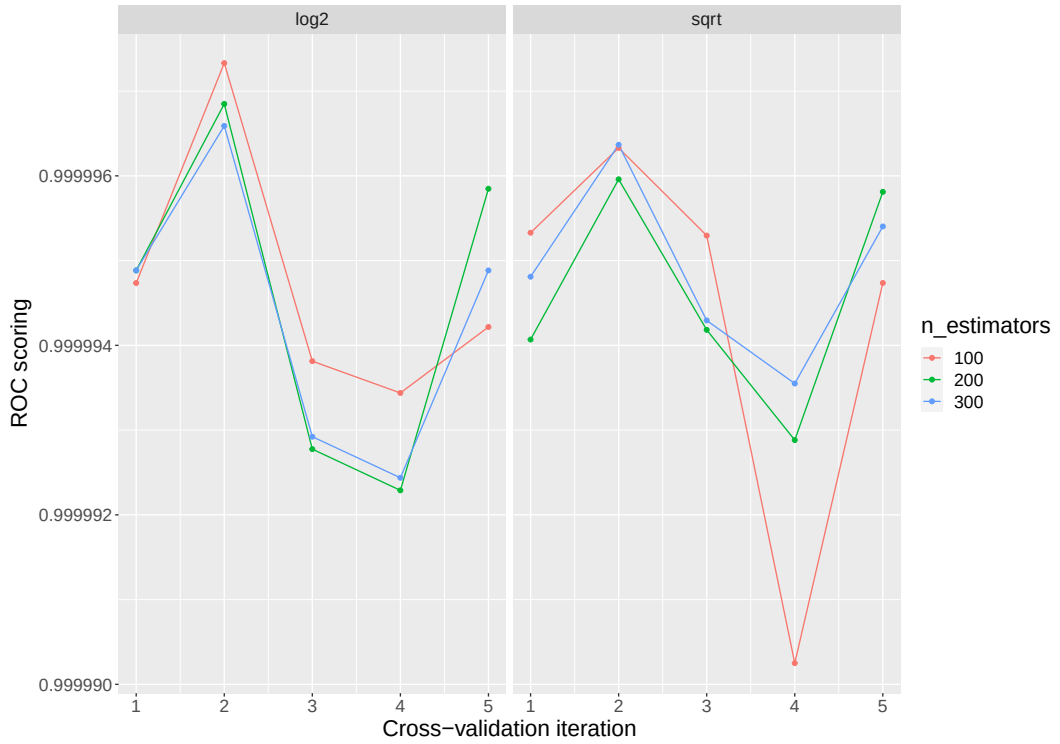

Figura 8.3: Medias de las puntuaciones de Random Forest por iteración en DAD.

La tabla 8.3 muestra los mejores parámetros sobre cada iteración interna para el modelo Random Forest. Existen varias parametrizaciones que obtienen una respuesta óptima en todas las iteraciones. Sin embargo, la cantidad de árboles y características pueden afectar el tiempo y el rendimiento del modelo durante su ejecución.

Tabla 8.3: Mejores modelos Random Forest por iteración en DAD.

cv	**max_features**	**n_estimators**	**media**	**desviación típica**
1	sqrt	100	1	$3{,}75 \times 10^{-6}$
2	log2	100	1	$0{,}98 \times 10^{-6}$
3	sqrt	100	1	$3{,}03 \times 10^{-6}$
4	sqrt	300	0.99	$0{,}99 \times 10^{-6}$
5	log2	200	1	$6{,}12 \times 10^{-6}$

El clasificador AdaBoost por defecto usa un clasificador de árbol de decisión como estimador base, inicializado con max_ depth=1. Para ajustar un buen modelo puede ser deseable aumentar el número de árboles del clasificador, aunque esto puede afectar el

rendimiento del sistema. Para este modelo, el número de estimadores utilizados fue 500, 1000 y 2000, y los valores de la tasa de aprendizaje fueron 0.01, 0.1 0.5 y 1. El ROC AUC promedio global obtenido fue de 0.997 con una desviación típica de $4{,}69 \times 10^{-3}$.

En la figura 8.4 podemos ver cómo el aumento del parámetro de *learning_rate* aumenta el rendimiento del modelo sobre el *n_estimators*. Aun así, se pueden lograr excelentes resultados con niveles medios de *learning_rate* a un bajo número de árboles en el clasificador.

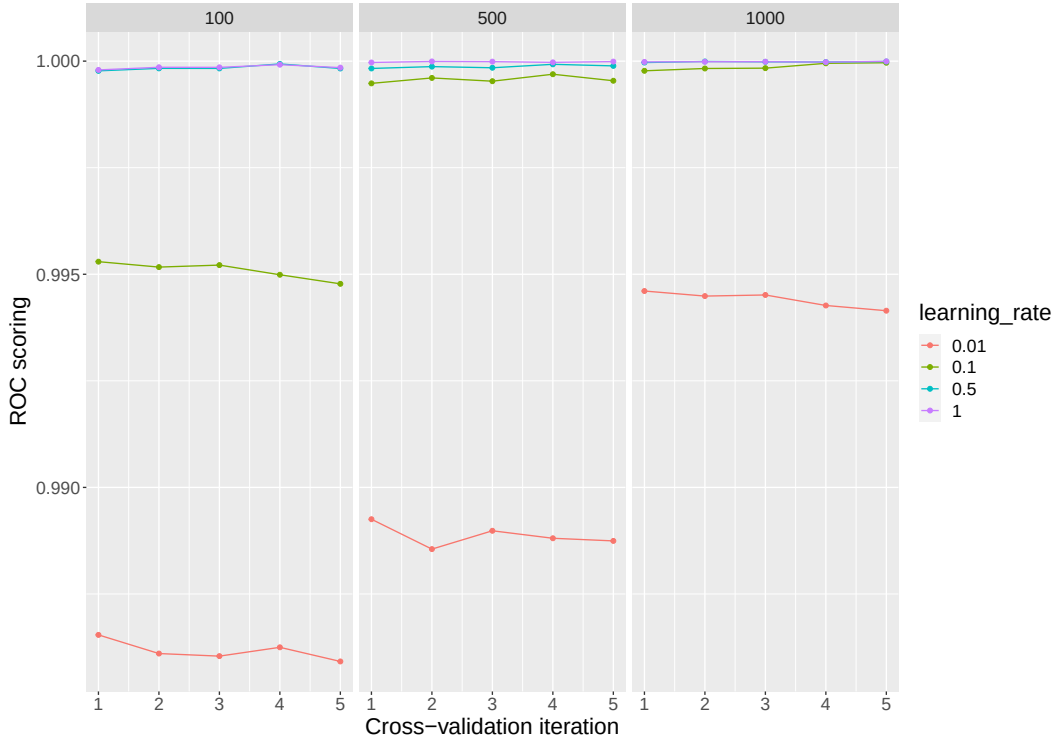

Figura 8.4: Medias de las puntuaciones de AdaBoost por iteración en DAD.

La tabla 8.4 muestra los parámetros que alcanzan el promedio más alto sobre la implementación de validación cruzada interna. Con valores de desviaciones muy bajos, y obteniendo puntuaciones de 1 en algunas iteraciones, los mejores modelos se encuentran con valores de tasa de *learning_rate* de 1 y empleando 1000 árboles para clasificación.

Tabla 8.4: Mejores modelos de AdaBoost por iteración en DAD.

cv	**learning rate**	**n_estimators**	**media**	**desviación típica**
1	1	1000	0.99998	$2{,}63 \times 10^{-5}$
2	1	500	1	$0{,}62 \times 10^{-5}$
3	1	500	0.99998	$0{,}82 \times 10^{-5}$
4	0.5	1000	0.99997	$2{,}10 \times 10^{-5}$
5	1	1000	1	$1{,}12 \times 10^{-5}$

Finalmente, para SVM, se adoptó el clasificador de vector lineal porque tiene más flexibilidad en la elección de funciones de penalización y pérdida y debería escalar mejor a un gran número de muestras. Para mejorar la precisión del modelo, se ajustó el parámetro C, el cual controla el equilibrio entre el límite de decisión y el término de clasificación errónea. Se fija en 0.1, 1, 5 y 10. La figura 8.5 muestra los valores medios obtenidos a lo largo de las iteraciones, mostrando el detrimento del modelo en un valor de C igual a 10.

