Escuela Politécnica Superior

Sistema de detección de anomalías para Infraestructuras IoT

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autor:
Lucía Arnau Muñoz

Tutor/es:
José Vicente Berná Martínez

Junio 2023

Universitat d'Alacant
Universidad de Alicante

Resumen

En el mundo en el que vivimos actualmente, la necesidad de mantener seguros tanto los sistemas que utilizamos, como la información que transmitimos se ha convertido en una de las prioridades imprescindibles para el correcto funcionamiento de la sociedad. Cada día aparecen nuevas vulnerabilidades que pueden ser explotadas, por lo que la preocupación en cuanto a las medidas de seguridad aplicadas en nuestros sistemas nunca cesa.

El crecimiento de esta necesidad, ha desencadenado que las propias empresas, entidades investigadoras, e incluso personas por su propia cuenta, desarrollen cada día nuevas técnicas, herramientas y métodos con los que aplicar seguridad a sus ámbitos de trabajo.

En el caso de los entornos IoT, han resultado un avance en nuestra vida cotidiana con las facilidades que nos llegan a ofrecer sus dispositivos, sin embargo, su utilización también supone desafíos para la seguridad de la información con la que trabajan. Algunas de las herramientas más utilizadas para mantener seguros estos entornos, son los sistemas de detección de anomalías, con lo que se analizan e identifican patrones inusuales o sospechosos en los datos enviados por los dispositivos IoT.

Por ello, con este proyecto, se propone un sistema de detección de anomalías, centrado en el control de infraestructuras de entornos IoT, de manera que se pueda controlar el correcto funcionamiento de las mismas, permitiendo la toma de acciones casi inmediata tras un aviso de funcionamiento anómalo, o simplemente para su análisis y monitorización diario.

Motivación, justificación y objetivo general

Cada día, la seguridad toma más importancia dentro de cualquier servicio que se ofrezca, y debe ser tomada de forma innata en cualquier sistema que se implemente. Sin embargo, hay entornos que aún están muy lejos de poder tener una seguridad a la altura de las necesidades que se crean con la evolución de las nuevas tecnologías. Una de estas áreas son las de Internet de las Cosas, cuyo objetivo es el de conectar dispositivos entre sí y obtener información mediante su constante emisión de datos.

Dichos datos que proporcionan estos entornos IoT son muy útiles para el desarrollo de nuevas soluciones, sin embargo, estos servicios suelen ser ad-hoc con las infraestructuras y los dispositivos y por tanto es muy costoso tener en cuenta consideraciones de seguridad generales. Sabiendo la cantidad de datos que se pueden obtener de estos sistemas de sensorización, una buena forma de aprovecharlos para la seguridad sería idear sistemas de control que sean independientes del servicio y de la tipología de dispositivos. Sistemas que fuesen capaces de aprender de sí mismos para mejorar la seguridad, aprovechando las características intrínsecas de la información.

Siendo conscientes de todo esto, la motivación por la cual nació la idea de este proyecto fue para implementar una mejora en la seguridad del proyecto en el que trabajo actualmente, Smart University, pero más enfocado para las Smart Cities en general. Desde el inicio se planteaba la idea de elaborar un algoritmo de Machine Learning con el que poder controlar ciertos parámetros entrantes del proceso de sensorización que tienen implementado, de forma que se pudieran detectar anomalías o problemas de seguridad a nivel de infraestructuras.

El objetivo de este proyecto pretende ser una solución útil para el funcionamiento de las Smart Cities, haciendo un claro énfasis en el ámbito de la seguridad, debido principalmente a que es un apartado de las implementaciones de IoT que es muy difícil tener en cuenta, puesto que se sale de la línea de constante evolución que viven las nuevas tecnologías pertenecientes a Internet de las Cosas y debido a que no existen estándares ampliamente extendidos o su número es muy alto.

Además, el desarrollo de este algoritmo para la detección de anomalías encaja con las tecnologías que se utilizan cada día más, puesto que actualmente muchos procesos utilizan técnicas de Machine Learning durante sus procesos de producción, detección, predicción, etc. Es un método de trabajo que está normalizado y en auge, de forma que seguir ampliando este campo es una ventaja para el perfeccionamiento en el uso de técnicas con redes neuronales.

Uno de los motivos que más me ayudó a decidirme por realizar este proyecto fue el hecho de que los temas relacionados con Machine Learning e Inteligencia Artificial siempre me han llamado mucho la atención, y pude aprender y disfrutar más de su uso durante el grado, puesto que en la mención que cursé durante el grado tenía varias asignaturas relacionadas tanto con AI y ML, además de otras relacionadas con la sensorización y entornos IoT.

Agradecimientos

En primer lugar, a mi tutor, José Vicente Berná Martínez, por su ayuda incondicional durante el desarrollo del proyecto. Sin sus consejos, sabiduría y paciencia, este proyecto no habría sido igual, por lo que siempre tendrá mi gratitud, respeto y admiración. De la misma manera, agradecer a los profesores que me han acompañado durante mi etapa en esta nueva universidad, por todos los conocimientos que me han transmitido, junto con el tiempo y la atención que han dedicado en cada lección o práctica que nos presentaban.

En segundo lugar, a mi familia, mi madre Juani, mi padre Juan y mi hermana Victoria. A mis compañeros y amigos, que han estado a lo largo del trayecto apoyándome en todo momento y animándome a seguir adelante.

Desarrollar este proyecto marca el fin de una etapa, pero también da inicio a muchas otras, con las que seguir aprendiendo y mejorando a cada paso.

A todos ellos, mil gracias por compartir esta experiencia conmigo.

Citas

El éxito no es la clave de la felicidad. La felicidad es la clave del éxito. Si amas lo que haces, tendrás éxito.

Albert Schweitzer

La victoria no siempre será tuya, pero la constancia sí.

Napoleon Hill

Si algo se vuelve demasiado complicado, se atasca o no te convence: reinicia.

José Vicente Berná

No confío en la gente que no se ríe.

Audrey Hepburn

Índice de contenidos

Índice de figuras

Índice de códigos

Índice de tablas

1. Introducción

Con la constante evolución de las nuevas tecnologías, la seguridad ha pasado a ser un factor muy importante para todo tipo de empresas y organizaciones, a la hora de desarrollar un producto o servicio. Cuando no se tienen en cuenta, las consecuencias de no aplicar seguridad a los entornos de trabajo, o a cualquier otro, se pueden sufrir situaciones catastróficas como pérdidas de datos, secuestro y extorsión, mal funcionamiento de los sistemas o una denegación de nuestro servicio, resultando en posibles impactos tanto económicos como humanos, dependiendo del tipo de error que se produzca.

Por esta razón, es necesario desarrollar soluciones capaces de garantizar los aspectos de seguridad fundamentales de la información: Disponibilidad, Integridad, Confidencialidad, Autenticidad y Trazabilidad.

Durante el desarrollo de este proyecto, vamos a centrarnos en ciertos aspectos de la seguridad en las infraestructuras, puesto que no dejan de ser sistemas esenciales en nuestra vida, pero por esa misma razón, algunas de ellas pueden resultar en pérdidas críticas cuando no funcionan como corresponde, como pueden ser los sistemas que proveen energía o redes de comunicaciones, los cuales son más propensos a sufrir tanto ataques físicos como "ciberataques".

Algunas de las medidas que se deben tener en cuenta para securizar las infraestructuras son:

1. Evaluación de riesgos: Antes de comenzar a trabajar en las infraestructuras que se desean desarrollar, es necesario realizar un análisis exhaustivo de las amenazas y vulnerabilidades que pueden surgir después del despliegue, de forma que se puedan tomar las medidas de seguridad adecuadas.
2. Controles de acceso: A día de hoy, esta es una de las medidas más generalizadas en cualquier tipo de infraestructura, ya sean edificios comunes o lugares de trabajo más críticos, como salas de servidores. Por ello, se utiliza el control de acceso de manera física y lógica, teniendo un constante seguimiento de quienes acceden a dónde en todo momento.
3. Detección de intrusos: Este va a ser uno de los aspectos en los que nos centraremos durante el desarrollo del proyecto. Al implementar una infraestructura, muchas veces no se tiene en cuenta este tipo de situaciones, en las que alguien accede a un lugar que no le corresponde, etc. De esta forma, es imprescindible que las infraestructuras cuenten con alertas sobre cualquier actividad inusual o anómala, tanto de manera física como lógica.

4. Ciberseguridad: Cómo bien se ha estudiado a lo largo del máster, al igual que era indispensable contar con un sistema de detección de intrusos, también lo es el contar con medidas que protejan la información con la que trabajamos, como cifrar dicha información, usar protocolos de red seguros durante el envío de paquetes en red, para posteriormente monitorizar su trayectoria, o la autenticación de los usuarios, de forma que cada persona acceda con su determinado rol a los permisos correspondientes.
5. Copias de seguridad: Cualquier infraestructura que se precie debe contar con un sistema de respaldo de los datos con los que trabaja, puesto que la pérdida de la información durante situaciones inesperadas puede resultar en situaciones y consecuencias críticas.
6. Formación del personal: Una de las primeras lecciones que nos enseñan durante el máster es que el elemento más inseguro e impredecible de cualquier sistema es el ser humano. Es el eslabón al que primero van a atacar porque saben que puede ser la manera más sencilla de obtener información o acceso a determinados recursos. Sabiendo esto, es fundamental que el personal asignado en las áreas más críticas de la infraestructura tenga conocimientos sobre las posibles amenazas que pueden sufrir, además de las medidas de seguridad que deben aplicar, por mínimas que sean.

Una vez hemos resumido los aspectos principales de la seguridad en la infraestructura, vamos a centrarnos en el tercer punto comentado anteriormente, la detección de anomalías.

Por otra parte, al igual que las infraestructuras pueden sufrir determinados ataques y se desarrollan herramientas para su detección, como es el caso de los IDS, también pueden simplemente tener un mal funcionamiento, el cual supone un resultado incluso más grave que los ciberataques o ataques físicos, puesto que nos alejamos del problema de perder información, y pasamos a que un fallo puede afectar a muchos más dependientes del mismo.

En el caso de los IDS, se centran en la detección de ataques en red, en base al análisis de parámetros determinados, como pueden ser las cabeceras de un paquete. En este caso, el contenido de la información recibida se analiza y se compara con una base de datos con ataques conocidos, de forma que se evalúa si su comportamiento entra dentro de la normalidad esperada, o, por el contrario, es anómalo.

Este mismo procedimiento es aplicable a las infraestructuras IoT, en los que se pueden realizar detecciones de anomalías según su comportamiento habitual, destacando los momentos en los que se reciben parámetros fuera de lo común, o incluso se detecta la propia falta de información, lo cual también es motivo de alerta.

Por ello, se va a desarrollar un Sistema de Detección de Anomalías en infraestructuras IoT, de manera que se puedan detectar estos comportamientos anómalos, fallos en el sistema, etc. Pero en nuestro caso vamos a abordar la generación de un detector de anomalías de amplio espectro, en vez de un identificador de incidentes concretos. Esto se debe a que en el mundo IoT a menudo se utilizan infraestructuras open source o sobre las que no tenemos el control, mientras que el sistema se sensorización puede ser nuestro, parte del medio o software de transmisión puede no serlo. Y por ello es muy interesante poder detectar anomalías en la infraestructura IoT que posteriormente podemos identificar con un incidente.

2. Planificación

El desarrollo del proyecto se llevará a cabo en 5 meses, desde enero hasta mayo de 2023. Durante este proceso, tanto la evolución como los resultados obtenidos serán supervisados por miembros del proyecto de Smart University, perteneciente a la Universidad de Alicante. [1]

- En el mes de enero, se lleva a cabo un estudio de la temática del proyecto, las tecnologías a utilizar, además de la solución que se quiere obtener con este proyecto, de manera que cumpla con las necesidades presentadas por el grupo de Smart University.
- En el mes de febrero, se definieron algunos matices más para el proyecto, además de investigar y redactar el estado del arte, objetivos propuestos y contexto general de la solución.
- En los meses de marzo, abril y parte de mayo, se desarrolla el código necesario para la obtención de datos, su procesamiento y finalmente detección de las anomalías.
- Los días restantes de mayo estuvieron destinados a completar la memoria.

DIAGRAMA DE GANTT
Planificación

MES	ENERO	FEBRERO	MARZO	ABRIL	MAYO
SEMANA 1-2	Estudio de temáticas y tecnologías	Definición de matices del proyecto	Desarrollo de código	Desarrollo de código	Desarrollo de código
SEMANA 3-4	Investigación y estado del arte	Investigación y redacción de objetivos	Desarrollo de código	Desarrollo de código	Desarrollo de código
SEMANA 5-8			Desarrollo de código	Desarrollo de código	Desarrollo de código
SEMANA 9-12				Pruebas y depuración	Pruebas y depuración
SEMANA 13-16				Implementación	Implementación
SEMANA 17-18					Completar memoria

Tabla 1. Planificación temporal TFG

3. Estado del Arte

Durante este apartado se van a detallar ciertos estudios o implementaciones relacionadas con el tema del proyecto, además de una explicación más exhaustiva de los apartados más relevantes para su desarrollo.

3.1. Introducción a las Smart Cities

Según la definición de la Fundación Telefónica[2], una ciudad inteligente es una ciudad que utiliza las Tecnologías de la Información y la Comunicación para hacer que, tanto su infraestructura crítica, como sus componentes y servicios públicos ofrecidos, sean más interactivos, eficientes y los ciudadanos puedan ser más conscientes de ellos.

Una gestión más eficiente de los servicios y la infraestructura conlleva una reducción del gasto público, además de una mejoría en la prestación de los mismos, tanto en términos de toma de decisiones del proveedor de servicios como en términos de información a los ciudadanos sobre ellos. [3]

Algunos de los servicios que se pueden dar dentro de las funcionalidades de las Smart Cities es el uso de algoritmos de Machine Learning para diferentes controles de información, como puede ser la gestión de la movilidad urbana, el tráfico en una ciudad. Por ejemplo, esta gestión también puede ayudarnos a identificar las áreas con más contaminación del aire por alta concentración de vehículos, detectando un incremento de enfermedades respiratorias en la ciudad.

Muchos de estos servicios están interconectados y pueden utilizar los mismos datos para sacar diferentes conclusiones, por lo que compartir información entre diferentes servicios será clave para el éxito de las ciudades así de inteligentes. (Fondo Telefónico, 2017)

Algo que caracteriza las Smart Cities es precisamente su alta sensorización o captación de información a través de dispositivos o agentes que son capaces de percibir el entorno, y proporcionar información útil sobre él. Debemos tener en cuenta que no solo se tratan de dispositivos físicos como sensores de temperatura, sensores de llenado de contenedores de basura o sensores de humedad para conocer el estado del césped en un parque. También se puede extraer una gran cantidad de datos a través de otras infraestructuras, como estimar las personas que circulan por una zona en base a las conexiones a la red WIFI pública o conocer el estado del tráfico en base al movimiento de los vehículos debido que cada usuario lleva encima un teléfono móvil con acelerómetro. Esta información se extrae de otros subsistemas más complejos a graves de

procesadores o agentes. Pero lo realmente importante es que en este nuevo escenario de ciudades inteligentes entran en acción lo que conocemos como IoT, o todas aquellas cosas que están conectadas a Internet y están generando datos. Y donde entra una nueva tecnología, también aparecen nuevos problemas de seguridad [4]. Problemas que pueden ser muy graves y que pueden dar al traste con los sistemas de las ciudades.