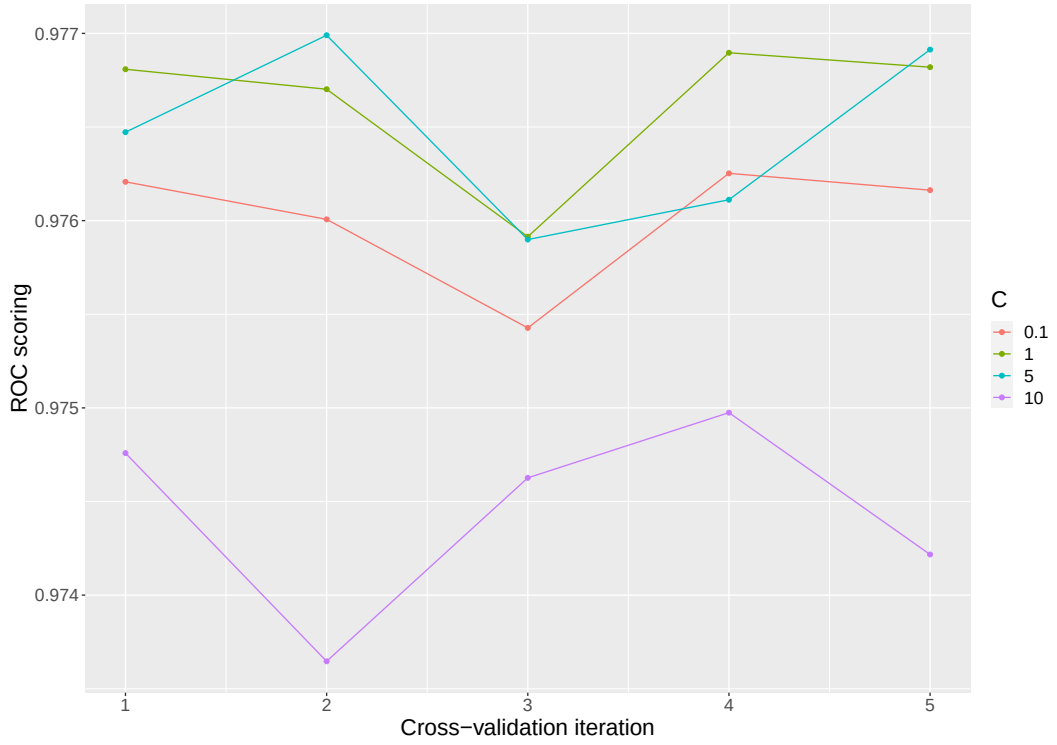

Figura 8.5: Medias de las puntuaciones de Maquinas de vectores de soporte por iteración en DAD.

La tabla 8.5 presenta los resultados obtenidos para las curvas ROC AUC en cada una de las iteraciones para el modelo de SVM.En la etapa de entrenamiento, este clasificador obtiene medias de clasificación muy altas, alcanzando un valor promedio máximo de 0.9759 y una desviación típica media de $1{,}7411 \times 10^{-3}$.

Tabla 8.5: Mejores modelos SVM por iteración en DAD.

cv	C	**media**	**desviación típica**
1	1	0.9768	$1{,}55 \times 10^{-3}$
2	5	0.9770	$1{,}39 \times 10^{-3}$
3	1	0.9759	$0{,}96 \times 10^{-3}$
4	1	0.9769	$1{,}22 \times 10^{-3}$
5	5	0.9769	$1{,}98 \times 10^{-3}$

8.2.2. Dataset CIDAD

Para la Regresión Logística se consideró una penalización de $L1$ y $L2$ y se ajustó el parámetro C a valores de 0.001, 0.1, 1, 100 y 1000. Por lo general, se espera un mejor rendimiento con valores altos de C, sin embargo como se puede ver en la figura 8.6, los mejores resultados se obtuvieron a valores de C relativamente bajos (0.1).

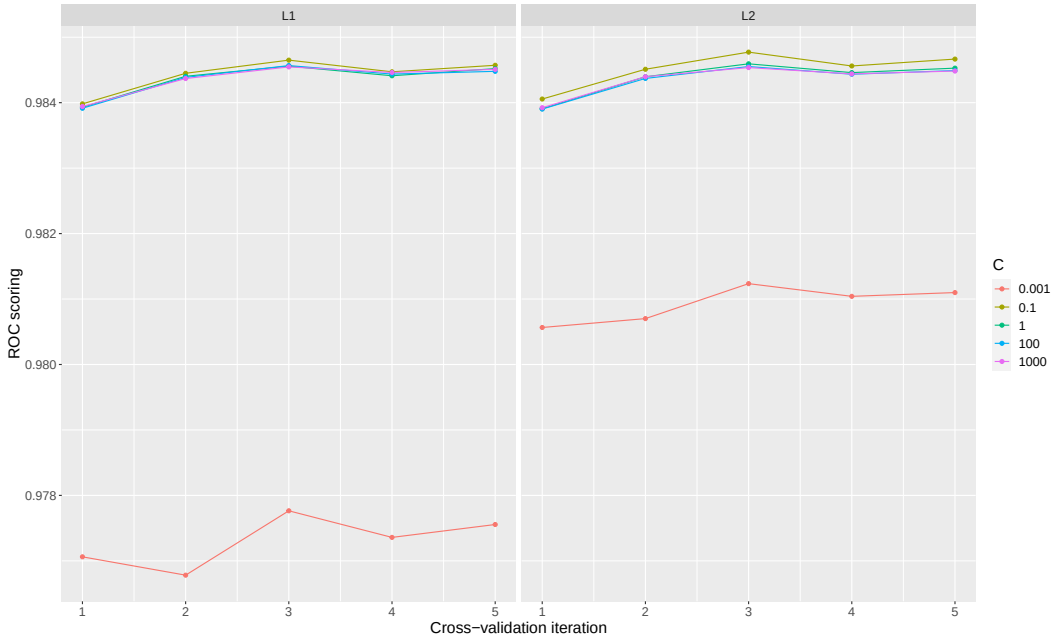

Figura 8.6: Medias de las puntuaciones de Regresión Logística por iteración en CIDAD.

La tabla 8.6 presenta los mejores parámetros obtenidos para la mejor puntuación media y la SD en cada uno de los cv internos para el modelo de Regresión Logística. Como se puede ver en esta, los mejores resultados en todas las iteraciones fueron de 0.984 con penalización L2 y un valor C de 0.1, presentando este mismo valor como media global con una desviación típica de 1×10^{-3}. Los parámetros mostrados en la tabla 8.6 son los seleccionados para la evaluación del modelo en la división de prueba.

Tabla 8.6: Mejores modelos de Regresión logística por iteración en CIDAD.

cv	C	**penalty**	**media**	**desviación típica**
1	0.1	l2	0.9841	$1{,}28 \times 10^{-3}$
2	0.1	l2	0.9845	$0{,}66 \times 10^{-3}$
3	0.1	l2	0.9848	$1{,}01 \times 10^{-3}$
4	0.1	l2	0.9846	$1{,}27 \times 10^{-3}$
5	0.1	l2	0.9847	$0{,}82 \times 10^{-3}$

Para Naive Bayes, los resultados se calcularon variando el parámetro α en 0.01, 0.1,

0.5, 1 y 10. El valor ROC AUC medio presentado en todas las iteraciones es de 0.9390 con una desviación típica de $7{,}53 \times 10^{-3}$. La figura 8.7 muestra el promedio por iteración de los resultados obtenidos en el ajuste de hiperparámetros para NB, en donde, al igual que para DAD, las variaciones del parámetro α no mejoran el rendimiento de forma sistemática en cada iteración.

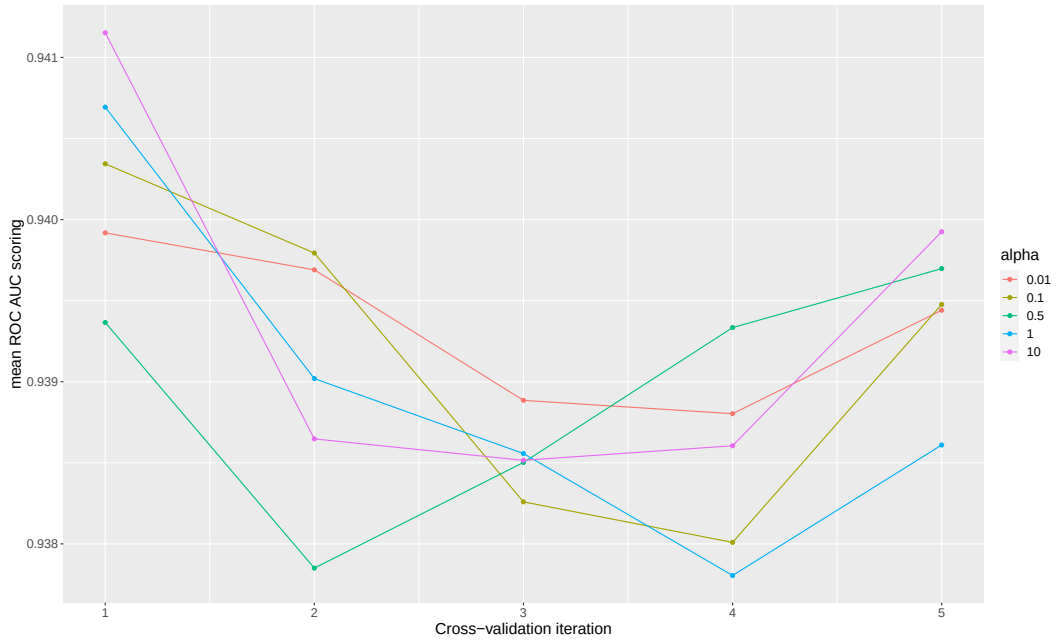

Figura 8.7: Medias de las puntuaciones de Naive Bayes por iteración en CIDAD

La tabla 8.7 presenta los parámetros que ofrecen una mejor puntuación en cada iteración interna. Estos parámetros se utilizarán en la evaluación del modelo.

Tabla 8.7: Mejores modelos Naive Bayes por iteración en CIDAD.

cv	α	**media**	**desviación típica**
1	10.0	0.9412	$9{,}39 \times 10^{-3}$
2	0.1	0.9398	$4{,}90 \times 10^{-3}$
3	0.01	0.9389	$8{,}72 \times 10^{-3}$
4	0.5	0.9393	$6{,}93 \times 10^{-3}$
5	10.0	0.9399	$6{,}81 \times 10^{-3}$

Durante la ejecución de Random Forest, la evaluación se probó con 100, 200 y 300 árboles, y *sqrt* y *log*2 como *max_features*. La media global obtenida fue una puntuación perfecta. La figura 8.8 presenta los resultados promedio de ROC AUC por iteración. Los puntos altos en el gráfico significan algunos valores de 1 en ese ciclo. Random Forest presenta el mejor desempeño en la etapa de entrenamiento.

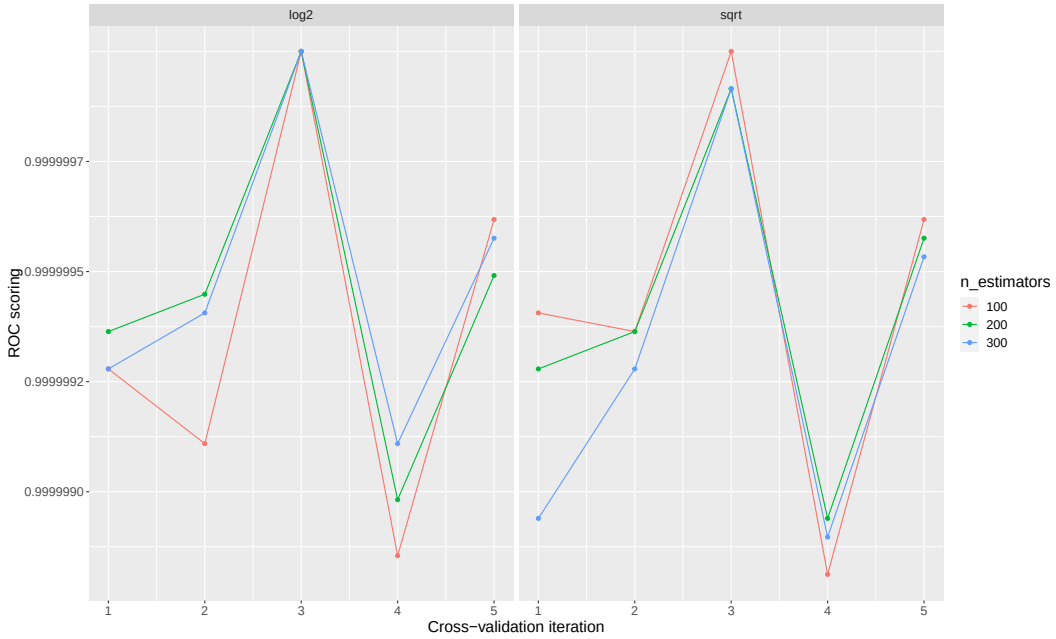

Figura 8.8: Medias de las puntuaciones de Random Forest por iteración en CIDAD

Varias parametrizaciones obtienen una puntuación perfecta. Sin embargo, la cantidad de árboles y características pueden afectar el tiempo y el rendimiento del modelo durante su ejecución. La tabla 8.8 muestra todas las combinaciones de parámetros que funcionan perfectamente en cada iteración interna.

Tabla 8.8: Mejores modelos de Random Forest por iteración en CIDAD.

cv	**max_features**	**n_estimators**	**media**	**desviación típica**
1	sqrt	100	1	$9{,}89 \times 10^{-7}$
2	log2	200	1	$7{,}54 \times 10^{-7}$
3	sqrt	100	1	0
4	log2	300	1	$1{,}39 \times 10^{-6}$
5	log2	100	1	$6{,}62 \times 10^{-7}$

Por otro lado, el clasificador AdaBoost, usando árboles de decisión como estimador base con 500, 1000 y 2000 árboles y max_depth=1, realiza la hiperparametrización mediante el barrido por los valores de learning_rate en 0.001, 0.1, 0.5 y 1. El ROC AUC promedio global obtenido fue 0.99 con una desviación típica de $4,04 \times 10^{-3}$

Típicamente, cuanto menor sea la tasa de aprendizaje, mayor será el número de árboles necesario, como se presenta en la figura 8.9,

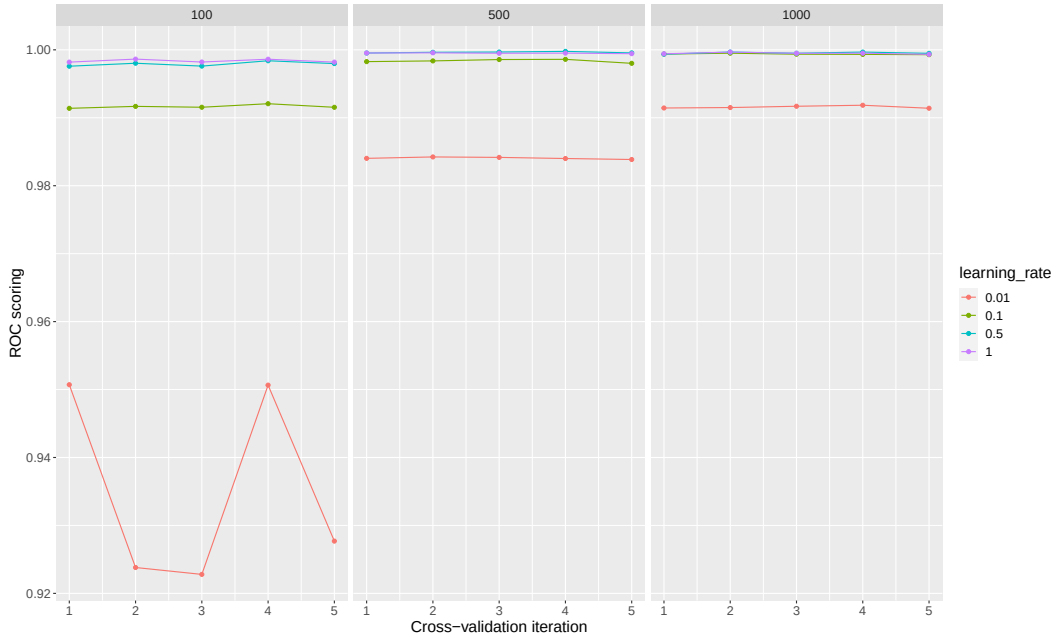

Figura 8.9: Medias de las puntuaciones de AdaBoost por iteración en CIDAD

La tabla 8.9 muestra los parámetros que alcanzan el promedio más alto sobre la implementación de la cv interna, alcanzando una media de 0.99 en todas las iteraciones. En las mejores estimaciones de los modelos, predomina un ajuste de *learning rate* de 0.5 y un *n_estimators* de 500.

Tabla 8.9: Mejores modelos de AdaBoost por iteración en CIDAD.

cv	**learning rate**	**n_estimators**	**media**	**desviación típica**
1	1	500	0.99	$4{,}60 \times 10^{-4}$
2	0.5	1000	0.99	$3{,}96 \times 10^{-4}$
3	0.5	500	0.99	$3{,}87 \times 10^{-4}$
4	0.5	500	0.99	$3{,}38 \times 10^{-4}$
5	0.5	500	0.99	$4{,}39 \times 10^{-4}$

Finalmente, para mejorar la precisión de SVM, se ajustó el parámetro de penalización C, y se fija en 0.01, 1, 5 y 10. La SVC alcanzó una puntuación media de ROC AUC de 0.98, con una desviación típica de $1{,}04 \times 10^{-3}$.