3.2. Seguridad en IoT

En los últimos años se ha asistido al rápido desarrollo y despliegue de aplicaciones de Internet de las Cosas (IoT) en diversos ámbitos de aplicación. El IoT se perfila como la tercera ola en la evolución de Internet. La ola de La ola de Internet de los 90 conectó a 1.200 millones de abonados, mientras que la ola móvil de los 2000 conectó a otros 2.400 millones. En la actualidad, se espera que la IoT esté formada por más de 84.000 millones de dispositivos conectados que generen 186 zettabytes de datos en 2025, según IDC [5]. IoT incluye grandes tipos de redes, como las distribuidas, ubicuas o vehiculares, que han conquistado el mundo de la informática durante una década. IoT está creciendo rápidamente en varias verticales de la industria junto con el aumento del número de dispositivos interconectados y la diversificación de aplicaciones IoT. A pesar de ello, las tecnologías IoT aún no han alcanzado la madurez y quedan muchos retos por superar. El Internet de las Cosas combina lo real y lo virtual en cualquier momento y lugar, fascinando la atención tanto del constructor como del hacker. Necesariamente, dejar los dispositivos sin interferencia humana durante un largo periodo podría dar lugar a robos, malfuncionamientos, manipulación e incluso accidentes [6]. Es por ello que es necesario diseñar sistemas para protegerlos teniendo en cuenta que es un nuevo paradigma en sí mismo.

3.2.1. Vulnerabilidades en IoT

Dentro de IoT podemos encontrar un gran número de vulnerabilidades conocidas [7], algunas dependientes de las infraestructuras y otras más generales, como son:

- Contraseñas débiles o fáciles de adivinar: son contraseñas que se pueden obtener fácilmente otorgando acceso no autorizado al dispositivo en cuestión.
- Servicios de red inseguros. Muchos dispositivos actualmente en funcionamiento no cuentan con las medidas de seguridad necesarias para proteger sus servicios de red, dejándolos vulnerables a ataques.

- Interfaces de ecosistemas inseguros. Esta vulnerabilidad ocurre cuando el problema no está en el dispositivo, sino en las aplicaciones que ejecuta.
- Ausencia de mecanismos de actualización seguros. A menudo, los desarrolladores no lanzan actualizaciones de parches con regularidad, lo que deja el dispositivo y su software desprotegidos de la capacidad de explotar agujeros de seguridad conocidos.
- Uso de componentes inseguros u obsoletos: usar recursos de fuentes confiables cuando se configura el dispositivo, para evitar tantos agujeros de seguridad como sea posible.
- Protección de la privacidad insuficiente. Muchos dispositivos no están configurados inicialmente para proteger la información privada y confidencial con la que siempre trabajan.
- Almacenamiento y transmisión de datos no seguros. Como en el caso anterior, por regla general, los dispositivos no cuentan con herramientas de seguridad de la información integradas, como algoritmos de encriptación.
- Sin gestión de dispositivos. Es muy común en muchas grandes empresas que desconozcan la cantidad de dispositivos que utilizan sus servicios, como conectarse a su servicio de red o el tipo de información que fluye a través de ellos.
- Valores predeterminados inseguros: si bien esta es una utilidad reconocible para muchos usuarios, todavía representa una línea directa de ataque porque los valores predeterminados proporcionados por el fabricante son básicos e inseguros, completos en la mayoría de los casos.
- Falta de seguridad física: dado que los dispositivos son intrínsecamente seguros, no se debe olvidar que el acceso físico es el factor más común y más fácil, por lo que el acceso debe administrarse, asegurarse y controlarse. establecer.

Estos ejemplos de vulnerabilidades abren las puertas a otro número ingente de posibilidades de atentar contra nuestro sistema IoT

3.2.2. Ataques más comunes en entornos IoT

Aquí se recogen algunos de los ataques más comunes detectados [8]:

- Control de entidades: También conocido como Sybil, está basado en la manipulación de entidades falsas con tal de comprometer el correcto funcionamiento del sistema. Aplicado en entornos IoT, el ataque pretende controlar el mayor número de nodos posibles, con tal de acceder a un recurso en particular, o a la propia red.
- Denegación de servicio: Se basa en saturar el servidor con una gran cantidad de peticiones, con tal de interrumpir el servicio.
- Espionaje: También llamado sniffing o snooping, con el que se quiere obtener información de forma ilícita.
- Filtración: Este tipo de ataques se aprovecha de las comunicaciones que se realizan de forma inalámbrica, detectando el tipo de eventos IoT que se producen en los campos de mensajes que se envían en texto claro.
- Modificación de la información: Comúnmente conocido como Man In The Middle, se produce cuando un nodo puede leer, modificar o añadir información en una comunicación de manera no lícita.
- Interferencia: O jamming, se utiliza para interrumpir las transmisiones radio de los dispositivos IoT a través del envío de señales falsas.

Estos ejemplos vienen a declarar la necesidad de de incorporar urgentemente sistemas de seguridad que no solo cubran las actuales, sino también las futuras incidencias que puedan darse.

Sin embargo, aunque en principio pueda parecer abrumador, podemos mirar hacia otros sistemas, las redes, para intentar inspirarnos en nuestra solución.

3.3. Seguridad en Red

Según Cisco, la seguridad de red es cualquier actividad diseñada para proteger el acceso, el uso y la integridad de la red y los datos corporativos. [9]

Si nos centramos en las redes Ethernet, el desarrollo de medidas de seguridad pertenecientes a la arquitectura de las redes Ethernet ha recibido poca atención por parte de la comunidad investigadora durante estos años, puesto que la mayoría de investigaciones existentes están

centradas, por ejemplo, en aportar soluciones criptográficas a las transmisiones de envío de paquetes. Otros muchos estudios se centran en mantener la red de Ethernet segura mediante productos de los proveedores, sin comprender realmente las propiedades relacionadas con la seguridad de la tecnología Ethernet. [10]

Actualmente existe el protocolo EtherNet/IP, una adaptación de un protocolo a Ethernet, en la que su seguridad engloba tanto la del nivel de aplicación (seguridad CIP), como la de transporte (EtherNet/IP), con mecanismos como la autenticación de los equipos, de los mensajes, además del cifrado de estos. [11]

Utilizando el concepto de redes Ethernet tradicionales, nosotros podemos entender que dentro de nuestros entornos IoT, LoRa serían los cables que interconectan los dispositivos en una red Ethernet, mientras que el protocolo de LoRaWAN, es la comunicación de estos dispositivos a nivel de direcciones MAC e IP. [12]

Teniendo esta aclaración en cuenta, vamos a explicar más en profundidad la red utilizada para este proyecto, LoRaWAN.

3.3.1. Redes LoRaWAN

Como se ha comentado anteriormente, LoRa es la capa física de la red LPWAN, conocida como LoRaWAN. La red LoRa consta de los siguientes tipos de componentes [13]:

- Nodos: Son dispositivos finales que envían y reciben información hacia el gateway.
- Gateways: Son los dispositivos que reciben la información procedente de los nodos y la reenvían a los servidores de red.
- Network Server: Es el software encargado de controlar la red y las comunicaciones entre los nodos y los gateways, evitando situaciones como recibir paquetes duplicados.

Por otra parte, hay otro componente perteneciente a las arquitecturas LoRa, sin ser un elemento principal de la arquitectura:

- Application Server: Se trata de un software externo a la red de LoRaWAN, con el que se trabajan los datos obtenidos durante el envío por LoRa. Un ejemplo de ello es el protocolo MQTT, o el uso de una API REST.

En cuanto al protocolo, LoRaWAN (Long Range) es un protocolo de radiofrecuencia de largo alcance y baja potencia, permitiendo a los dispositivos de bajo consumo el envío de paquetes en redes tanto

locales, regionales, nacionales o globales. En la red LoRaWAN, los dispositivos se dividen en clases según las funcionalidades que soportan, en concreto: la clase A, B y C.

- Los dispositivos de clase A tienen un menor consumo de energía y se utilizan generalmente en dispositivos enfocados a funcionar mediante una batería. La planificación de las transmisiones corre a cargo del propio dispositivo final. La recepción, que sólo está permitida después de una transmisión completada correctamente, está formada por dos ventanas separadas de recepción.
- Los dispositivos de clase B, los cuales consumen más energía con respecto a los de clase A, y admiten una trasmisión de datos periódica y continua. Esta transmisión se realiza mediante el envío de beacons por parte del gateway, que se sincroniza con el nodo, con el fin de planificar el tiempo en el que el dispositivo debe abrir la ventana de recepción.
- Finalmente están los dispositivos de clase C, los cuales están en modo de recepción permanente, que sólo se interrumpe cuando se produce una transmisión. Esta clase de dispositivo presenta la mejor latencia de conexión entre los nodos y los gateways, a cambio de un mayor consumo.

En la siguiente figura se muestra un diagrama de la arquitectura [14]:

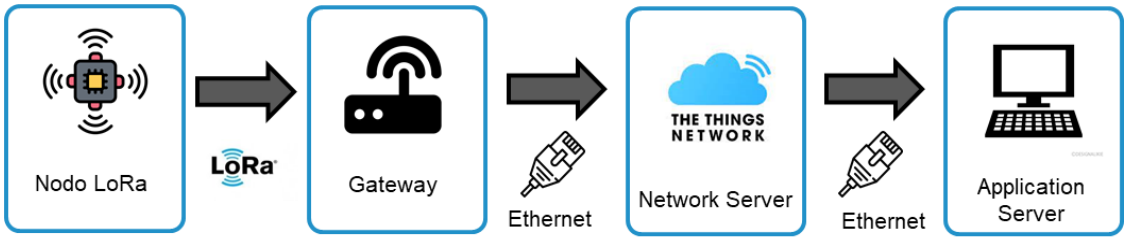


Figura 1. Arquitectura LoRa

3.3.2. Ataques y vulnerabilidades más frecuentes

- Ataque de Eavesdropping: Este ataque afecta principalmente a la información que viaja entre el gateway y el nodo final, de forma que se escuchan (sniffing) y graban (capturing) dichos datos de la transmisión. Durante el ataque, la privacidad se ve comprometida aunque los datos permanezcan intactos.
- Ataque por Replay: Este tipo de ataques se suele llevar a cabo a la vez que se lanza un ataque DoS o de spoofing. Cuando se lanza este ataque, se suplanta la identidad de uno de los nodos, de manera que el servidor sigue recibiendo información de un nodo que él cree

que está autenticado en su red. Al recibir un mensaje desde ese nodo, se produce la denegación del servicio en el servidor.

- Ataque por Jamming: El objetivo de este ataque es interrumpir las transmisiones de radio, enviando una fuerte señal de radio en las áreas cercanas a donde se encuentran los dispositivos, ya sean nodos o gateways.
- Ataque por Denegación de Servicio (DoS): Al realizar este ataque, se satura el servicio víctima durante un determinado periodo de tiempo, de forma que no se encuentra disponible para procesar las peticiones que le llegan. En la actualidad, la mayoría de infraestructuras tienen la capacidad de manejar una gran cantidad de datos, por lo que este ataque ha quedado más en segundo plano, dejando paso a los ataques DDoS (Denegación de Servicios Distribuidos). A diferencia del ataque DoS, se utilizan varias fuentes al mismo tiempo, en lugar de uno único que lance el ataque.

3.4. Redes Neuronales aplicadas a la Ciberseguridad

3.4.1. Qué es una red neuronal

Antes de empezar a hablar de las posibles aplicaciones que tienen las redes neuronales en el ámbito de la ciberseguridad, es necesario definir qué es exactamente una red neuronal.

En primer lugar, una red neuronal se puede definir como un sistema que permite establecer una relación entre entradas y salidas inspiradas en el sistema nervioso y diferenciándose de la computación tradicional, ya que estos no utilizan una algoritmia secuencial. Las redes neuronales artificiales se comportan como un cerebro humano, en donde se procesa la información en paralelo, con la posibilidad de aprender y generalizar situaciones no incluidas en procesos de entrenamiento. [15]

Dentro del uso de las redes neuronales, podemos diferenciar varios tipos de aprendizaje [16]:

- Aprendizaje supervisado: en este tipo de aprendizaje, el diseñador de la red neuronal supervisará y controlará el entrenamiento de la red neuronal, para determinar que la respuesta de la red sea una específica dependiendo de la entrada.
- Aprendizaje por corrección de error: El objetivo de este aprendizaje es que no haya una diferencia significativa entre la salida obtenida y la salida deseada. Para lograrlo, se comparan ambas salidas y se ajustan los pesos de las conexiones de la red, en función de las diferencias con los valores deseados y obtenidos.

- Aprendizaje por refuerzo: En este tipo de aprendizajes supervisados, no se proporciona un ejemplo completo de una salida esperada; en cambio, el diseñador de la red indica mediante una señal de refuerzo (éxito o fracaso) si la salida que se obtuvo de la red se acerca a la salida deseada.
- Aprendizaje Estocástico: Durante este entrenamiento, se realizan cambios aleatorios en los pesos de la red durante el entrenamiento y se comparan las salidas obtenidas con las salidas esperadas.
- Aprendizaje no-supervisado: Este tipo de aprendizaje, la red neuronal aprende por sí sola las características, regularidades, correlaciones y categorías con los datos de entrada, sin necesidad de supervisión o comparación de salidas externas.

Una vez se ha definido el concepto de red neuronal, algunas de las aplicaciones más conocidas en la actualidad dentro de la ciberseguridad son [17]:

- Thead hunting
- Gestión de vulnerabilidades (Vulnerability Management)
- Data centers
- Seguridad en las redes
- Identificación segura de usuarios (Securing Authentication)
- Privacidad de la información y compliance
- Bloqueo de bots a partir de su comportamiento

3.4.2. Ejemplos de uso de redes neuronales

Tras definir lo que son las redes neuronales, vamos a explicar algunos de los ejemplos reales pertenecientes al ámbito de la ciberseguridad en los que se aplican, utilizando algoritmos de Machine Learning, redes neuronales, etc.

3.4.2.1. Sistema de Detección/Prevención de Intrusiones – IDS/IPS

Para poder entender la funcionalidad del algoritmo a desarrollar, un buen ejemplo podrían ser los Sistemas de Detección de Intrusiones, o IDS, simplemente como una explicación a grandes rasgos del objetivo principal.