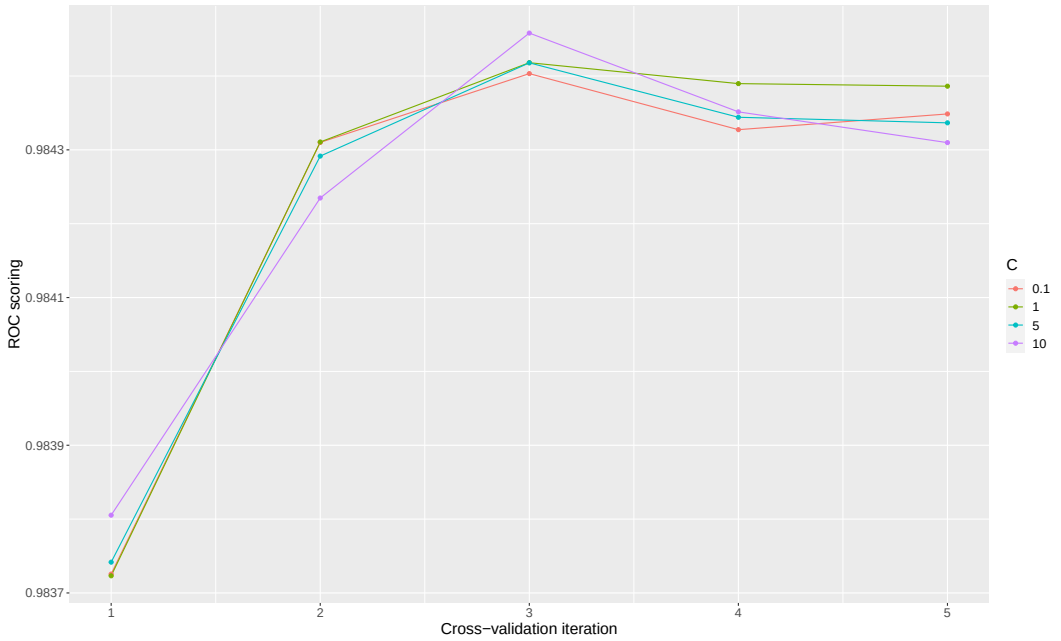

Figura 8.10: Medias de las puntuaciones de Maquinas de vectores de soporte por iteración en CIDAD

La figura 8.10 muestra la puntuación media obtenida a lo largo de las iteraciones, donde las modificaciones realizadas en el parámetro C no mejoran el rendimiento del modelo. Las mejores puntuaciones medias del modelo se encuentran en valores de C de 10 y 1, como se puede observar en la tabla 8.10

Tabla 8.10: Mejores modelos de SVM por iteración en CIDAD.

cv	C	**media**	**desviación típica**
1	10	0.9838	$1{,}39 \times 10^{-3}$
2	1	0.9843	$7{,}22 \times 10^{-4}$
3	10	0.9845	$1{,}04 \times 10^{-3}$
4	1	0.9844	$1{,}24 \times 10^{-3}$
5	1	0.9844	$8{,}34 \times 10^{-4}$

Capítulo 9

Detección de anomalías

La estimación del rendimiento del modelo de aprendizaje automático se verifica en la validación cruzada del conjunto de datos no utilizados para entrenar el modelo. Una tarea de clasificación se puede evaluar de muchas maneras diferentes dependiendo de los objetivos que quieran alcanzarse y las tareas para las cuales ha sido diseñado. Esta sección presenta los resultados de cuatro métricas tradicionales para todos los modelos de aprendizaje automático seleccionados: Exactitud (Accuracy), Precisión (Precision), Exhaustividad (Recall), y F1. Para obtener el mejor modelo, se realiza una comparación entre el rendimiento de los algoritmos en cada uno de los entornos.

9.1. Entorno MQTT

Los parámetros aplicados a cada uno de los modelos para el *dataset* DAD son los que se definen en la sección 8.2.1.

La figura 9.1 (a) muestra los valores promedio de Exactitud en cada uno de los modelos sobre todas las iteraciones realizadas. Esta métrica determina el porcentaje total de elementos clasificados correctamente, lo que permite establecer en general que tan bien funciona el clasificador. El clasificador Naive Bayes presenta el rendimiento más bajo para esta métrica en todas las iteraciones y los modelos basados en árboles de decisión presentan los mejores resultados, obteniendo valores muy cercanos al resultado óptimo.

La Precisión determina que tan bien clasifica los valores positivos en el total de elementos identificados como positivos. Es esta caso, una alto valor en precisión indica que muchos paquetes normales han sido clasificados como anómalos. La figura 9.1 (b) presenta

los valores obtenidos para Precisión en cada una de las iteraciones realizadas. Los modelos lineales, NB, SVM y LR presentan una alta tasa de falsos positivos (FP), como puede verse en las matrices de confusión presentadas en las tablas 9.2, 9.3 y 9.4, respectivamente, lo que implica un detrimento en esta métrica. Los clasificadores basados en árboles de decisión alcanzan valores de 1 en dos de las cinco iteraciones realizadas.

Debido a que el interés se basa en la detección de anomalías en sistemas de intrusiones, La clasificación de paquetes anómalos como normales puede implicar un riesgo para el sistema, y por lo tanto es importante minimizar los falsos negativos (FN). La métrica de Exhaustividad, que se ilustra figura 9.1 (c), nos da un indicio de que tan bien clasifica elementos identificados correctamente como positivos del total de positivos verdaderos. Los mejores resultados se obtienen con NB, AdaBoost y RF, obteniendo clasificaciones aceptables para SVM y bajas para LR, por debajo del 0.5.

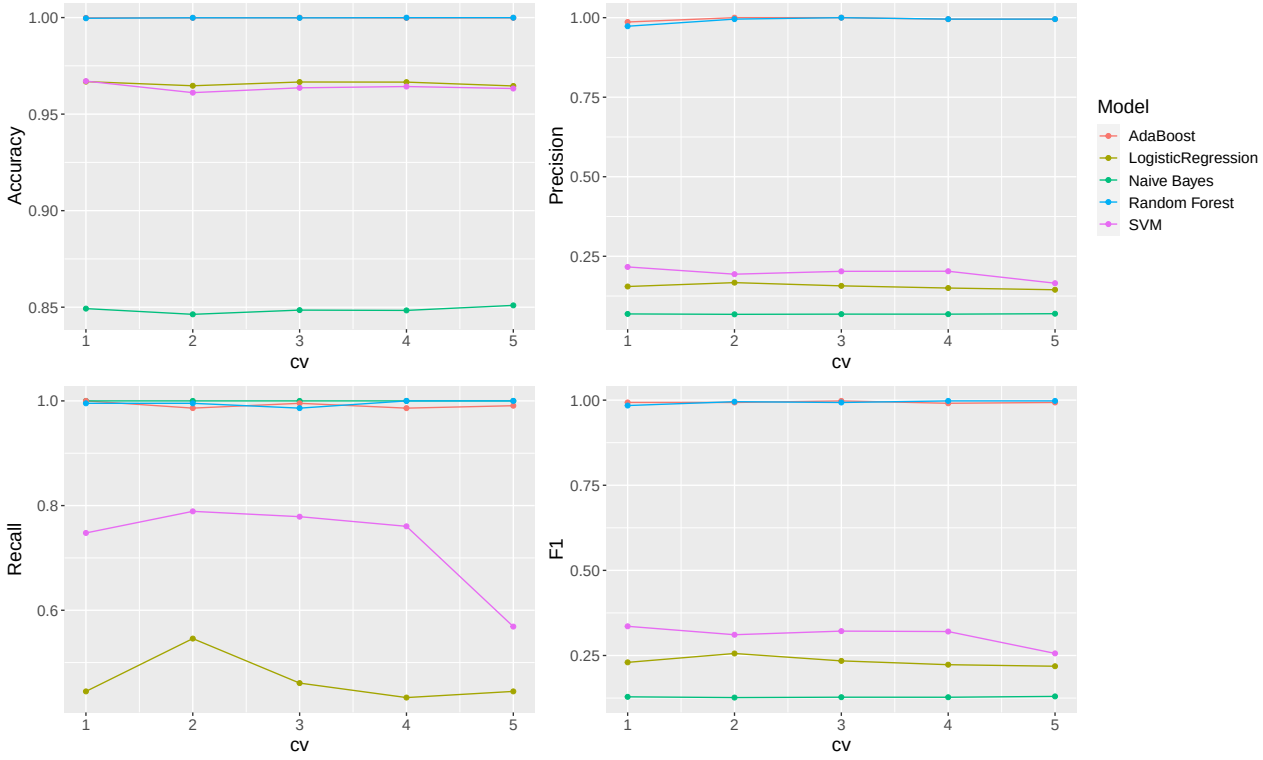

Figura 9.1: Métricas por iteración: a) Exactitud, b) Precisión, c) Exhaustividad, d) F1.

Por otro lado, para la métrica F1 representada en la figura 9.1 (d), AdaBoost y RF son los clasificadores que mejores resultados logran en todas las iteraciones, alcanzando resultados sobresalientes en comparación con el resto de clasificadores. La métrica F1 es una media armónica entre los valores de Precisión y Exhaustividad, por lo tanto, los clasificadores de NB, LR y SVM que presentan valores muy bajos en Precisión, tienen un bajo rendimiento en esta métrica.