La detección de intrusiones es el proceso de monitorizar redes de ordenadores y sistemas en busca de vulneraciones de políticas de seguridad [18]. El análisis del tráfico de la red se realiza mediante

la utilización de mecanismos dE identificación de patrones y métodos estadísticos. Los sistemas de detección de intrusiones están compuestos por tres elementos funcionales básicos: [19]

- Una fuente de información que proporciona eventos de sistema.
- Un motor de análisis que busca evidencias de intrusiones.
- Un mecanismo de respuesta que actúa según los resultados del motor de análisis.

Existen dos campos de clasificación para los sistemas de detección de intrusos [20]:

- Sistemas de vigilancia
- Como se hace esa vigilancia

Dentro de esos dos campos, encontramos diferentes tipos de IDS [21]:

- Sistema de detección de intrusos en la red (NIDS): Se encarga de monitorear todo el tráfico de un segmento estratégico de la red o de un dispositivo, analizando la red y la actividad de los protocolos para encontrar actividades maliciosas o sospechosas y comparando los datos del tráfico con una biblioteca de ataques conocida.
- Sistema de detección de intrusos en host (HIDS): Monitorea las características de un host y los eventos que ocurren en él para encontrar actividades maliciosas o sospechosas. Tanto el tráfico malicioso que entra en el host como el que genera el propio host pueden ser detectados por los HIDS, que un sistema de detección basado en red no podría detectar.
- Sistema de detección de intrusos basados en firmas (SIDS): Es un sistema de detección basado en firmas que alerta si detecta una coincidencia analizando los paquetes de datos que ingresan a la red y comparándolos con firmas de amenazas conocidas almacenadas en su base de datos.
- Sistema de detección de intrusos basado en anomalías (SIDA): Observa el tráfico de la red para detectar comportamientos o actividades inusuales, es decir, que no coinciden con las firmas, pero que se consideran extraños para el funcionamiento normal de la red en relación al ancho de banda, protocolos, puertos y otros dispositivos conectados. Este sistema basado en anomalías detecta nuevas amenazas desconocidas mediante el uso del aprendizaje automático.

3.4.2.2. Algoritmos de detección de SPAM

Spam es el término que se emplea comúnmente para designar el correo electrónico no solicitado que se envía a través de Internet. Al igual que en el caso de los IDS, es común el uso de modelos de Machine Learning para su detección y filtrado.[22]

Existen varios tipos de modelos basados en su contenido, que aplican distintas técnicas de aprendizaje automático supervisado, algunos ejemplos de ellos son:

- Aproximaciones colaborativas: En este tipo de técnicas generalmente no se considera el contenido del mensaje, y la clasificación de los correos se lleva a cabo mediante la colaboración de grupos de usuarios que comparten información sobre los mensajes spam. Tras analizar los criterios del mensaje, se elaboran listas negras, denegando la entrada de correos que contengan contenido similar al encontrado.
- Modelos Basados en contenido: Al contrario que en las aproximaciones colaborativas, las técnicas basadas en contenido emplean características extraídas de la cabecera o del cuerpo del mensaje para realizar la clasificación del correo. Para este tipo de modelos, se suelen utilizar algoritmos como Naïve y Flexible Bayes, utilizado principalmente para la clasificación de textos, aunque también se encuentran ejemplos empleando SVM (Support Vector Machines), empleado en tareas de clasificación y regresión, e incluso algoritmos de decisión como los Decision Trees.

3.4.2.3. Algoritmos de detección de URLs maliciosas

Las URL se utilizan para referirse a la ubicación de recursos de internet únicos. En la estructura de una URL, se incluye primero el esquema o protocolo, seguido del nombre de dominio y posibles subdominios con la extensión del dominio o Top Level Domain (TLD), terminando con la dirección a un determinado recurso o ruta específica.[23]

Debido a la cantidad de características que se suele extraer para este tipo de detecciones, los mejores algoritmos para poder modelar las relaciones entre las entradas y la predicción final, son los algoritmos supervisados, basados en árboles de decisión:

- Árboles de decisión: Dado un conjunto de datos se fabrican diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la resolución de un problema. [24]

- Bosques aleatorios: Son una combinación de árboles predictores tal que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos. Es una modificación sustancial de bagging que construye una larga colección de árboles no correlacionados y luego los promedia. [25]

3.4.2.4. Algoritmos de detección de transacciones bancarias fraudulentas

El fraude se refiere al mal uso de métodos de pago robados (como tarjetas de crédito) o información personal para transacciones financieras no autorizadas. A medida que aumenta el acceso de la población a los servicios financieros, también aumenta la cantidad y diversidad de formas de fraude. En la actualidad, gracias al uso de algoritmos de Machine Learning previamente entrenados, es posible determinar si una transacción corresponde a un fraude o no. [26]

Algunos de los algoritmos o métodos más utilizados para ellos son:

- One-Class Support Vector Machines: Es una derivación del método de clasificación binaria Support Vector Machines (SVM), simplemente que en este caso se separan los datos de una sola clase del origen. De esta manera, al entrenar un modelo de este tipo e introducir datos anómalos, se espera que su clasificación sea distinta a la clase de los datos válidos.
- Autoencoders: Consiste una red neuronal artificial entrenada para entregar en su salida el mismo dato que se le introduce.
- Isolation Forest: Consiste en un conjunto de árboles binarios construidos a partir de particiones aleatorias de un conjunto de datos. El objetivo es que, por medio de estas particiones aleatorias en cada atributo del conjunto de datos, se consiga aislar cada dato en cada isolation tree.

3.5. The Things Network

The Things Network es una plataforma colaborativa a nivel global de Internet de las Cosas, la cual trabaja creando redes, dispositivos y soluciones utilizando el protocolo LoRaWAN, que es una red abierta, descentralizada y de colaboración abierta. [27]

La plataforma de The Things Network, además de proporcionar una plataforma con la que trabajar fácilmente en las comunicaciones por LoRa, también ha desarrollado la parte de servidor, construyendo todo el “backend” de la red, dando soporte a los gateways distribuidos por todo el

mundo. Este servicio interno es el encargado de lidiar con las duplicidades de mensajes, transmisión de mensajes uplink y downlink, gestión de integraciones con plataformas, etc.

Por otra parte, es una red interoperable con soporte de gran variedad de protocolos de comunicación para su integración, como HTTP y MQTT, además de una serie de APIs en distintos lenguajes con las cuales se puede construir una aplicación end-to-end, mediante la integración de nodos, gateways, server de TTN y una plataforma IoT.

3.5.1. Seguridad y privacidad

La plataforma de TTN, pese a ser una herramienta pública, mantiene segura la información al realizar un cifrado de extremo a extremo y permite el uso de diferentes claves de cifrado de 128 bits para cada end device.

Al trabajar en base al protocolo de LoRaWAN, se deben seguir sus medidas de seguridad, como la comprobación de la integridad de los mensajes AES de 128 bits, junto con el cifrado de carga útil.

4. Objetivos

El objetivo principal de este proyecto es el de diseñar e implementar un sistema de detección de anomalías basado en algoritmos de inteligencia artificial, que permita su implementación sobre cualquier tipo de infraestructura IoT y para cualquier tipo de anomalía. El carácter general de la propuesta pretende ser capaz de detectar anomalías, pero sin establecer cuál es la anomalía concreta que se está generando. Este sistema difiere de otras propuestas donde se trata de detectar cuál es el incidente que se está produciendo para así establecer las medidas acordes. Nuestro detector de anomalías busca detectar tráfico sospechoso para su descarte y posterior análisis.

La principal ventana de nuestra propuesta es su generalidad. Los sistemas IoT están caracterizados por un alto tráfico de paquetes con pequeñas cantidades de información. Una respuesta rápida del sistema requiere al menos que los paquetes potencialmente dañinos no entren en el flujo de trabajo, ya que podrían generar acciones no deseadas sobre los sistemas que sensorizan.

Por ello, con tal de ampliar y mejorar las soluciones disponibles para este tipo de monitorización y control de las infraestructuras, se aprovechan los diferentes métodos de aprendizaje automático existentes, de forma que se pueda crear un algoritmo lo más eficiente posible, con tal de implementarlo posteriormente en los procesos que trabajan con los datos, con tal de discernir entre datos válidos o normales, de los que se consideran una anomalía.

Por tanto, teniendo esta información en cuenta, los subobjetivos a cumplir serían los siguientes:

- Investigar y enumerar diferentes ejemplos de sistemas de detección de anomalías que puedan servirnos de inspiración en este trabajo.
- Investigar las diferentes técnicas de Machine Learning que se pueden aplicar en el desarrollo de un algoritmo de detección de anomalías.
- Diseñar un flujo de trabajo que permita, de forma aséptica a la infraestructura, la adquisición y procesamiento de datos para su posterior análisis.
- Implementar y entrenar el algoritmo mediante el uso de conjuntos de datos de prueba.
- Validar y optimizar el algoritmo mediante una ingesta de datos en tiempo real.
- Estudiar la aplicabilidad del algoritmo resultante para diferentes sectores relacionados con los entornos de IoT.

5. Metodología

Para el correcto desarrollo de cualquier proyecto es necesario establecer una metodología de trabajo que permita su consecución exitosa. Dado el carácter de nuestra propuesta y que en realidad requiere de una mayor carga de conceptualización frente a la implementación (idear flujos de trabajo, definir procesos de captación y adaptación de datos, búsqueda de algoritmos, pruebas y validación), nos decantamos por una metodología de desarrollo software clásica o metodología en Cascada, donde definiremos las siguientes fases:

- Realizar un estudio previo del estado del arte que permita conocer planteamientos cercanos al nuestro, alternativas y otros sistemas y algoritmos existentes.
- Análisis de los datos, desde un punto de vista independiente de las infraestructuras para la conceptualización de los procesos de adquisición y adaptación de datos.
- Definición de procesos para el entrenamiento y puesta en marcha de algoritmos IA.
- Implementación de procesos y codificación.
- Pruebas y validación sobre un escenario real a través de un caso de estudio, la implementación de un ADS y su puesta en marcha.

Para el control del proyecto utilizamos un software de gestión de proyectos, *Trello* [28], que permitirá definir las tareas, su planificación temporal, y facilitará la comunicación con el tutor para que pueda supervisar el progreso logrado. A través de esta herramienta se definirán varias columnas: *backlog*, *to do*, *doing*, *done*. Las actividades que se añadirán al backlog se irán categorizando en las fases indicadas anteriormente. Es decir, utilizaremos un Kanban para la gestión del proyecto, pero realizará seguimiento de la metodología en cascada. La metodología nos permitirá no pasar a actividades posteriores hasta que no hayan sido perfectamente definidas y validadas las actividades antecesoras. Kanban nos permitirá tener una vista visual de las actividades para un seguimiento más intuitivo. Cada actividad quedará definida por su fase, un título, una descripción y un color asociado a la fase que permita identificar fácilmente ésta. En general todas las actividades de una fase deberán estar acabadas antes de pasar a la siguiente, aunque de forma excepcional pueden existir tareas que requieran de revisiones y por tanto entren en la columna *doing* de nuevo.

Para el control del tiempo invertido en el desarrollo del proyecto utilizaremos una aplicación específica, Clockify [29], que nos permitirá hacer un seguimiento del tiempo invertido en cada actividad. El objetivo del seguimiento del tiempo es doble: por un lado, conocer con certeza el tiempo que se invierte en las actividades asegurando que es suficiente y que son tratadas con la

profundidad necesaria; controlar el tiempo para que no sea excesivo, ya que sin darnos cuenta podemos quedar anclados en una actividad eterna, como el estado del arte o el diseño y refinamiento de procesos de datos, y entonces el proyecto se vería gravemente afectado.

6. Diseño Conceptual – Modelo de ADS

El diseño y desarrollo de un sistema de detección de anomalías (ADS por sus siglas en inglés) requiere de un conjunto de procesos perfectamente estructurados y coreografiados. Estos procesos deben cubrir las fases de adquisición de datos, su procesamiento y preparación, el entrenamiento de los algoritmos de detección, y la puesta en producción del sistema. Es muy importante analizar todos los posibles pasos que intervienen en la solución, desde los datos pueden ser captados desde las infraestructuras hasta que se logra una catalogación como normal o anormal. Recordamos que el objetivo de esta propuesta no es la detección de incidentes concretos en sí, como una suplantación, una modificación de datos o un ataque de denegación de servicios. Lo que se busca es un sistema que permita, de forma general, detectar anomalías en el flujo de datos para que estos puedan ser descartados o destinados a un flujo secundario de análisis donde ahí sí, se podría detectar el incidente. Al estar orientado a infraestructuras IoT, se han de tener en cuenta que muchas de las anomalías pueden deberse a problemas con los dispositivos como por ejemplo que las baterías se agoten, qué elementos pantalla interfieran en las señales, movimiento involuntario o malicioso de los dispositivos, etc. Todas estas anomalías también quieren ser consideradas.

Es por ello que vamos a diseñar un modelo de detección de ADS que no solo considere la ejecución de la solución, es decir, la fase de puesta en producción, sino que también considere los procesos necesarios para la fase de construcción. Nuestra propuesta gira en torno a la utilización de algoritmos de Machine Learning de amplio espectro, es decir, capaces de catalogar en múltiples segmentos los datos analizados. Pero en la construcción de dichos algoritmos, la preparación y adecuación de los datos puede introducir inconscientemente errores si no se hace de forma sistemática y correcta. Es por ello que hemos incluido dichos procesos como parte del proceso de desarrollo de ADS.

De forma general, definimos el ADS como un sistema formado por dos procesos:

- CM - Creación del Modelo IA: este proceso incluye las acciones para la adquisición y preparación de datos y la generación de los modelos de entrenamiento y algoritmos para preparar la IA para la detección.
- D - Detección: este proceso es el responsable de utilizar los resultados del proceso anterior, los modelos entrenados, y someter al tráfico del sistema a la detección mediante ellos, generando la detección en sí mediante la catalogación del tráfico. El tráfico detectado posteriormente podrá ser tratado a través de alertas o acciones.

La Figura 2. Diagrama general del modelo de ADS muestra un diagrama de dichos procesos.

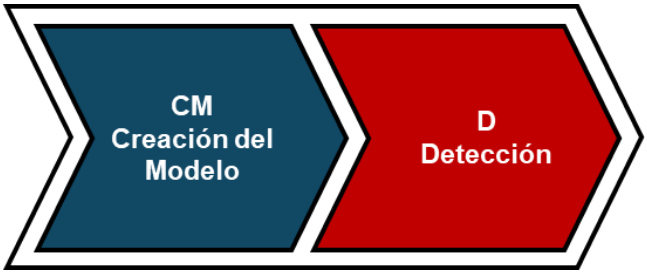


Figura 2. Diagrama general del modelo de ADS

Cada uno de estos procesos está formado por a su vez por subprocesos, que nuevamente se dividen en etapas, que finalmente se dividen en actividades. Las actividades pueden ser vistas como las acciones atómicas indivisibles en el modelo. El resto de secciones van profundizando en los distintos niveles de detalle del modelo, generando la nomenclatura de cada uno de los elementos que forman el modelo y su relación. Para identificar cada uno de los subprocesos, etapas y actividades se utilizarán las abreviaturas de cada elemento antecesor, es decir, todos los elementos que dependan del proceso creación del modelo contendrán en su nombre CM-... y los de detección D-... De la misma forma los subprocesos dentro de CM serán CM-AP-... y así sucesivamente. De esta forma cada elemento identificará sus dependencias.

6.1. Creación del Modelo (CM)

El primer proceso principal para el desarrollo del sistema de detección de anomalías es la creación del modelo de Machine Learning. Este proceso se va a descomponer en varios subprocesos más, que separan dos bloques de lógica estrechamente relacionados, el subproceso de adquisición y procesado, y el subproceso de entrenamiento de la IA. Estos procesos se describen y nombran de la siguiente manera:

- CM-AP – Adquisición y procesado de datos: este subproceso es responsable de obtener la data en formato raw o sin alterar desde los dispositivos IoT, y los prepara para ser utilizables en los algoritmos de entrenamiento.
- CM-E – Entrenamiento: partiendo de la data ya preparada en el subproceso anterior, aquí se utilizan los algoritmos de IA seleccionados para generar un modelo de detección conveniente.

La Figura 3 muestra un esquema de estos procesos.

Figura 3. Diagrama de los subprocesos que forman la Construcción del Modelo.

Una vez más, estos subprocesos se dividen en diversas etapas, que separan cada una las funciones que será necesario realizar.

6.1.1. Adquisición y procesado de datos (CM-AP)

Cómo se ha comentado anteriormente, este será el primer subproceso del proceso de "Creación del Modelo", la cuál también se va a dividir en diferentes etapas.

- CM-AP-A – Adquisición: esta etapa aísla específicamente todas las actividades relacionadas con la adquisición del dato desde la infraestructura IoT específica. Esta data en bruto será almacenada en un sistema de persistencia adecuado y que permita su almacenamiento sin transformaciones ni adaptaciones a esquemas. Un Almacén NoSQL puede ser ideal ya que se trata de un almacenamiento de tránsito que busca mantener los datos hasta que puedan ser inyectados en la siguiente etapa. Es necesario almacenar la data de esta forma para poder partir de un conjunto de datos en el estado más puro que sea posible.
- CM-AP-PP – Preprocesado: esta etapa realiza una primera transformación de datos y homogeneización, mutando la estructura original de la data a columnas que ya puedan ser tratables y generando una etiquetación de cada columna que en los datos originales puede no existir. Una de las peores características de los datos IoT es la heterogeneidad de sus estructuras, la inclusión de objetos dentro de campos de datos y la utilización de formatos no estructurados. Esta etapa tiene como misión desentramar la estructura y hacerla plana y entendible.
- CM-AP-F – Filtrado relevancia: se realiza un estudio de las características relevantes de la data, ahora ya en columnas tratables, encontrado las relaciones de correspondencia entre ellas y dejando solo las de interés. Los paquetes de datos pueden contener decenas o incluso cientos de campos, pero no todos aportan valor para los algoritmos de IA, por lo

que a través de esta etapa se trata de reducir el conjunto de campos a los que aportan valor y son relevantes para el análisis.

- CM-AP-J – Ajuste: en esta etapa se realiza la primera adecuación de la data a los algoritmos que van a ser utilizados para el entrenamiento de la AI, por ejemplo, ajustando el tipado de datos, ajuste de etiquetas, tratamiento de nulos, etc. Los datos tal y como llegan a esta etapa, a menudo, no serán útiles para ser incorporados al entrenamiento de ningún algoritmo, es necesario estructurarlos para su utilización.

La Figura 4 muestra el diagrama de las etapas descritas.

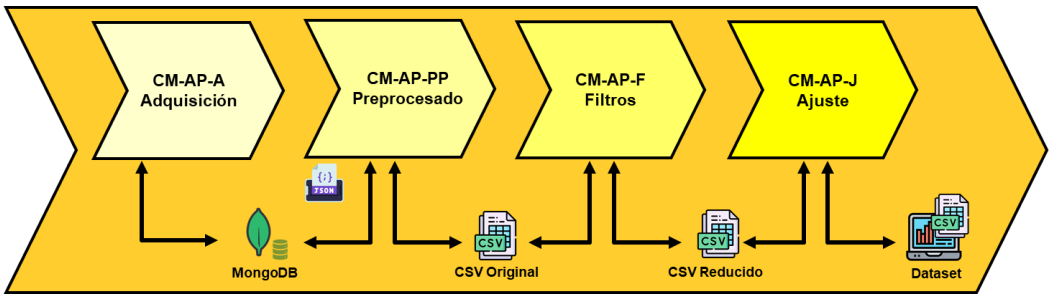


Figura 4. Diagrama de las etapas que forma parte del subproceso Adquisición y Procesamiento.

Finalmente, y cómo último nivel de detalle, estas etapas se dividen a su vez en actividades concretas que pueden ser llevadas a la implementación.

6.1.1.1. Adquisición

La etapa de adquisición se divide en las siguientes actividades:

- CM-AP-A-CL – Conexión y lectura de datos: para la adquisición de datos es necesario establecer un mecanismo de conexión a la infraestructura física, por ejemplo, MQTT, Webhook, API Rest. Esta actividad es la responsable de esta realizarlo.
- CM-AP-A-P – Parse: esta actividad se encarga de realizar un análisis gramatical de la data o parse, para lograr un objeto tratable de datos y debidamente decodificado, por ejemplo, en JSON.
- CM-AP-A-A – Almacenamiento: esta actividad deposita la data tratable en un almacén de datos adecuado para ella, idealmente algún tipo NoSQL que permita almacenar estructuras flexibles, cambiantes y de tamaños desconocidos, por ejemplo, MongoDB.

La Figura 5 muestra el diagrama con las actividades descritas.

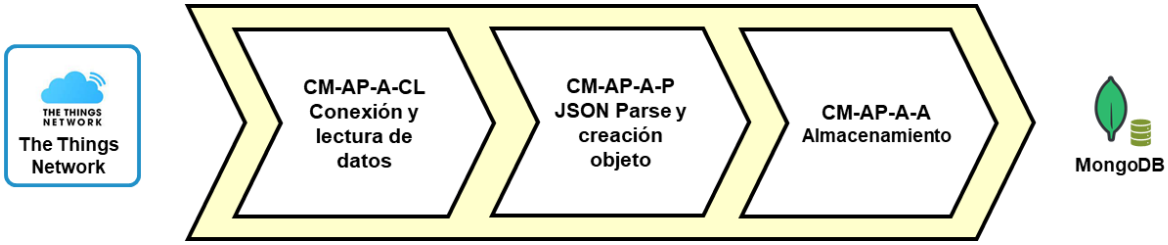


Figura 5. Diagrama de las actividades de la etapa Adquisición.

6.1.1.2. Pre-procesado

La etapa de pre-procesado se divide en las siguientes actividades:

- CM-AP-PP-SF – Selección y formateado inicial: esta actividad extrae los datos desde el almacén o base de datos, y aplica una transformación que permite tratar los datos en memoria, por ejemplo, al transformarlos de CSV a JSON.
- CM-AP-PP-EA – Eliminación de arrays: esta actividad elimina las matrices que habitualmente contienen los datos IoT, desnormalizado los paquetes y así haciendo así que sean tratables por los algoritmos. Esto genera datos con columnas en los objetos.
- CM-AP-PP-RV – Renombrado de variables: los procesos de desnormalización y adaptación de datos pueden generar etiquetas duplicadas, esta actividad realiza un renombrado de etiquetas de forma que sean únicas e identificables.
- CM-AP-PP-GP – Generar paquetes: esta actividad finalmente empaqueta los datos tratados y preparados en formatos tratables por los algoritmos de estudio de correlación posteriores o de entrenamiento de la IA, como por ejemplo CSVs.

La Figura 6 muestra un diagrama de dichas actividades.

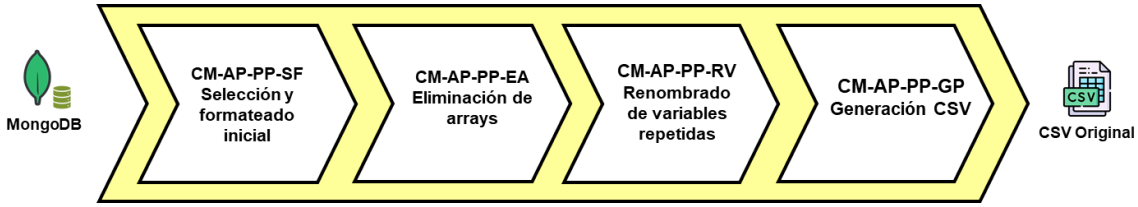


Figura 6. Diagrama de las actividades de la etapa de Pre-procesado.

6.1.1.3. Filtrado

La etapa de filtrado se divide en 3 actividades más:

- CM-AP-F-TC – Transformación de valores categóricos: otro paso importante en la adecuación de datos es la transformación de valores categóricos a numéricos. Esta actividad es la que realiza este proceso.
- CM-AP-F-EC – Estudio de correlación: previo a los entrenamientos, es necesario minimizar la data a tratar pues puede contener cientos o miles de columnas, ya que IoT suele proveer de mucha información sobre la infraestructura intermedia. Esta actividad realiza un estudio de correlación entre las variables para su posterior selección.
- CM-AP-F-SC – Selección de características: a partir del estudio de correlación se hace la selección de los valores relevantes, eliminando los que no aportan información útil al futuro proceso de entrenamiento y obteniendo un dataset minimizado, por ejemplo, en CSV.

La Figura 7 muestra el diagrama de actividades.

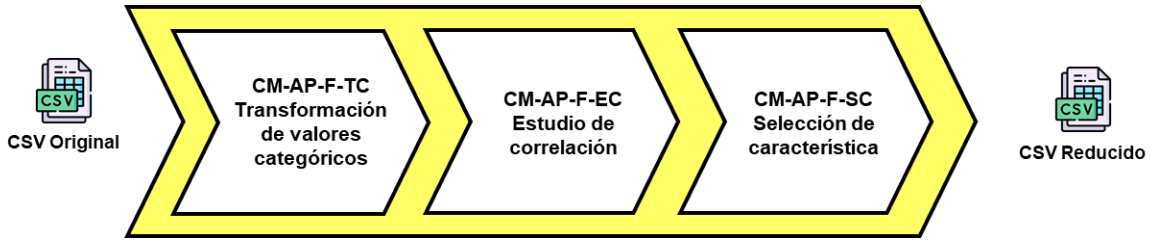


Figura 7. Diagramas de actividades de la etapa de Filtrado

6.1.1.4. Ajuste

La etapa de ajuste se descompone en las siguientes actividades:

- CM-AP-J-LD – Lectura del conjunto de datos: esta actividad realiza la lectura y carga de datos ya filtrados en la etapa anterior en la etapa de ajuste.
- CM-AP-J-D – División del conjunto de datos: esta actividad divide el conjunto de datos en subconjuntos para el entrenamiento, test y validación. Esta división dependerá de los futuros algoritmos a utilizar, pero es necesaria para generar un entrenamiento fiable. Habitualmente generará varios datasets con un porcentaje de datos del conjunto global.
- CM-AP-J-L – Limpieza de datos: se realizará la limpieza de datos por ejemplo eliminando posibles duplicados o series inadecuadas.

- CM-AP-J-TN – Tratamiento de nulos: esta actividad trata los valores nulos y que frecuentemente son inadecuados en los modelos de entrenamiento, fijando valores adecuados a los algoritmos o bien descartando los campos.
- CM-AP-J-EN – Escalado y normalización: esta actividad realiza el escalado y normalización de los datos números para que no produzcan efectos indeseados en los algoritmos de entrenamiento, ya que el entrenamiento de la IA no se comporta adecuadamente cuando el rango de valores es muy irregular.

La Figura 8 muestra los procesos de esta etapa.

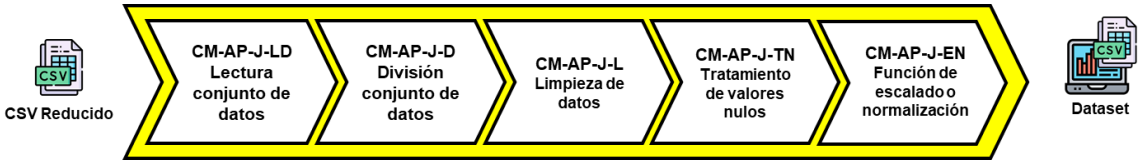


Figura 8. Actividades de la etapa de Ajuste

Con estas actividades se dan por definidas todas las etapas y actividades que forman el subproceso de adquisición y procesamiento. A continuación, se describe el subproceso de entrenamiento.

6.1.2. Entrenamiento

Igual que en el caso anterior, este subproceso se vuelve a dividir en etapas que también se dividen en actividades atómicas. Este es el primer nivel de descripción, las etapas:

- CM-E-SIA – Selección del modelo de IA: esta actividad realiza la selección del modelo de IA que será utilizada, condicionado por ejemplo al tamaño de los datasets, número de dimensiones, etc. Esta actividad es muy importante ya que del algoritmo dependerá la fiabilidad del ADS. Se utilizarán inicialmente los recomendados por la bibliografía.
- CM-E-E – Entrenamiento: esta actividad es la que ya genera el entrenamiento del modelo de IA concreto con los datos ya preparados. Habitualmente se emplea el 70% del dataset general, obtenidos en la actividad CM-AP-J-D y posteriormente tratados.
- CM-E-P – Prueba: esta actividad realiza las pruebas de comprobación sobre el entrenamiento del modelo IA, para ello utiliza uno un 20% de los datasets obtenidos en la actividad CM-AP-J-D y posteriormente tratados, pero disjunto del conjunto de

entrenamiento. Esta actividad puede producir ajustes en el modelo o determinar la necesidad de reentrenamiento.

- CM-E-V – Validación: esta actividad completa el proceso de comprobación del modelo entrenado, validando su funcionamiento sobre el modelo probado. Esta actividad debe obtener el modelo final a utilizar. Se utiliza el 10% restante del dataset obtenido en la actividad CM-AP-J-D y que no ha sido mostrado durante el entrenamiento ni prueba.

La Figura 9 muestra el diagrama de actividades descritas en los párrafos anteriores.