La tabla 9.1 presenta las medias de todas las iteraciones de todos los clasificadores junto con su desviación típica.

Tabla 9.1: Mejores modelos por iteración en DAD.

	NB	**AdaBoost**	**LR**	**RF**	**SVM**
media Exactitud	0.8486	0.9998	0.9658	0.9998	0.9639
media Precisión	0.0684	0.9954	0.1548	0.9919	0.1962
media Exhaustividad	1	0.9917	0.4659	0.9954	0.7289
media F1	0.1279	0.9935	0.2323	0.9935	0.309
std Exactitud	$1{,}6767 \times 10^{-3}$	$5{,}5900 \times 10^{-5}$	$1{,}1226 \times 10^{-3}$	$1{,}2706 \times 10^{-3}$	$2{,}1527 \times 10^{-3}$
std Precision	$7{,}4721 \times 10^{-4}$	$5{,}5421 \times 10^{-3}$	$8{,}3510 \times 10^{-3}$	$1{,}0682 \times 10^{-2}$	$1{,}9084 \times 10^{-3}$
std Exhaustividad	0	$5{,}9923 \times 10^{-3}$	$4{,}5742 \times 10^{-2}$	$5{,}644 \times 10^{-3}$	$9{,}0932 \times 10^{-2}$
std F1	$1{,}3089 \times 10^{-3}$	$2{,}5304 \times 10^{-3}$	$1{,}4532 \times 10^{-2}$	$5{,}6361 \times 10^{-3}$	$3{,}0803 \times 10^{-2}$

En términos generales, el clasificador Naive Bayes tiene los valores de Exactitud más bajos entre todos los modelos, pero es el único que obtiene una Exhaustividad perfecta. A pesar de ser un modelo que presenta una Exactitud aceptable, una alta tasa de detección de verdaderos positivos (TP) garantiza un buen rendimiento ante presencia de anomalías. Sin embargo, para este modelo se obtiene un índice Kappa de 0.11, lo que implica una concordancia dada por casualidad.

Tabla 9.2: Matriz de confusión de NB en DAD.

		Estimado	
		Normal	Anómalo
Real	Normal	82066	14828
	Anómalo	0	1088

Como era de esperar, debido a la similitud en la construcción de los modelos, los clasificadores LR y SVM ofrecen resultados muy similares. Como se aprecia en la tabla 9.1 ambos clasificadores tienen una Exactitud de 0.96, una Precisión de alrededor de 0.1 y un F1 de 0.3. Las matriz de confusión para LR se presenta en las tabla 9.4 y para SVM en la tabla 9.3. Adicionalmente, ambos clasificadores obtienen coeficientes Kappa muy bajos, de 0.22 para LR y de 0.30 para SVM, haciendo estos clasificadores poco confiables.

Tabla 9.3: Matriz de confusión de SVM en DAD.

		Estimado	
		Normal	Anómalo
Real	Normal	93649	3245
	Anómalo	295	793

Tabla 9.4: Matriz de confusión de LR en DAD.

		Estimado	
		Normal	Anómalo
Real	Normal	94130	2764
	Anómalo	581	507

Por otra parte, los clasificadores basados en árboles de decisión son los que presentan mejor rendimiento para este tipo de entornos. Tanto el clasificador RF como el AdaBoost presentan valores de Exactitud muy cercanos al 1, valores casi perfectos. Las matriz de confusión del modelo RF se muestra en la tabla 9.6 y la del modelo AdaBoost se presenta en la tabla 9.5. Adicionalmente, presentan unos índices de Kappa Cohen muy altos, 0.99 en ambos casos, lo que garantiza la estabilidad y fiabilidad en el clasificador.

Tabla 9.5: Matriz de confusión de AdaBoost en DAD.

		Estimado	
		Normal	Anómalo
Real	Normal	96889	5
	Anómalo	9	1079

Tabla 9.6: Matriz de confusión de RF en DAD.

		Estimado	
		Normal	Anómalo
Real	Normal	96885	9
	Anómalo	5	1083

9.2. Entorno CoAP

Tras la aplicación de los modelos para el *dataset* CIDAD, la figura 9.2 muestra los valores medios para las métricas tradicionales, en cada uno de los modelos sobre todas las iteraciones realizadas. En esta gráfica, podemos identificar a *Random Forest* como el clasificador que presenta mejores resultados en todas las métricas en todas las iteraciones.

El clasificador *Naive Bayes* presenta nuevamente el rendimiento más bajo en todas las métricas, aunque en Exactitud sigue logrando un buen resultado de 0.9210, que se observa en la tabla 9.7. NB obtiene unas puntuaciones bajas en Precisión y por consiguiente en F1, debido al alto número de FN y FP que presenta, tal y como se muestra en la matriz de confusión en la tabla 9.8. Presenta un índice de Kappa muy bajo de 0.16, lo que indica que

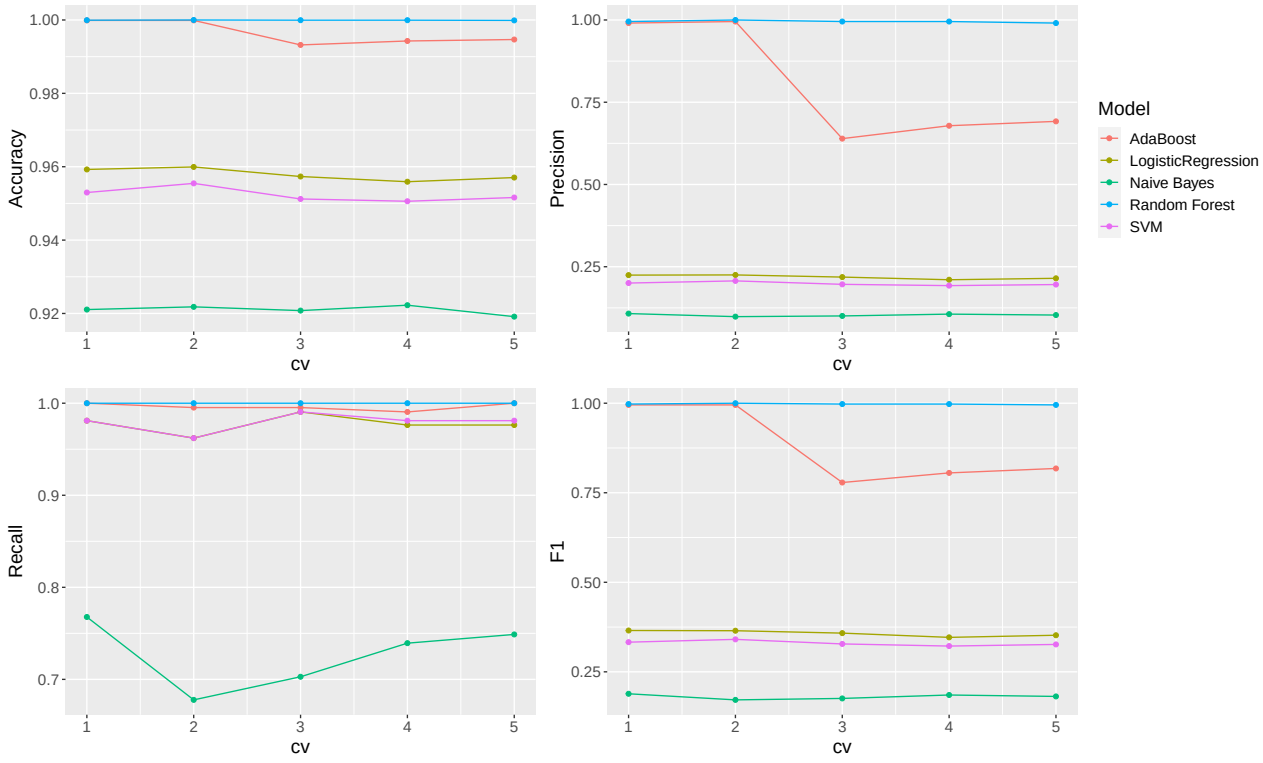

Figura 9.2: Métricas por iteración: a) Exactitud, b) Precisión, c) Exhaustividad, d) F1.

que la concordancia observada es precisamente la que se espera a causa exclusivamente del azar.

Tabla 9.8: Matriz de confusión de NB en CIDAD.

		Estimado	
		Normal	Anómalo
Real	Normal	80500	6682
	Anómalo	288	768

Los modelos lineales LR y SVM, ofrecen resultados similares, obteniendo ambos una Exactitud satisfactoria de 0.95. La matriz de confusión para el clasificador de LR se muestra en la tabla 9.10, y obtiene un índice Kappa de 0.34. El clasificador SVM presenta su matriz de confusión en la tabla 9.9, con índice Kappa de 0.32. Estos modelos consiguen valores muy bajos de precisión y de F1, por las altas tasas de de falsos positivos (FP) .