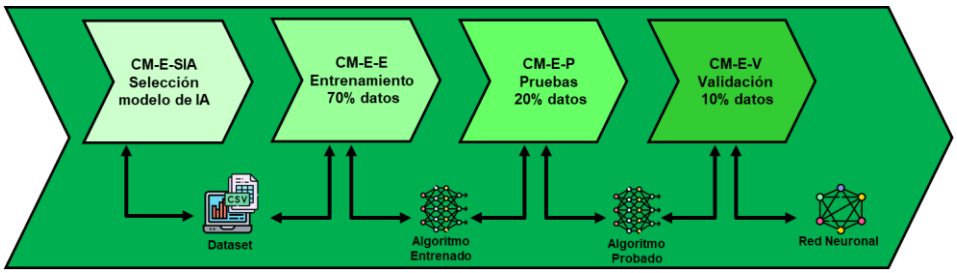


Figura 9. Diagrama de actividades del subproceso Entrenamiento

6.1.2.1. Selección del modelo de IA

Una vez se ha completado la adecuación de los datos, necesitamos seleccionar el modelo de Machine Learning que se va a utilizar para el entrenamiento de la IA. Para este caso, se hizo un estudio previo de los algoritmos no supervisados más utilizados en la actualidad para la detección de anomalías, eligiendo finalmente el algoritmo de Isolation Forest.

Este tipo de algoritmos trabaja en la detección de valores erróneos y sin notificar dentro de los datasets, también llamados outliers o anomalías, definidos como: “una observación que, al ser atípica y/o errónea, se desvía decididamente del comportamiento general de los datos experimentales con respecto a los criterios que deben analizarse sobre ella” [30].

La metodología que emplea el algoritmo de Isolation Forest se basa en la detección de valores atípicos basados en el aislamiento de datos anómalos del resto de los datos mediante el uso de árboles de decisión. Para ello, se selecciona una característica y realiza una división aleatoria entre el valor mínimo y el máximo, repitiendo este proceso hasta realizar todas las divisiones posibles de datos, o en su lugar, se alcance un límite especificado en el número de divisiones. El número de divisiones necesarias para aislar un dato será menor cuando se trate de un valor atípico, mientras que, en caso de valores normales, el número de divisiones será mayor, puesto que el algoritmo

atribuye a cada división una “puntuación” o “anomaly score”, calculado con la media del número de subdivisiones que se han necesitado para aislar el valor anómalo.

El “anomaly score” se trata de un valor calculado mediante la siguiente fórmula (1):

$$s(x,n) = 2^{\frac{-E(h(x))}{c(n)}}$$

Cuyos parámetros son:

- h(x): es la profundidad media de árboles construidos
- c(n): es la altura media para encontrar un nodo en uno de los árboles.
- n: tamaño del dataset. Si el valor obtenido es cercano a 1, generalmente se tratará de una anomalía, mientras que si el valor de s es inferior a 0.5, será un valor correcto.

El motivo de utilizar este algoritmo sobre otros existentes, es principalmente porque se trata de un algoritmo fácilmente escalable para su uso en conjuntos grandes de datos, además de funcionar correctamente cuando se incluyen características que en un principio pueden ser irrelevantes, es decir, conjuntos de datos multimodales, como se da en este caso de infraestructuras IoT, en la que se desconocen la cohesión o correlación interna entre los datos que se envían, y simplemente adquirimos un conjunto de datos con sus correspondientes parámetros y se quieren detectar los valores fuera de lo común.

En cuanto a la implementación para la construcción del modelo, debemos tener en cuenta los elementos básicos con los que podremos entrenar y posteriormente mejorar la precisión del resultado:

- “contamination”, siendo esta la cantidad de datos generales que esperamos que se consideren anomalías. En base a este valor, se establece el límite por el cual se clasifican los valores como anómalos o normales.
- “n_estimators”, número de árboles que forman el modelo.
- “max_samples”, número de observaciones empleadas para entrenar cada árbol.
- “random_state”, semilla utilizada para garantizar la reproducibilidad de los resultados. Hay muchos más parámetros que se pueden aplicar, pero para este caso no se han utilizado.

Queremos destacar que el objetivo de este trabajo no es hallar el mejor de los algoritmos de machine learning, sino de diseñar una metodología que permita utilizar ML sobre una infraestructura IoT general. De hecho, se considera que es posible que dependiendo de la

cantidad y tipo de datos puedan ser mejores o peores unos u otros algoritmos. Se deja como objetivo futuro un posible estudio de la adecuación de algoritmos de ML en función de la caracterización de los datos de la infraestructura. Es decir, poder añadir una etapa más que además de estudiar la relevancia, caracterice los datos, y así seleccione el algoritmo idóneo.

6.2. Detección (D)

El otro gran proceso del modelo de ADS es el de la detección, en el que se utilizarán los resultados del proceso CM, adecuando nuevamente los datos entrantes a un formato que pueda entender el modelo entrenado, para así generar una ejecución en tiempo real. Para ello se establece que el proceso D va a estar dividido en dos grandes subprocesos:

- D-AP – Adquisición y procesado: este subproceso deberá realizar una serie de actividades destinadas a recoger y preparar los datos para ser utilizados en el modelo. Similares a las anteriores pero necesarias algunas de ellas en este subproceso.
- D-EX – Ejecución: este subproceso realizará la ejecución del modelo y tratamiento de los resultados.

Estos subprocesos van estar divididos a su vez en elementos, pero esta vez no existirán etapas sino directamente se dividen en actividades, por lo que solo existirán dos niveles de profundamente del detalle, los subprocesos y las actividades (el proceso de CM tenía tres niveles, subprocesos, etapas y actividades).

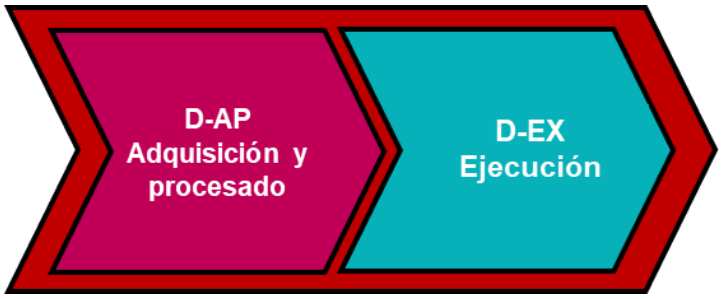


Figura 10. Diagrama de los subprocesos que forman la Detección

6.2.1. Adquisición y procesado durante la detección

Siguiendo el esquema anterior, D-AP estará dividido actividades concretas:

- D-AP-CL – Conexión y lectura: esta actividad realizará las mismas funciones que el proceso CM-AP-A-CL, pero hacia el flujo de datos en directo de la infraestructura IoT a analizar, es decir, capta datos desde el streaming en tiempo real de datos IoT.
- D-AP-P – Parse: actividad que genera un objeto de datos tratable, similar a CM-AP-A-P.
- D-AP-EA – Eliminación de arrays: actividad que prepara el objeto en un formato adecuado a los modelos de IA, similar a CM-AP-PP-EA.
- D-AP-RV – Renombrado de variables: eliminar variables duplicadas y adaptar el objeto de datos al formato utilizado para entrenamiento, similar a CM-AP-PP-RV.
- D-AP-SC – Selección de características: eliminación de todas las variables no utilizadas en el modelo IA, según el proceso CM-AP-F-SC.

La Figura 11 muestra las actividades descritas.

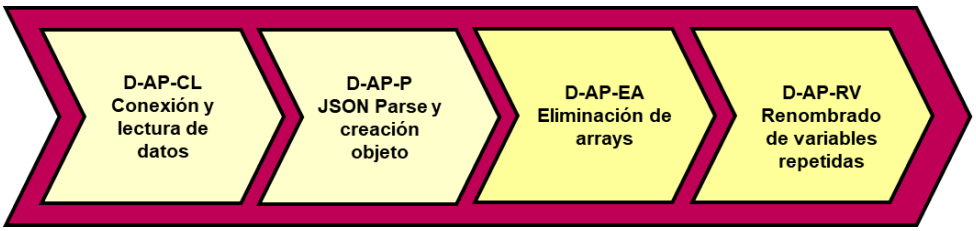


Figura 11. Diagrama de actividades del subproceso Adquisición y procesamiento durante la detección.

6.2.2. Ejecución

Finalmente, D-EX también contendrá una serie de actividades que generarán finalmente el análisis:

- D-EX-IA – Aplicar el modelo IA: esta actividad somete al flujo de datos al modelo IA entrenado en los procesos anteriores.
- D-EX-D – Decisión: ante el resultado de la AI, el modelo emite una decisión, una catalogación del dato para que pueda continuar el tránsito hacia la lógica de negocio del sistema o bien que sea puesto bajo observación.

La Figura 12 muestra el diagrama de las actividades descritas.

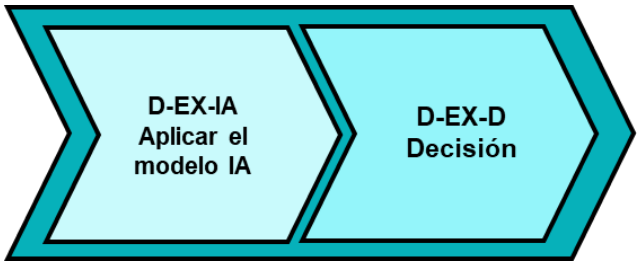


Figura 12. Diagrama de las actividades del subproceso Ejecución

6.3. Vista general del modelo

Finalmente se porta una vista general del modelo descrito, figura X11, con todos los elementos ordenados por procesos, subprocesos, etapas y actividades. Se puede observar claramente que, aunque la detección es el proceso que clasifica realmente el flujo de datos, es la fase de construcción la que contiene más actividades que pueden influir negativamente en el desarrollo de ADS en caso de no ser adecuadamente tratadas.

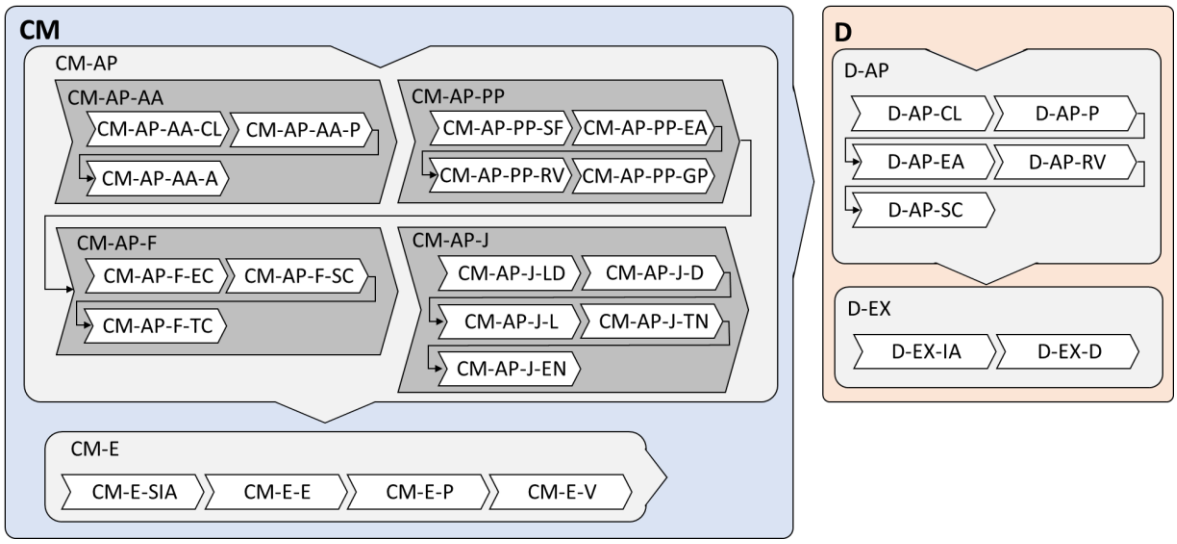


Figura 13. Diagrama general del modelo

7. Desarrollo e implementación de la solución

Una vez se han definido los aspectos generales del diseño para la creación de un sistema de detección de anomalías, pasamos al desarrollo de la solución. Para ello se utilizarán las diferentes tecnologías descritas en el capítulo 6. Para el desarrollo se crearán unidades de lógica en forma de fragmentos de código o script, que aislen las funcionalidades de cada uno de los procesos descritos anteriormente. Cada unidad recibirá una entrada, realizará una actividad y generará una salida, tanto entrada como salida podrán ser elementos de diversa naturaleza como conjuntos de datos, redes neuronales, resultado de testing, etc.

Dado que algunos procesos ya son dependientes de las tecnologías a las que dan cobertura, será necesario utilizar librerías, procedimientos y estructuras de datos propias de la plataforma IoT concreta. Sobre todo, los procesos más cercanos a la plataforma, como los que extraen datos, requieren de librerías específicas. Vamos a particularizar estos scripts para conectar con la plataforma IoT de *The Thing Networks* - TTN [31]. Esta plataforma es un ecosistema colaborativo global del Internet de las Cosas que crea redes, dispositivos y soluciones utilizando LoRaWAN. La plataforma permite utilizar de forma abierta todos los gateways que la forman, o incluso añadir tu propio gateway, para captar paquetes de datos generados desde sensores y transmitido por Lora. Esta plataforma ejemplifica un caso típico donde no se tiene control sobre las infraestructuras y no se tiene información detallada sobre los dispositivos que la forman, y por lo tanto debemos basarnos sólo en los datos que conocemos. El prototipo que va a ser creado en esta sección es válido para el procesamiento ante cualquier infraestructura, aunque requeriría algunas adaptaciones en las partes que dependan de la tecnología IoT (por ejemplo, el método de conexión para obtener el tráfico IoT). Además, se han elegido tecnologías como NodeJS o Python para la implementación por comodidad, pero se podrían utilizar otras. La importancia de este desarrollo reside en las actividades y acciones que se llevan a cabo, que conceptualmente sí han de ser las diseñadas.

7.1. Etapa CM-AP-A - Adquisición

Como se ha comentado durante el capítulo 7, Diseño conceptual de la Solución, la primera etapa a realizar es la adquisición de los datos con los que se creará el dataset para el entrenamiento, además de posteriormente ser los mensajes que se analizarán en la implementación del ADS. En esta etapa se implementan varias actividades.

Para la implementación de esta etapa se utilizará lenguaje de scripting, y la lógica de negocio estará dividida en las actividades de la etapa. El siguiente pseudocódigo muestra las acciones concretas a llevar a cabo en cada etapa y sus actividades:

```
[Actividad CM-AP-A-CL]
      Definir conexión a BD
      Definir conexión a plataforma IoT - TTN
      Activar Listener MQTT

[Actividad CM-AP-A-P]
      Cargar mensaje a JSON
      Extraer campos para creación de objeto
      Generar objeto de datos para cada gateway (deserializado)
      Apilamos objeto en cola de datos

[Actividad CM-AP-A-A]
      Almacenar en BD
```

Código 1. Pseudocódigo de las acciones de la implementación de la etapa CM-AP-A

En nuestro caso la solución ha sido desarrollada mediante un script en NodeJS [32], utilizando dos librerías para conectar a la BD MongoDB[33] donde se almacenarán los datos adquiridos, y para conectar a MQTT [34] que es el broker de mensajes a través del cual se obtienen los datos. El siguiente fragmento de código muestra cómo se realiza la implementación de CM-AP-A-CL.

```
const MongoClient = require('mongodb').MongoClient;
const mqtt = require('mqtt')

const url = "mongodb://localhost:27017/iotInfraestructuras";
const clientUrl = new MongoClient(url);

const database = clientUrl.db("iotInfraestructuras");
const db = database.collection("sensores");

const appID='ua-sensor@ttn';
const accessKey='accessKey';

const client = mqtt.connect(('mqtt://eu1.cloud.thethings.network:1883'), {
  username: appID,
  password: accessKey
});

client.on('connect', () => {
  client.subscribe(['v3/ua-sensor@ttn/devices/+/up']);

});
```

Código 2. Fragmento de código que implementa la conexión a BD y a broker MQTT

Para empezar, es necesario conectarse a la plataforma desde la que se reciben los datos de los sensores[CM-AP-A-CL], The Things Network, mediante el protocolo MQTT. Para ello, se utilizan tres parámetros imprescindibles para la conexión, el primero de ellos es el nombre de la aplicación desde la que trabajarán los sensores, a la que llamaremos "appID", además de una clave de acceso llamada "API Key". Esta clave es única a la hora de su creación y se obtiene desde la plataforma de TTN. Por último, desde el método de conexión por MQTT de JS, se añade también el *cluster* al que se va a conectar, junto a los dos anteriores parámetros. Esta conexión activa el *listener* que quedará a la escucha para recibir paquetes de datos.

Una vez tenemos los elementos necesarios para establecer la conexión con la plataforma, creamos la conexión a una base de datos en local, en este caso MongoDB, en la que se almacenará toda la información en raw que nos llega de cada mensaje. Se ha elegido este tipo de base de datos no relacional debido a la flexibilidad que presenta a la hora del formato en que se almacenan los datos, puesto que al recibir los mensajes en formato JSON [35], es más cómodo poder almacenar el dato con una estructura de documento JSON idéntica al formato que se recibe mediante MQTT.

Los elementos para la conexión a la base de datos de MongoDB son los básicos y generales como es crear una URL de cliente con el nombre de la base de datos a la que se va a conectar, junto al nombre de la colección principal a utilizar.

Tras especificar los elementos indispensables para ambas conexiones, tanto para la plataforma desde la que provienen los datos como la herramienta a utilizar para almacenarlos, pasamos a la obtención de estos.

Será necesario "parsear" el mensaje obtenido a JSON [CM-AP-A-P], puesto que si no se encuentra en un formato desde el que no se puede leer la información fácilmente.

```
var payload = JSON.parse(message.toString());
```

Código 3. Conversión del mensaje a JSON

Una vez que se tiene el mensaje listo para su tratamiento se realiza la extracción de campos. Este proceso es dependiente de la plataforma IoT, ya que en función de la tecnología IoT se reciben unos datos u otros. Algunos datos importantes a localizar son, por ejemplo: timestamp del mensaje, device_id, appliction_id, uplink, uplink_bites y correlation_ids. Estos datos suelen ser comunes en IoT e indican información importante en la transmisión de datos. En nuestro caso la información se extrae mediante este código:

```
timestamp = Date.parse(payload.uplink_message.rx_metadata[0].time);
ttn_device_id = payload.end_device_ids.device_id;
ttn_application_id = payload.end_device_ids.application_ids.application_id;
ttn_uplink = [payload.uplink_message.decoded_payload.bytes];
ttn_uplink_bytes = payload.uplink_message.decoded_payload.bytes;
ttn_correlation_ids = payload.correlation_ids;
```

Código 4. Código de extracción de datos básicos

Tras este paso, se crea el objeto con toda la información contenida en el mensaje, pero cada elemento añadido de manera específica en su correspondiente "etiqueta".

```
for(i=0; i<rx_metadata.length; i++){
        objData = {
            "gateway_ids": {
                "gateway_id": rx_metadata[i].gateway_ids.gateway_id,
                "eui": rx_metadata[i].gateway_ids.eui
            },
            "time": rx_metadata[i].time,
            "timestamp": rx_metadata[i].timestamp,
            "rssi": rx_metadata[i].rssi,
            "channel_rssi": rx_metadata[i].channel_rssi,
            "snr": rx_metadata[i].snr,
           …
```

Código 5. Ejemplo de creación de objeto de datos para su almacenamiento

Algunos de los elementos que se reciben se encuentran dentro de un array, por lo que es necesario recorrerlos y añadirlos al objeto principal. Este proceso se repite con algunos de los parámetros recibidos como son los "correlation_ids", o la información que nos llega de los diferentes *gateways* que detecta un sensor durante su transmisión. Como se ha comentado anteriormente, este proceso será dependiente de la estructura interna de los datos.

Finalmente, en cuanto se termina de construir el objeto, se inserta dentro de la colección de MongoDB. Este proceso se repetirá mientras que se mantenga la conexión con la plataforma de TTN. [CM-AP-A-A]

7.2. Etapa CM-AP-PP - Preprocesado

Este proceso en particular ha precisado de dos scripts diferentes, el primero de ellos para la selección de los datos que iban a constituir el primer CSV, puesto que no era viable realizar el resto de fases del desarrollo de la solución a partir de un archivo .JSON de más de 300 mil datos.

Al igual que en el proceso anterior, el siguiente pseudocódigo muestra las actividades correspondientes a esta etapa:

```
[Actividad CM-AP-PP-SF]
      Importar Librerías
      Abrir archivo .json
      Recorrer gateways del objeto

[Actividad CM-AP-PP-EA]
      Adecuar formato de los datos
      Comprobar si hay parámetros faltantes
      Extraer campos para creación de objeto
      Eliminación de arrays


[Actividad CM-AP-PP-RV]
      Renombrado de variables para los arrays desglosados

[Actividad CM-AP-PP-GP]
      Apilamos objeto en cola del dataframe
      Generar CSV
```

Código 6. Pseudocódigo de las acciones de la implementación de la etapa CM-AP-PP

Por ello, para empezar, obtendremos el archivo JSON recientemente comentado desde la base de datos de MongoDB, lo que nos deja un primer archivo desde el que crear los diferentes CSV de pruebas, con diferentes tamaños entre ellos, además de variar la cantidad de datos correctos y anómalos dentro de cada uno de ellos. [CM-AP-PP-SF]

Para procesar este bloque de datos, serán necesarias dos librerías con las que poder procesar el objeto JSON, una especializado en tratar con JSON (o el formato que se fuese a utilizar) y otra especializada en tratar DATASETS, en nuestro caso serán *json* [36] y *pandas* [37]. A continuación, se lee el archivo .JSON y empezaremos a recorrer en base a la cantidad de datos que queramos añadir a nuestro CSV, separando el proceso principal en dos bucles, uno para la primera parte del archivo en que se encuentran los valores anómalos, y otro para los datos que tienen todos los valores en el formato correcto.

El siguiente fragmento de código muestra cómo se realiza la implementación de CM-AP-PP-SF.

```
import pandas as pd
import json

dt = pd.DataFrame()

with open('sensores_con_id.json') as data:
    sensors = json.load(data)
    while count <= XXXX:
        for sensor in sensors:
```

Código 7. Fragmento de código que implementa la importación de librerías, apertura del archivo .json y bucle principal

Para este script, dentro de cada bucle se aplica el primer cambio respecto a la entrada de datos original, puesto que hay elementos que no existen en el objeto en comparación a los datos que se añadieron posteriormente, por lo que en cuanto se detecta la inexistencia de este campo en particular, llamado "uplink_message.decoded_payload.bytes", se añade de manera manual un array con un valor de 0 en su interior, de forma que tenga el mismo formato de que este campo contenga un array al igual que el resto de datos, pero resultando en un valor erróneo, puesto que los sensores están programados para enviar los bytes en formato Float, es decir, de 4 bytes en 4 bytes, lo que resulta en el valor mínimo del array.

El siguiente cambio a aplicar al JSON original, se va a realizar en los dos campos de timestamp que nos llegan desde el mensaje, uno dentro del campo de rx_metadata, "timestamp", y el otro dentro del mensaje general, "uplink_message.settings.timestamp". Este cambio se debe a que en algunos de los mensajes recibidos desde TTN, el campo de timestamp puede aparecer de dos formas, una en la que se muestra directamente el valor en formato integer, o bien dentro de otro subcampo llamado $numberLong, de forma que si se hiciera una obtención de los datos común para todos los valores, habría elementos que se adquirirán con el valor incorrecto. De esta forma, en cualquiera de los casos se obtiene el valor del tiempo correctamente.

```
for gateway in sensor['obj']['data']['uplink_message']['rx_metadata'][0]:
    valor = sensor['obj']['data']['uplink_message'].get('decoded_payload')

    if valor is None:
       sensor['obj']['data']['uplink_message']['decoded_payload']={'bytes': [0]}

    timestamp = gateway['timestamp']

    if type(timestamp) is str or type(timestamp) is int:
       gateway['timestamp'] = gateway['timestamp']
    else:
       gateway['timestamp'] = gateway['timestamp'].get('$numberLong')
```

Código 8. Fragmento de código que implementa algunos cambios en los datos obtenidos del archivo .json

Tras aplicar estos dos cambios a los datos que lo requieran durante el bucle que se está llevando a cabo, se construye el data frame que nos facilita la librería de pandas, añadiendo además dos elementos nuevos al objeto original, que son el campo *id* e *_id.$oid* que nos genera MongoDB al insertar el objeto.

Para la creación del dataframe, puesto que no se puede realizar un objeto genérico debido al formato de los datos entrantes, es necesario ajustar el script en base al mensaje recibido además de que también sea un formato que posteriormente pueda ser adecuado para el modelo, de manera que hay ciertos campos que ajustar. En primer lugar, existe dentro del mensaje procedente de TTN un elemento llamado “correlation_ids”, el cuál es un array con un número fijo de elementos, aunque su contenido cambie a cada mensaje enviado. Sabiendo esta particularidad, es necesario separar el array original en elementos separados [CM-AP-PP-EA], y cambiar cada uno de sus nombres, de manera que resulten en columnas independientes con respecto al contenido inicial. La nueva denominación con la que se crea el campo

de este valor, es el nombre del campo, junto con su correspondiente numeración, por ejemplo, “correlation_ids0”, “correlation_ids1”... y así sucesivamente. [CM-AP-PP-RV]

El siguiente fragmento de código muestra cómo se realiza la implementación de CM-AP-PP-EA, CM-AP-PP-RV y CM-AP-PP-GP.

```
gateway_dataframe = pd.DataFrame({
    "id": [id],
    "_id.$oid": [sensor['_id']['$oid']],

    "end_device_ids.device_id":[sensor['obj']['data']
      ['end_device_ids']['device_id']],
    "end_device_ids.application_ids.application_id":
      [sensor['obj']['data']['end_device_ids']
      ['application_ids']['application_id']],

    "correlation_ids0":[sensor['obj']['data']['correlation_ids'][0]],
    "correlation_ids1":[sensor['obj']['data']['correlation_ids'][1]],
    "correlation_ids2":[sensor['obj']['data']['correlation_ids'][2]],
    "correlation_ids3":[sensor['obj']['data']['correlation_ids'][3]],
    "correlation_ids4":[sensor['obj']['data']['correlation_ids'][4]],
    "correlation_ids5":[sensor['obj']['data']['correlation_ids'][5]],
    "correlation_ids6":[sensor['obj']['data']['correlation_ids'][6]],
    "received_at": [sensor['obj']['data']['received_at']],
    "location.latitude": [gateway['location']['latitude']],
    "location.longitude": [gateway['location']['longitude']],
    "location.altitude": [gateway['location']['altitude']],
    "location.source": [gateway['location']['source']],

    })
    dt = pd.concat([dt, gateway_dataframe], axis=0)
  dt.to_csv("csv5000Datos.csv")
break
```

Código 9. Fragmento de código que implementa partes de la creación del DataFrame y la generación del archivo CSV

En el momento que se termina de construir el data frame, utilizaremos el método “to_csv” para convertir dicho data frame en un archivo CSV. [CM-AP-PP-GP]

La única diferencia con este script en cuanto al segundo que se utiliza, es que desde este se introducen de manera concreta los valores que vamos a recorrer, tanto el inicio y el final, como la cantidad de ellos que se van a añadir al archivo CSV. El segundo archivo se utilizará cuando no sea necesario generar un csv con una cantidad de elementos específico, sino que se utilizarán todos los elementos disponibles en la base de datos de MongoDB.

7.3. Etapa CM-AP-F – Filtrado de relevancia

Al igual que en el resto de procesos mencionados a lo largo de este documento, este apartado también estará subdividido en diferentes actividades, con la diferencia de que, en este caso, cada actividad tendrá su propio script diferente, de manera que posteriormente se puedan reutilizar sin que sea necesario cambiar o repetir pasos que son innecesarios.

El pseudocódigo correspondiente a esta etapa es el siguiente:

```
[Actividad CM-AP-F-TC]
      Importar librerías
      Leer archivo CSV en formato DataFrame
      Transformación de valores categóricos a numéricos

[Actividad CM-AP-F-EC]
      Estudio de correlación

[Actividad CM-AP-F-SC]
      Selección de características
      Eliminar valores innecesarios
      Mostrar resultados de la correlación
```

Código 10. Pseudocódigo de las acciones de la implementación de la etapa CM-AP-F

Para empezar, antes de poder realizar un filtrado de los datos que no aportarán nada al modelo, es necesario realizar un estudio de la correlación entre los diferentes datos que tenemos. Para poder hacer esto, mediante la librería de pandas, se lee el archivo CSV generado durante la fase de preprocesamiento [*38*].

Antes de poder realizar el estudio, es importante recordar que este método sólo realizará el estudio de los valores numéricos, de forma que los valores categóricos no se mostrarán en el resultado. Por ello, se aplica previamente un proceso de conversión de variables categóricas a numéricas mediante la técnica de codificación de one-hot, basado en etiquetar a qué clase pertenecen los datos, donde la idea es asignar 0 a toda la dimensión, excepto 1 para la clase a la que pertenecen los datos. Este proceso se realiza nuevamente en base a la librería de pandas, con el método ***factorize***. [CM-AP-F-TC]

```
from google.colab import files
uploaded = files.upload()

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('csvDatos.csv')
df['_id.$oid'], oid_c = pd.factorize(df['_id.$oid'])
df['time'], time_c = pd.factorize(df['time'])
df['location.source'], source_c = pd.factorize(df['location.source'])
df['uplink_token'], uplink_token_c = pd.factorize(df['uplink_token'])
```

Código 11. Fragmento de código que implementa parte de las conversiones categóricas a numéricas

Finalmente, utilizando el método ***.corr(),*** lo que nos va a permitir es realizar esa matriz de correlación entre los elementos. Este resultado se mostrará en pantalla mediante las librerías de seaborn [39] y matplotlib.pyplot [40], con el objetivo de crear un "heatmap" o mapa de calor, mostrando tanto de manera visual como numérica el resto de este estudio. [CM-AP-F-EC]

```
matriz_correlacion = df.corr()

plt.figure(figsize = (40,15))
sns.heatmap(matriz_correlacion, annot=True, cmap='coolwarm')
```

Código 12.Fragmento de código que implementa el estudio de correlación y su representación

Al observar el resultado de la Figura 14, se puede observar que hay algunos parámetros que no muestran ningún resultado de correlación. Esto se debe a que se tratan de columnas constantes, es decir, son columnas que contienen el mismo valor en cada una de sus entradas, de forma que no se calcula la correlación al no haber una variación suficiente de los datos. Como norma general, estos valores se deben eliminar de manera previa a realizar la matriz de correlación, sin embargo, partimos de no saber con suficiente seguridad el contenido de los datos, por lo que en este caso se realiza ese estudio aún con columnas que nos pueden mostrar este tipo de resultados.