Tabla 9.9: Matriz de confusión de SVM en CIDAD.

		Estimado	
		Normal	Anómalo
Real	Normal	83001	4181
	Anómalo	22	1034

Tabla 9.7: Mejores modelos por iteración en CIDAD.

	NB	**AdaBoost**	**LR**	**RF**	**SVM**
media Exactitud	0.9210	0.9964	0.9579	0.9999	0.9524
media Precisión	0.1030	0.7991	0.2187	0.9953	0.1984
media Exhaustividad	0.7272	0.9962	0.977	1	0.9791
media F1	0.1805	0.8785	0.3573	0.9976	0.3299
std. Exactitud	$1{,}1980 \times 10^{-3}$	$3{,}2419 \times 10^{-3}$	$1{,}6584 \times 10^{-3}$	$4{,}0069 \times 10^{-3}$	$1{,}9381 \times 10^{-3}$
std. Precisión	$3{,}8186 \times 10^{-3}$	0.1779	$6{,}2834 \times 10^{-3}$	$3{,}3197 \times 10^{-3}$	$5{,}5056 \times 10^{-3}$
std. Exhaustividad	$3{,}6408 \times 10^{-2}$	$3{,}9638 \times 10^{-3}$	$1{,}0289 \times 10^{-3}$	0	$1{,}0395 \times 10^{-2}$
std. F1	$6{,}9215 \times 10^{-3}$	0.10756	$8{,}2403 \times 10^{-3}$	$1{,}6677 \times 10^{-3}$	$7{,}1099 \times 10^{-3}$

Tabla 9.10: Matriz de confusión de LR en CIDAD.

		Estimado	
		Normal	Anómalo
Real	Normal	83491	3691
	Anómalo	24	1032

Los modelos basados en árboles de decisión presentan los mejores resultados, obteniendo en algunas ocasiones el resultado óptimo. Para RF, en todas las métricas alcanza valores por encima de 0.99. La matriz de confusión para el modelo RF se presenta en la tabla 9.11. Además, obtiene un índice Kappa de 1, lo que significa resultados excelentes e ideales para un clasificador. El modelo AdaBoost obtiene también un alto índice Kappa de 0.87 y una media de Exactitud y Exhaustividad por encima del 0.99. La matriz de confusión corresponde a la de la tabla 9.12.

Tabla 9.11: Matriz de confusión de RF en CIDAD.

		Estimado	
		Normal	Anómalo
Real	Normal	87177	5
	Anómalo	0	1056

Tabla 9.12: Matriz de confusión de AdaBoost en CIDAD.

		Estimado	
		Normal	Anómalo
Real	Normal	86867	315
	Anómalo	4	1052

9.3. Comparativa entre los entornos

Tras hacer una comparativa entre la tabla 9.1 y la tabla 9.7, que corresponden a los resultados obtenidos por los clasificadores en MQTT y CoAP, respectivamente, se puede observar que los clasificadores tienen respuestas similares para ambos entornos. Todos los clasificadores obtienen una media en Exactitud por encima del 92 %, exceptuando el clasificador NB en MQTT con un valor de 0.8486 debido a sus altos valores de FP.

Como se mencionó anteriormente, en el caso de detección de anomalías, si un paquete normal es considerado como anormal y es descartado, el sistema no se ve afectado. Sin embargo, es necesario minimizar la cantidad de paquetes anormales que son clasificados como normales FN. Para ambos *datasets* la medida de Exhaustividad es buena a excepción del modelo LR en MQTT con un valor de 0.46 . Los clasificadores presentan resultados de Precisión muy bajos en los clasificadores lineales (NB, LR y SVM), y estos resultados también repercuten en las medidas obtenidas de F1.

Contrariamente, los árboles de decisión presentan muy buenos resultados tanto para el entorno CoAP como para el MQTT, obteniendo valores por encima de 98 % para todos los clasificadores en todas las métricas, siendo el clasificador de RF el clasificador óptimo en ambos entornos. Esto demuestra que los clasificadores por árboles de decisión son las mejores opciones de clasificación en estos entornos, con estas características, usando las técnicas descritas.

Capítulo 10

Conclusiones y trabajos futuros

Tras el inminente avance de las redes IoT en los diferentes entornos cotidianos y la difícil tarea de incorporar sistemas seguros, los Sistemas de Detección de Intrusiones han resultado una herramienta versátil para garantizar su integridad y confidencialidad. Las técnicas de aprendizaje automático han ganado credibilidad recientemente en una aplicación exitosa para la detección de anomalías en la red, incluidas las redes IoT. Por consiguiente, esta tesis presenta un sistema de detección de anomalías de tráfico mediante el uso de algoritmos de aprendizaje automático en entornos IoT con gran eficiencia y bajas tasas de error.

A continuación, se describen las conclusiones de la tesis y algunos de los posibles trabajos futuros que pueden continuar desarrollándose como resultado de la investigación.

10.1. Conclusiones

Todo IDS basado en anomalías requiere de un conjunto de datos específico que cumpla con una serie de requisitos para poder ser empleado para clasificación. Una de las principales aportaciones de esta tesis es la publicación de dos *datasets* realistas, fáciles de manejar y perfectamente etiquetados, basados en la reproducción de un escenario real, con un número suficiente de muestras, extracción de características específicas y mezcla de anomalías, que posibilitan la aplicación de algoritmos de aprendizaje supervisado.

Para la generación de los *datasets* se construyeron dos escenarios virtuales modificando el intercambio de mensajes en cada uno de ellos. La utilización de protocolos de amplio uso, nos permitió generar entornos flexibles, prácticos y que simulan un sistema real,

considerando las condiciones de una red IoT. La inyección de anomalías controladas en cinco de los siete días de la semana, interceptación, modificación, duplicación, y la mezcla de ellas, posibilitó exhibir diferentes comportamientos en la red.

El primero de los *datasets* generados, llamado DAD, realiza el intercambio de mensajes haciendo uso del protocolo MQTT en la capa de aplicación. Contiene 67.848 flujos y 101.583 paquetes. En total, el $3,4\,\%$ de los paquetes son paquetes UDP, mientras que el $96,9\,\%$ son paquetes TCP. Presenta flujos en TCP/MQTT unidireccionales y el $1,08\,\%$ de los paquetes fueron etiquetados como anómalos. El segundo *dataset* generado, denominado CIDAD, hace el intercambio de mensajes implementando el protocolo CoAP. Contiene un total de 88.238 paquetes, entre los que encontramos el uso de los protocolos ARP, IP e IPV6. 64.459 paquetes corresponden a UDP, y todas las anomalías se realizan sobre los datos CoAP. Esto significa que todos los paquetes anómalos pertenecen a UDP/CoAP y representan el $1,2\,\%$ del total de paquetes presentados en el *dataset*.

La mayoría de los sistemas de detección de anomalías de tráfico muestran las alteraciones en los paquetes de red. Por el tipo de redes aquí definidas, las modificaciones fueron realizadas en el *payload* de los mensajes de la capa de aplicación. Para realizar esto de la forma más realista posible, se tomaron datos reales de los sensores de temperatura del CPD del CITIC, y se desarrolló un modelado matemático aplicando métodos paramétricos y no paramétricos. Tras la exploración de diversas técnicas, se encontró que, debido a las condiciones de las señales, el uso de descomposición de series temporales con STL tenía en cuenta distribución, eventos pasados, estacionariedad y estacionalidad en los datos presentados y requería una menor carga computacional, manteniendo la exactitud de los datos.

Para hacer que los datos fueran perfectamente comprensibles para el clasificador, se utilizaron técnicas de generación de características y discretización proporcionando una clasificación de valores de alto orden al suavizar las relaciones entre las observaciones. Se aplicó RFE como técnica para reducción de características, eliminando dependencias y colinealidades entre características, minimizando errores y esfuerzos computacionales.

Finalmente, para la optimización del modelo, La técnica de balanceo de datos con SMOTE, junto con la cv estratificada de k-iteraciones, garantizaron la presencia de la clase minoritaria de una manera simple que mejora sustancialmente el desempeño del clasificador y la hiperparametrización del modelo evita el sobreajuste.

Para validar los conjunto de datos, se seleccionaron cinco algoritmos de aprendizaje automático ampliamente utilizados: LR, NB, RF, AdaBoost Y SVM.

En la etapa de entrenamiento, utilizando la métrica ROC AUC, los mejores valores en media alcanzados en DAD fueron 0.9770 para LR y SVM, 0.9868 para NB, 1 para RF y 0.99 para Adaboost, en diferentes valores de combinación de parámetros. Los mejores

valores alcanzados en CIDAD fueron 0.9848 para LR, 0.9412 para NB, 1 para RF, 0.99 en Adaboost y 0.9845 en SVM en diferentes valores de combinación de parámetros.

Para la etapa de prueba, se tomaron cuatro métricas tradicionales para determinar la calidad del clasificador: Exactitud, Precisión, Exhaustividad y F1. Además, se empleó el índice Kappa, que determina cómo afecta el azar a las decisiones.

Los clasificadores tienen respuestas similares en DAD y CIDAD. La mayoría de los ellos obtienen una media en Exactitud por encima del 92 %, menos el clasificador NB en MQTT, que consigue un valor de 0.8486 debido a sus altos valores de FP. Según los resultados logrados con esta métrica, los modelos basados en árboles de decisión son los mejores clasificadores para ambos *datasets* diseñados, alcanzando valores casi perfectos (cercanos a 1,0) y presentando las desviaciones típicas más bajas, lo que garantiza la estabilidad en el clasificador.

En el caso de detección de anomalías, cuyo interés es minimizar la cantidad de FN, la Exhaustividad es una métrica significativa. Para ambos *datasets* la medida de Exhaustividad se encuentra por encima del 0.7, a excepción del modelo LR en MQTT. A pesar de obtener medidas de Exactitud al menos buenas, los clasificadores lineales (NB, LR y SVM) presentan bajos valores de Precisión y, por consiguiente, de F1, junto con índices Kappa bajos, lo que los hace clasificadores poco fiables.

En contraste, los árboles de decisión presentan muy buenos resultados tanto para el entorno CoAP como para el MQTT, siendo el clasificador de RF el clasificador óptimo en ambos entornos.

En conclusión, se ha llevado a cabo una detección exitosa de anomalías de trafico en redes IoT mediante la aplicación de modelos de aprendizaje automático en *datasets* con protocolos específicos y con alteraciones en el *payload* de los mensajes.

10.2. Trabajos futuros

La continua expansión en la implementación de redes y dispositivos IoT y el interés constante en la aplicación de tecnologías de punta, trae consigo nuevos tipos de amenazas y vulnerabilidades. Esta sección propone una serie de trabajos con propuestas de interés, siguiendo la línea de investigación expuesta en la presente tesis.

Con la generación de dos nuevos *datasets* en redes IoT con anomalías en la capa de aplicación que utilizan protocolos CoAP Y MQTT, se plantea como trabajo futuro la aplicación de otro tipo de técnicas de Machine Learning, así como Deep Learning, a los *datasets* presentados, y así pues, realizar una comparativa que permita ratificar o debatir los resultados aquí obtenidos.

Así mismo, pueden validarse otros *datasets* recientes, empleando las técnicas de optimización aquí propuestas (o adecuaciones de estas considerando las condiciones de los *datasets*), junto con los modelos de aprendizaje automáticos empleados, y realizar el análisis que permita corroborar la calidad de los clasificadores.

Además, los conceptos proporcionados en esta tesis pueden servir de base para la creación de nuevos *datasets* usando diferentes protocolos IoT, incluso protocolos que pertenezcan a otros niveles de la pila TCP/IP, como Modbus, BACnet, etc., que incorporen anomalías de duplicación, interceptación y modificación.

En muchos dominios industriales de IoT, el gran número de sensores y actuadores necesarios en la monitorización, hace que replicarlos en entornos simulados requieran de una modelización matemática. Por lo tanto, las lecciones aprendidas para la modelización de datos de sensores pueden ser aplicadas en otros entornos.

Bibliografía

[1] N. Moustafa, B. Turnbull, and K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things," *IEEE Internet of Things Journal*, vol. 6, pp. 4815–4830, June 2019.

[2] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things*, vol. 7, p. 100059, 2019.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, pp. 51–58, sep 2011.

[5] Q. Jing, A. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: Perspectives and challenges," *Wireless Networks*, vol. 20, pp. 2481–2501, 11 2014.

[6] M. Wazid, A. K. Das, V. K, and A. Vasilakos, "Lam-ciot: Lightweight authentication mechanism in cloud-based iot environment," *Journal of Network and Computer Applications*, vol. 150, p. 102496, 11 2019.

[7] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 12 2015.

[8] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions," *Electronics*, vol. 9, no. 7, p. 1177, 2020.

[9] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, 2018.

[10] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection," *I. J. Network Security*, vol. 17, pp. 683–701, 2015.

[11] "Centro de investigación en tecnoloxías da información e as comunicacións de galicia." `https://www.citic-research.org/`. Último acceso: 2022-10-30.

[12] "Dad: Dataset for anomaly detection." `https://github.com/dad-repository/dad`. Último acceso: 2022-10-17.

[13] L. Vigoya, D. Fernandez, V. Carneiro, and F. J. Nóvoa, "Iot dataset validation using machine learning techniques for traffic anomaly detection," *Electronics*, vol. 10, no. 22, 2021.

[14] S. Rosset and A. Inger, "Kdd-cup 99: Knowledge discovery in a charitable organization's donor database," *SIGKDD Explor. Newsl.*, vol. 1, no. 2, p. 85–90, 2000.

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, July 2009.

[16] N. Paulauskas and J. Auskalnis, "Analysis of data pre-processing influence on intrusion detection using nsl-kdd dataset," in *2017 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, pp. 1–5, April 2017.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications*, CISDA'09, p. 53–58, IEEE Press, 2009.

[18] S. Revathi and A. Malathi, "A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection," 2013.

[19] S. S. Panwar and Y. Raiwani, "Performance analysis of nsl-kdd dataset using classification algorithms with different feature selection algorithms and supervised filter discretization," in *Intelligent Communication, Control and Devices*, pp. 497–511, Springer, 2020.

[20] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, pp. 314–323, 2019.

[21] "Darpa intrusion detection evaluation." `https://archive.ll.mit.edu/ideval/data/2000data.html`. Último acceso: 2022-10-17.

[22] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, p. 262–294, nov 2000.

[23] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, Nov 2015.

[24] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[25] N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, "Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques," in *MONAMI*, 2017.

[26] V. Timcenko and S. Gajin, "Machine learning based network anomaly detection for iot environments," 03 2018.

[27] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0305–0310, 2019.

[28] "The nims dataset." `https://projects.cs.dal.ca/projectx/Download.html`. Último acceso: 2022-10-17.

[29] "Malware capture facility project." `https://mcfp.weebly.com/`. Último acceso: 2020-01-30.

[30] "Unibs: Data sharing." `http://netweb.ing.unibs.it/~ntw/tools/traces/`. Último acceso: 2022-10-17.

[31] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, p. 357–374, may 2012.

[32] D. Zhao, I. Traoré, B. Sayed, W. Lu, S. Saad, A. A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals.," *Computers & Security*, vol. 39, pp. 2–16, 2013.

[33] E. Beigi, H. Jazi, N. Stakhanova, and A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," *2014 IEEE Conference on Communications and Network Security, CNS 2014*, pp. 247–255, 12 2014.

[34] E. Biglar Beigi, H. Hadian Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *2014 IEEE Conference on Communications and Network Security*, pp. 247–255, Oct 2014.

[35] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting p2p botnets through network behavior analysis and machine learning," in *2011 Ninth Annual International Conference on Privacy, Security and Trust*, pp. 174–180, July 2011.

[36] A. Lemay and J. M. Fernandez, "Providing SCADA network data sets for intrusion detection research," in *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*, (Austin, TX), USENIX Association, 2016.

[37] S. Duque, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/tcp data set," *CoRR*, vol. abs/1905.11757, 2019.

[38] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, 2018.

[39] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, p. 12–22, Jul 2018.

[40] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," *CoRR*, vol. abs/1802.09089, 2018.

[41] F. Abbasi, M. Naderan, and S. E. Alavi, "Anomaly detection in internet of things using feature selection and classification based on logistic regression and artificial neural network on n-baiot dataset," in *2021 5th International Conference on Internet of Things and Applications (IoT)*, pp. 1–7, 2021.

[42] M. Pahl and F. Aubet, "All eyes on you: Distributed multi-dimensional iot microservice anomaly detection," in *2018 14th International Conference on Network and Service Management (CNSM)*, pp. 72–80, 2018.

[43] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[44] I. Frazao, P. H. Abreu, T. Cruz, H. Araújo, and P. Simoes, "Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process," in *Critical Information Infrastructures Security* (E. Luiijf, I. Žutautaitė, and B. M. Hämmerli, eds.), (Cham), pp. 230–235, Springer International Publishing, 2019.

[45] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, "Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity," in *Proceedings of the 2019 ACM Symposium on SDN Research*, SOSR '19, (New York, NY, USA), p. 36–48, Association for Computing Machinery, 2019.

[46] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779 – 796, 2019.

[47] B. Susilo and R. F. Sari, "Intrusion detection in iot networks using deep learning algorithm," *Information*, vol. 11, no. 5, 2020.