Figura 14. Estudio de Correlación sin aplicar factorize

El estudio de correlación con todos los valores iniciales se encuentra en el Apéndice I.

Al realizar el estudio de correlación, se analiza qué valores van a permanecer a lo largo del desarrollo del ADS. Hay que tener en cuenta que, para este modelo en particular, se toman decisiones tanto de manera lógica debido a los valores obtenidos en la matriz de correlación, como de manera subjetiva, en la que entra el factor de decisión humana sobre determinados valores que se requiere que se mantengan, puesto que algunos de ellos van a resultar en las futuras anomalías a detectar, aunque su correlación no sea la idónea para que permanezcan en el dataframe. [CM-AP-F-SC]

Es cierto que en este mismo script se podrían eliminar las columnas que no aportan información, pero dentro del desarrollo de modelos de Machine Learning, es una buena práctica separar los procesos, de manera que luego pueden reutilizarse para otros modelos, o incluso cuando todas estas fases se encuentran automatizadas, se puede monitorizar en qué momento de la cadena de ejecución ha fallado.

Para no repetir demasiados pasos, se genera un nuevo archivo CSV, de forma que el proceso de filtrado simplemente tendrá que eliminar las columnas que ya no requiera, sin necesidad de aplicar nuevamente el proceso de pasar los valores categóricos a numéricos.

De esta forma, el siguiente script leerá dicho CSV generado en la ejecución anterior, pero ahora especificaremos las columnas a eliminar del CSV, utilizando el método ***.drop().***

```
df = df.drop([  'Unnamed: 0',
    'id',
          '_id.$oid',
          'end_device_ids.device_id',
          'end_device_ids.application_ids.application_id',
          'correlation_ids0',
          'correlation_ids1',
          'correlation_ids2',
          'correlation_ids3',
          'correlation_ids4',
          'correlation_ids5',
          'correlation_ids6',
          'uplink_message.f_port',
          'uplink_message.f_cnt',
          'gateway_ids.eui',
          'location.source',
          'uplink_message.settings.data_rate.lora.bandwidth',
          'uplink_message.settings.data_rate.lora.coding_rate',
          'uplink_message.settings.frequency',
          'uplink_message.settings.timestamp',
          'uplink_message.settings.time',
          'uplink_message.network_ids.net_id',
```

```
            'uplink_message.network_ids.tenant_id',
            'uplink_message.network_ids.cluster_id',
            'uplink_message.network_ids.cluster_address'
            ], axis=1)
```

Código 13. Fragmento de código que implementa la selección de características

Finalmente, generamos el CSV reducido con el que trabajar durante la siguiente fase.

7.4. Etapa CM-AP-J - Ajuste

Esta es la última fase relacionada con el tratamiento de los datos, de manera que al completarla obtendremos un dataset con el que comenzar el entrenamiento del modelo.

De manera común a algunas fases o actividades mencionadas previamente, trabajaremos en base al archivo CSV generado en la fase anterior, de forma que se realiza la lectura de los datos con los que vamos a trabajar. [CM-AP-J-LD]

```
[Actividad CM-AP-J-LD]
      Lectura del conjunto de datos

[Actividad CM-AP-J-D]
      División del conjunto de datos

[Actividad CM-AP-J-L]
      Obtener todas las columnas del CSV
      Eliminar valores inadecuados de los datos

[Actividad CM-AP-J-TN]
      Rellenar los valores nulos con un valor predeterminado -1

[Actividad CM-AP-J-EN]
      Aplicar función de escalado
```

Código 14. Pseudocódigo de las acciones de la implementación de la etapa CM-AP-J

Debido a cómo se ha programado la creación de los CSV, siempre que se lee el archivo vemos que aparece una columna que no pertenece a nuestro conjunto de datos de los mensajes de TTN, sino que es un index numérico que se crea. Al resultar innecesario, se debe realizar una limpieza de este tipo de columnas o valores. [CM-AP-J-L]

Una vez tenemos el CSV con el que trabajar, lo primero que tendremos que hacer será comprobar si hay algún valor nulo entre las columnas de nuestro dataset. Para ello, utilizaremos los métodos “isna()”, además de “any()”, que nos ofrece la librería de pandas. El primero de ellos, nos devuelve valores booleanos de true o false si el dato corresponde a un NULL o a un NaN. Este método,

utilizado a la par que el método de any(), se comprueba si alguna columna contiene algún valor nulo.

Tras comprobar los valores vacíos de nuestro dataset, hay que determinar el tratamiento que se va a seguir para estos datos. Existen varias formas de tratar los valores nulos:

- Eliminar las filas correspondientes
- Eliminar el atributo correspondiente
- Rellenar con un valor determinado

Para el desarrollo del algoritmo, se ha decidido rellenar los valores nulos con un valor predeterminado, de manera que no se pierdan columnas, filas, o datos concretos. Mediante la librería de sklearn.impute [41], crearemos un objeto "SimpleImputer" con la estrategia de "constant", lo que significa que se van a reemplazar los valores nulos con un valor constante, pudiendo ser tanto un número como una cadena de texto, sin embargo, puesto que sabemos que los algoritmos de machine learning no comprendo los elementos de formato categórico, los valores nulos se rellenarán con el valor de -1.

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='constant', fill_value=-1)

imputer.fit(df)
df = imputer.transform(df)
df = pd.DataFrame(df, columns = attributes)
```

Código 15. Fragmento de código que implementa el tratamiento de nulos

Al crear el objeto, pasamos a ajustar el "imputador" de los datos, de forma que mediante el método .fit(), se le proporcionan los atributos numéricos para que calcule los valores necesarios, para finalmente rellenarlos mediante el método de transform().

A continuación, hay que tener en cuenta que los algoritmos de Machine Learning no se comportan adecuadamente si los valores de las características que reciben como entrada se encuentran en rangos muy dispares. Por ello, se utilizan distintas técnicas de escalado.

- Normalización: Los valores del atributo se escalan para adquirir un valor entre 0 y 1.
- Estandarización: Los valores del atributo se escalan y reciben un valor similar pero no se encuentran dentro de un rango.

Para ello, utilizaremos el método de RobustScaler, que aplica un escalado a las características de forma que sea más robusto a los valores atípicos o outliers.

```
from sklearn.preprocessing import RobustScaler
scale_attrs = df[['received_at',
                  'uplink_message.session_key_id',
                  'uplink_message.frm_payload',
                  'gateway_ids.gateway_id',
                  'time',
                  'uplink_message.received_at',
                  'uplink_message.consumed_airtime']]

robust_scaler = RobustScaler()
df_scaled = robust_scaler.fit_transform(scale_attrs)

df = pd.DataFrame(df_scaled, columns=['received_at',
                  'uplink_message.session_key_id',
                  'uplink_message.frm_payload',
                  'gateway_ids.gateway_id',
                  'time',
                  'uplink_message.received_at',
                  'uplink_message.consumed_airtime'])
df = df.append(df_scaled)
```

Código 16. Fragmento de código que implementa el escalado de datos

Finalmente, volvemos a obtener un CSV con el que trabajar en la siguiente fase.

7.5. Etapa CM-E - Entrenamiento

Durante el proceso de entrenamiento, estará dividido en tres scripts diferentes, el primero de ellos contendrá el entrenamiento principal, con el 70% de los datos del dataset. Para ello, nuevamente utilizaremos el CSV generado por el proceso anterior [CM-AP-J – Ajuste].

```
[Actividad CM-E-SIA]
      Selección del modelo de IA
      Definir parámetros de entrada
      Definir valores de entrenamiento

[Actividad CM-E-E]
      Entrenamiento

[Actividad CM-E-P]
      Prueba

[Actividad CM-E-V]
      Validación
```

Código 17. Pseudocódigo de las acciones de la implementación de la etapa CM-E

A partir de aquí, construiremos el modelo de Isolation Forest, utilizando también la librería de Scikit-Learn. Esta librería es una de las más utilizadas durante los procesos de aprendizaje automático, puesto que proporciona una gran variedad de algoritmos y herramientas para entrenar modelos enfocados a diferentes áreas.

Para poder entrenar nuestro modelo, será necesario definir los parámetros de entrada que se van a considerar posibles anomalías, en nuestro caso, se determina que todas las columnas son susceptibles de sufrir algún tipo de anomalía, por lo que se crea un array con todas las etiquetas restantes tras el proceso de filtrado.

```
import pandas as pd
import seaborn as sns
from sklearn.ensemble import IsolationForest

df = pd.read_csv('dataset300.csv')
anomaly_inputs = ['received_at', 'uplink_message.session_key_id',
'uplink_message.frm_payload', 'uplink_message.decoded_payload.bytes',
'gateway_ids.gateway_id'...]
```

Código 18. Fragmento de código que implementa la definición de parámetros de entrada

Finalmente, comenzamos con la parte del entrenamiento, en la que definimos los valores que se utilizan para mejorar la precisión del modelo, además de explicar el porqué de estos valores.

Las siguientes líneas son un ejemplo del código utilizado para el entrenamiento:

```
model_IF =
IsolationForest(n_estimators=100,contamination=float(0.1667),
max_features=1.0, max_samples='auto')

model_IF.fit(df[anomaly_inputs])