[48] J. Alsamiri and K. Alsubhi, "Internet of things cyber attacks detection using machine learning," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 12, 2019.

[49] N. Moustafa, "New generations of internet of things datasets for cybersecurity applications based machine learning: Ton_iot datasets." `http://handle.unsw.edu.au/1959.4/resource/collection/resdatac_921/1`, 2019.

[50] M. Sarhan, S. Layeghy, N. Moustafa, M. Gallagher, and M. Portmann, "Feature extraction for machine learning-based intrusion detection in iot networks," 2021.

[51] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "Iot network intrusion dataset," 2019.

[52] A. Parmisano, S. Garcia, and M. J. Erquiaga, "Stratosphere laboratory. a labeled dataset with malicious and benign iot network traffic." `https://https://www.stratosphereips.org/datasets-iot23`. Último acceso: 2022-10-17.

[53] D. Thamaraiselvi and S. Mary, "Attack and anomaly detection in iot networks using machine learning," *International Journal of Computer Science and Mobile Computing*, vol. 9, pp. 95–103, 10 2020.

[54] L. Aversano, M. Bernardi, M. Cimitile, R. Pecori, and L. Veltri, "Effective anomaly detection using deep learning in iot systems," *Wireless Communications and Mobile Computing*, vol. 2021, 10 2021.

[55] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2019.

[56] I. Ullah and Q. H. Mahmoud, "A scheme for generating a dataset for anomalous activity detection in iot networks," in *Advances in Artificial Intelligence* (C. Goutte and X. Zhu, eds.), (Cham), pp. 508–520, Springer International Publishing, 2020.

[57] A. Guerra-Manzanares., J. Medina-Galindo., H. Bahsi., and S. Nõmm., "Medbiot: Generation of an iot botnet dataset in a medium-sized iot network," in *Proceedings of the 6th International Conference on Information Systems Security and Privacy - ICISSP,*, pp. 207–218, INSTICC, SciTePress, 2020.

[58] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "Mqttset, a new dataset for machine learning techniques on mqtt," *Sensors*, vol. 20, no. 22, 2020.

[59] Z. Liu, N. Thapa, A. Shaver, K. Roy, M. Siddula, X. Yuan, and A. Yu, "Using embedded feature selection and cnn for classification on ccd-inid-v1—a new iot dataset," *Sensors*, vol. 21, no. 14, 2021.

[60] M. MontazeriShatoori, L. Davidson, G. Kaur, and A. H. Lashkari, "Detection of doh tunnels using time-series classification of encrypted traffic," *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 63–70, 2020.

[61] S. Ghazanfar, F. Hussain, A. U. Rehman, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot-flock: An open-source framework for iot traffic generation," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pp. 1–6, 2020.

[62] W. Dhifallah, M. Tarhouni, T. Moulahi, and S. Zidi, "A novel realistic dataset for intrusion detection in iot based on machine learning," in *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, 2021.

[63] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3962–3977, 2022.

[64] M. Al-Hawawreh and E. Sitnikova, "Developing a security testbed for industrial internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5558–5573, 2021.

[65] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the development of a realistic multidimensional iot profiling dataset," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, pp. 1–11, 2022.

[66] B. Nikolova, G. T. Nikolov, and M. H. Todorov, "Curve fitting of sensors' characteristics," 2009.

[67] P. D. Schloss, B. Chaves, and L. P. Walker, "The use of the analysis of variance to assess the influence of mixing during composting," *Process Biochemistry*, vol. 35, no. 7, pp. 675–684, 2000.

[68] S. Yu, O. G. Clark, and J. J. Leonard, "A statistical method for the analysis of nonlinear temperature time series from compost," *Bioresource Technology*, vol. 99, no. 6, pp. 1886 – 1895, 2008.

[69] J. J. Flores, M. Graff, and H. Rodriguez, "Evolutive design of arma and ann models for time series forecasting," *Renewable Energy*, vol. 44, pp. 225–230, 2012.

[70] S. Bhandari, N. Bergmann, R. Jurdak, and B. Kusy, "Time series data analysis of wireless sensor network measurements of temperature," *Sensors*, vol. 17, no. 6, 2017.

[71] R. Cleveland, W. S. Cleveland, J. E. McRae, and I. J. Terpenning, "Stl: A seasonal-trend decomposition procedure based on loess (with discussion)," 1990.

[72] X. Huo, G. Cui, L. Ma, B. Tang, R. Tang, K. Shao, and X. Wang, "Urban land surface temperature prediction using parallel STL-Bi-LSTM neural network," *Journal of Applied Remote Sensing*, vol. 16, no. 3, p. 034529, 2022.

[73] S. Cristina, C. Cordeiro, S. Lavender, P. Costa Goela, J. Icely, and A. Newton, "Meris phytoplankton time series products from the sw iberian peninsula (sagres) using seasonal-trend decomposition based on loess," *Remote Sensing*, vol. 8, no. 6, 2016.

[74] G. Wang, X. Li, K. Zhao, Y. Li, and X. Sun, "Quantifying the spatio-temporal variations and impacts of factors on vegetation water use efficiency using stl decomposition and geodetector method," *Remote Sensing*, vol. 14, no. 23, 2022.

[75] "American power conversion corporation (apc)." `https://www.apc.com/es/es/`. Último acceso: 2023-01-12.

[76] I. Pedrosa, J. Juarros-Basterretxea, A. Robles-Fernández, J. Basteiro, and E. García-Cueto, "Pruebas de bondad de ajuste en distribuciones simétricas, ¿qué estadístico utilizar?," *Universitas Psychologica*, vol. 14, Oct. 2014.

[77] R. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.

[78] "Mqtt." `http://mqtt.org/`. Último acceso: 2022-10-30.

[79] V. Karagiannis, P. Chatzimisios, F. Vázquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Trans. IoT Cloud Comput.*, vol. 3, pp. 11–17, 01 2015.

[80] "Mosquitto." `https://mosquitto.org/`. Último acceso: 2020-01-30.

[81] "Tcpdump." `https://www.tcpdump.org/`. Último acceso: 2022-10-30.

[82] "Scapy." `https://scapy.net/`. Último acceso: 2022-10-30.

[83] "The constrained application protocol (coap)." `https://tools.ietf.org/html/rfc7252`. Último acceso: 2022-09-30.

[84] C. C. Sobin, "A survey on architecture, protocols and challenges in IoT," *Wireless Personal Communications*, vol. 112, pp. 1383–1429, Jan. 2020.

[85] "Rfc 7252 constrained application protocol." `https://coap.technology/`. Último acceso: 2022-10-17.

[86] A. Pardal Noya, "Xeración dun dataset baseado en tráfico coap nun entorno iot." http://hdl.handle.net/2183/26187, 2020. Último acceso: 2022-01-30.

[87] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. R. Shearer, and R. Wirth, "Crisp-dm 1.0: Step-by-step data mining guide," 2000.

[88] "tshark - dump and analyze network traffic." https://www.wireshark.org/docs/man-pages/tshark.html. Último acceso: 2022-08-21.

[89] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 2037–2064, Fourth-quarter 2014.

[90] A. C. Muller and A. C. Müller, *Introduction to machine learning with Python : a guide for data scientists*. Sebastopol, Calif.: O'Reilly Media, first edition. ed., 2016.

[91] M. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, "Support vector machine and random forest modeling for intrusion detection system (IDS)," *Journal of Intelligent Learning Systems and Applications*, vol. 06, no. 01, pp. 45–52, 2014.

[92] C. Chen and L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, 01 2004.

[93] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, p. 321–357, 2002.

[94] "Recursive feature elimination with cross-validation." https://scikit-learn.org/stable/auto_examples/feature_selection/plot_rfe_with_cross_validation.html#sphx-glr-auto-examples-feature-selection-plot-rfe-with-cross-validation-py. Último acceso: 2022-10-17.

[95] R. A. Irizarry, *Introduction to Data Science*. Chapman and Hall/CRC, Nov. 2019.

[96] I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, p. 1502, 12 2016.

[97] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimizationb," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[98] P. Ghosh and R. Mitra, "Proposed ga-bfss and logistic regression based intrusion detection system," in *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, pp. 1–6, 2015.

[99] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119–128, 2012. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[100] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated, 2014.

[101] O. J. Mebawondu, O. D. Alowolodu, A. O. Adetunmbi, and J. O. Mebawondu, "Optimizing the classification of network intrusion detection using ensembles of decision trees algorithm," in *Communications in Computer and Information Science*, pp. 286–300, Springer International Publishing, 2021.

[102] Y. Hamid, M. Sugumaran, and V. Balasaraswathi, "IDS using machine learning - current state of art and future directions," *British Journal of Applied Science & Technology*, vol. 15, no. 3, pp. 1–22, 2016.