df['anomaly_scores'] = model_IF.decision_function(df[anomaly_inputs])
df['anomaly'] = model_IF.predict(df[anomaly_inputs])
```

Código 19. Fragmento de código que implementa la definición de parámetros de entrenamiento

- n_estimators=100: Este valor representa el número de *isolation trees* que se van a utilizar para construir el propio *isolation forest.* Usar valores altos de estimador puede mejorar la precisión para la detección, pero siempre debe adecuarse al dataset utilizado, puesto que también resulta en un incremento del tiempo de entrenamiento. Variar el valor de este parámetro servirá para ajustar el rendimiento final del modelo.

- contamination=float(0.0.1667): Este valor indica la proporción estimada de anomalías que posee el dataset. En el caso del dataset de entrenamiento, se intima que un 16’67% de los datos contienen un valor erróneo que se debe detectar, por lo que se ajusta el entrenamiento a ese índice de contaminación.

- max_features=1.0: Este valor indica el número máximo de características que se utilizarán al dividir cada nodo del árbol. Al determinar que el valor es 1, especificamos que se utilizarán todas las características disponibles para cada división. Este parámetro se puede utilizar así en este caso puesto que se han filtrado muchas columnas durante el proceso de adecuación de los datos, sin embargo, si el dataset tuviera un gran número de columnas diferentes, se puede ajustar de forma que se mejore el rendimiento y se evite el sobreajuste.

- max_samples='auto': Este valor sirve para controlar el máximo número de muestras que se van a utilizar para entrenar cada árbol que se genere. El valor de auto implica que se utilizarán todas las muestras del dataset, sin embargo, si se tuviera un dataset demasiado grande, este valor tendría que reajustarse.

Una vez finalizado el entrenamiento, podemos ir visualizando los resultados obtenidos por cada campo del dataset. Para poder tener una gráfica para cada campo, consideramos como elemento común el campo “time”, de forma que las siguientes gráficas se corresponden con el campo tiempo, y otro parámetro del dataset, como puede ser el snr. A continuación, se muestra el resultado del entrenamiento con algunas variables donde se pueden apreciar los outliers.

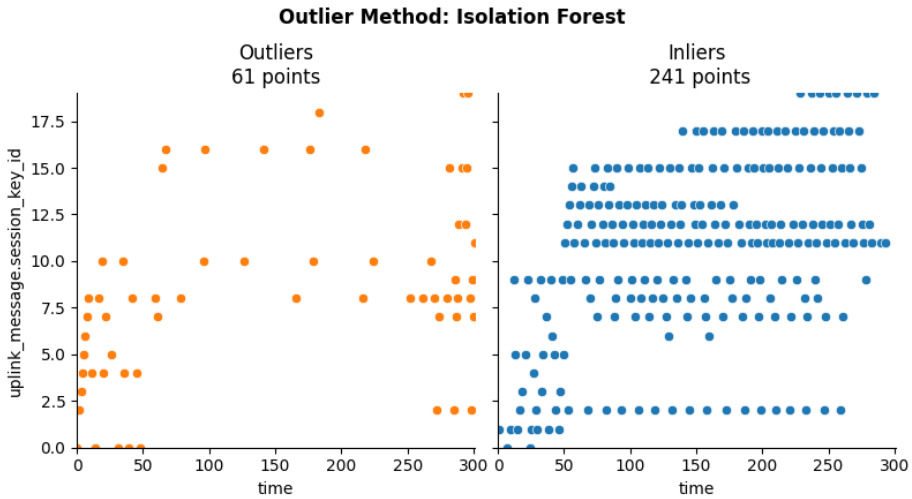


Figura 15. Resultado de la detección sobre la columna uplink_message_session_key_id en relación con time

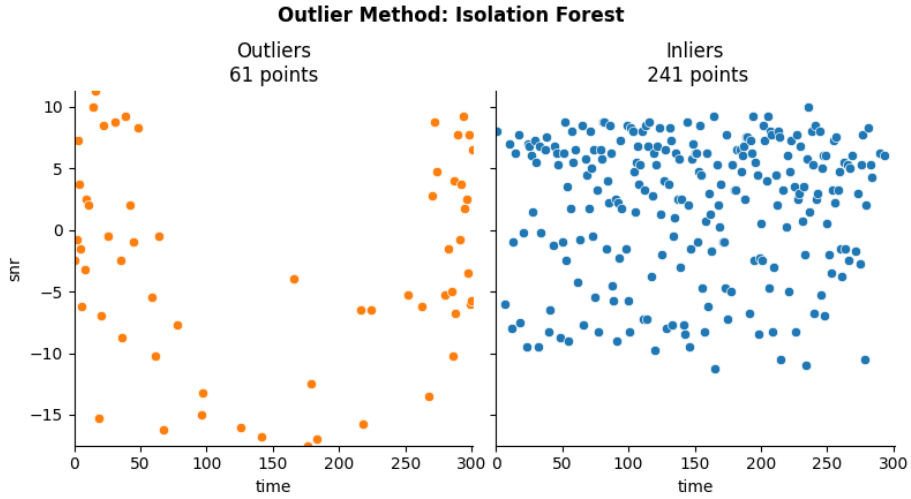


Figura 16. Resultado de la detección sobre la columna snr en relación con time

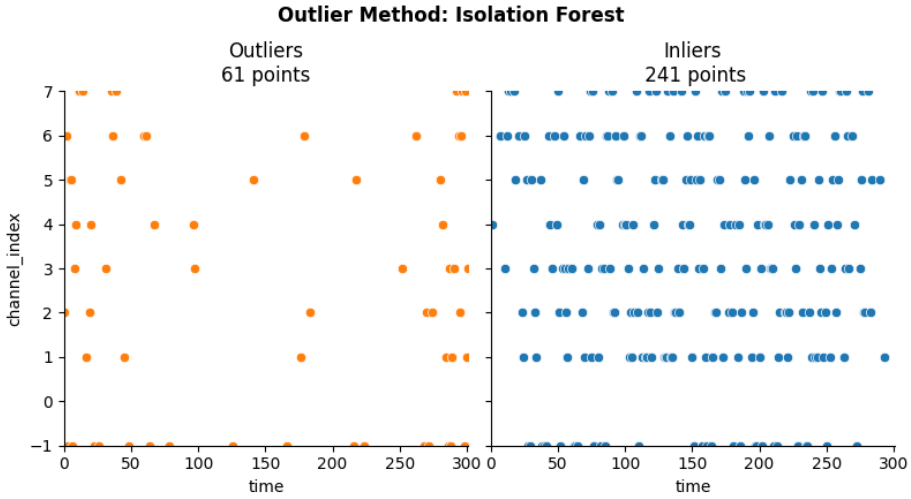


Figura 17. Resultado de la detección sobre la columna channel_index en relación con time

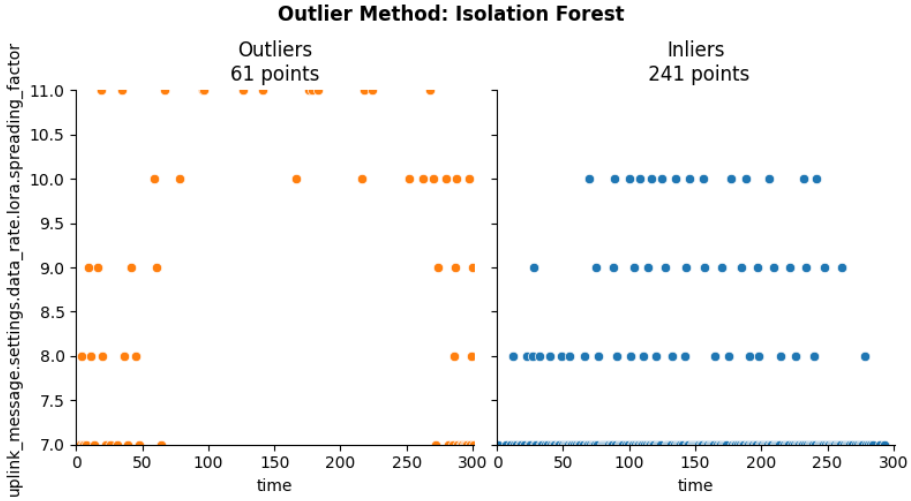


Figura 18. Resultado de la detección sobre la columna uplink_message_settings_data_rate.lora.spreading_factor en relación con time

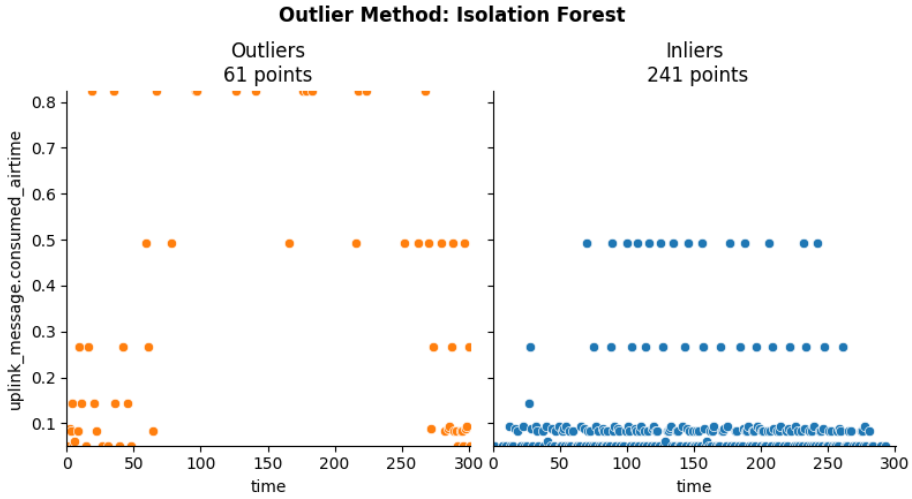


Figura 19. Resultado de la detección sobre la columna uplink_message.consumed_airtime en relación con time

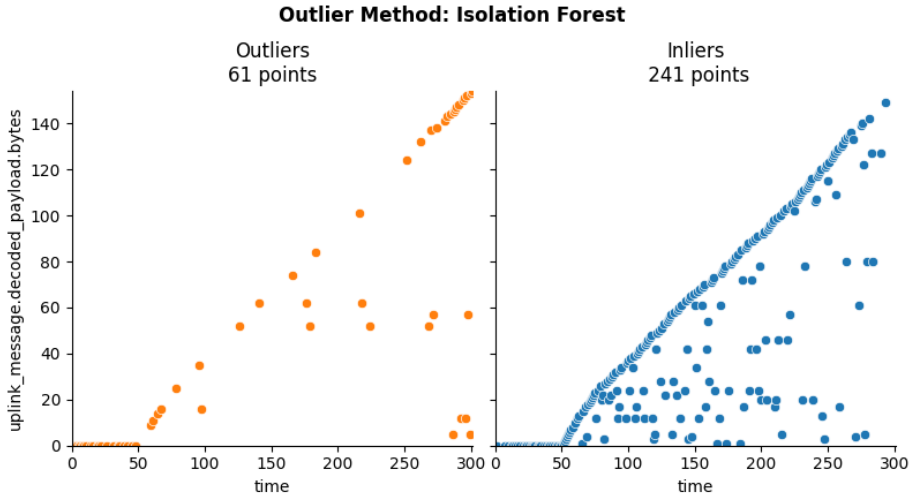


Figura 20. Resultado de la detección sobre la columna uplink_message.decoded_payload.bytes en relación con time

Para terminar, extraemos el modelo que utilizaremos para el resto de pruebas de Test y Validación, puesto que ya no se utilizará el método .fit(), sino que se implementará el modelo como si fueran casos de uso reales, mediante el método de .predict().

```
import pickle

modelo = 'model_IF.pk1'

with open(modelo , 'wb') as archivo:
    pickle.dump(model_IF, archivo)
```

Código 20. Fragmento de código que implementa la extracción del modelo

Repetiremos el proceso de leer el CSV, crear el array con los parámetros a considerar posibles anomalías y utilizar el método .predict() en ambas etapas, tanto para realizar las pruebas con el 20%, como la validación con el 10% restante.

7.6. Etapa D - Detección

Tras comprobar el correcto funcionamiento del modelo, adoptamos ciertas etapas desarrolladas anteriormente para poder recibir los datos y realizar detecciones en tiempo real.

```
[Actividad D-AP-CL]
      Definir conexión a BD
      Definir conexión a plataforma IoT - TTN
      Activar Listener MQTT

[Actividad D-AP-P]
      Cargar mensaje a JSON
      Control de datos de entrada nulos
      Generar objeto de datos para cada gateway (deserializado)
      Apilamos objeto en cola de DataFrame


[Actividad D-AP-RV]
      Renombrado de variables para los arrays desglosados

[Actividad D-AP-SC]
      Selección de características
      Eliminar valores innecesarios
      Tratamiento de valores nulos

[Actividad D-EX-IA]
      Cargar modelo
      Aplicar el modelo IA al flujo de datos

[Actividad D-EX-D]
      Decisión según umbral de anomalías
```

Código 21. Pseudocódigo de las acciones de la implementación de la etapa D

Para empezar, será necesario realizar la conexión y la lectura de los datos [D-AP-CL], la diferencia con el primer script realizado, es que para tener un código homogéneo y ejecutable en un sólo archivo, lo adaptamos a su versión en Python, dejando de lado JavaScript. Al igual que al inicio del proyecto, tendremos que conectarnos a la aplicación de TTN mediante unas credenciales concretas, y una vez establecida la conexión, finalmente obtenemos el mensaje, el cuál almacenamos en una variable con el método de json.loads(), de forma que sea legible la información recibida. [42] [43]

```
from numpy import NaN
import paho.mqtt.client as mqtt
import sys
import json
import random
import pandas as pd

# Credenciales de TTN
user = "ua-sensor@ttn"
password ="XXXXXXX.XXXXXXXXXXXXXXXXXX…"
public_tls_address = "eu1.cloud.thethings.network"
public_tls_address_port = 8883
```

```
def stop(client):
    client.disconnect()
    print("\nExit")
    sys.exit(0)

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("\nConexión establecida correctamente")
    else:
        print("\nFallo en la conexión")

def on_subscribe(client, userdata, mid, granted_qos):
    print("\nSubscribed with message id (mid) = " + str(mid))


def on_disconnect(client, userdata, rc):
    print("\nDisconnected with result code = " + str(rc))


client_id = f'python-mqtt-{random.randint(0, 1000)}'

mqttc = mqtt.Client(client_id)

mqttc.on_connect = on_connect
mqttc.on_subscribe = on_subscribe
mqttc.on_message = on_message
mqttc.on_disconnect = on_disconnect

mqttc.username_pw_set(user, password)

mqttc.tls_set()
mqttc.connect(public_tls_address, public_tls_address_port, 60)


if all_devices:
    mqttc.subscribe("#")
elif len(device_id) != 0:
    topic = "v3/" + user + "/devices/" + device_id + "/up"
    print("Subscribe to topic " + topic)
    mqttc.subscribe(topic)
else:
    print("Can not subscribe to any topic")
    stop(mqttc)

try:
    run = True
    mqttc.loop_forever()
except KeyboardInterrupt:
    stop(mqttc)
```

Código 22. Fragmento de código que implementa la conexión con TTN

A diferencia de cómo ocurría con el script ingestador de datos a la base de datos, en este paso configuraremos directamente el dataframe, sin necesidad de que previamente haya sido un objeto JSON, puesto que es nuestro objetivo final para que el modelo pueda comprender la información recibida. En el proceso de la creación del dataframe, se realizan algunos ajustes en la información recibida, como era el parseado de la información, la eliminación de arrays y el renombrado de variables [D-AP-P, D-AP-EA, D-AP-RV]. Recordemos que se crea un dataframe por cada gateway que recibe la información del sensor, por lo que la adecuación del dato se realiza en un bucle en el que se recorren los gateways que han recibido dicha información.

```
def on_message(client, userdata, message):
    print("\nMessage received on topic '" + message.topic)
    dt = pd.DataFrame()
    payload = json.loads(message.payload)

    if 'uplink_message' not in payload:
          payload['uplink_message'] = -1
    for rxm in payload['uplink_message']['rx_metadata']:
      timestamp = rxm['timestamp']
      if isinstance(timestamp, str) or isinstance(timestamp, int):
          rxm['timestamp'] = rxm['timestamp']
      else:
          rxm['timestamp'] = rxm['timestamp'].get('$numberLong')

      rx_metadata_dataframe = pd.DataFrame({
        "correlation_ids0": [payload['correlation_ids'][0]],
        "correlation_ids1": [payload['correlation_ids'][1]],
        "correlation_ids2": [payload['correlation_ids'][2]],
        "correlation_ids3": [payload['correlation_ids'][3]],
        "correlation_ids4": [payload['correlation_ids'][4]],
        "correlation_ids5": [payload['correlation_ids'][5]],
        "correlation_ids6": [payload['correlation_ids'][6]],
        "gateway_ids.gateway_id": [rxm['gateway_ids']['gateway_id']],
        "gateway_ids.eui": [rxm['gateway_ids']['eui']],
        "time": [rxm['time']],
        "timestamp": [rxm['timestamp']],
        "rssi": [rxm['rssi']],
        "channel_rssi": [rxm['channel_rssi']],
        "snr": [rxm['snr']],
        "location.latitude": [rxm['location']['latitude']],
        "location.longitude": [rxm['location']['longitude']],
        "location.altitude": [rxm['location']['altitude']],
        "location.source": [rxm['location']['source']]
      })

    dt = pd.concat([dt, rx_metadata_dataframe], axis=0)
```

Código 23. Fragmento de código que implementa la creación del DataFrame

Puesto que ya sabemos los parámetros que nos interesan para la detección, realizamos la selección de características, eliminando las que no aportan información útil, y aplicando el proceso de factorización a las restantes, de forma que convertimos la información categórica a numérica, siendo este el formato que precisa el modelo. [D-AP-SC]

```
def on_message(client, userdata, message):
    print("\nMessage received on topic '" + message.topic)
    dt['correlation_ids0'], correlation_ids0_c =
pd.factorize(dt['correlation_ids0'])
    dt['correlation_ids1'], correlation_ids1_c =
pd.factorize(dt['correlation_ids1'])
    dt['correlation_ids2'], correlation_ids2_c =
pd.factorize(dt['correlation_ids2'])
    dt['correlation_ids3'], correlation_ids3_c =
pd.factorize(dt['correlation_ids3'])
    dt['correlation_ids4'], correlation_ids4_c =
pd.factorize(dt['correlation_ids4'])
    dt['correlation_ids5'], correlation_ids5_c =
pd.factorize(dt['correlation_ids5'])
    dt['correlation_ids6'], correlation_ids6_c =
pd.factorize(dt['correlation_ids6'])
    dt['time'], time_c = pd.factorize(dt['time'])
    dt['location.source'], source_c =
pd.factorize(dt['location.source'])
    dt['uplink_token'], uplink_token_c =
pd.factorize(dt['uplink_token'])

    dt = dt.drop([
          'correlation_ids0',
          'correlation_ids1',
          'correlation_ids2',
          'correlation_ids3',
          'correlation_ids4',
          'correlation_ids5',
          'correlation_ids6',
          ], axis=1)

    x = str(dt['uplink_message.consumed_airtime'][0])[:-1]
    dt['uplink_message.consumed_airtime'] = float(x)
```

Código 24. Fragmento de código que implementa la factorización y eliminación de algunas columnas, además de la adecuación del valor uplink_message.consumed_airtime

Además de realizar la anterior transformación de los datos, se comprueba que no existan valores nulos durante la recepción, de forma que se rellena con un -1 los valores faltantes.

```
def on_message(client, userdata, message):
    print("\nMessage received on topic '" + message.topic)

    from sklearn.impute import SimpleImputer
    imputer = SimpleImputer(strategy='constant', fill_value=-1)

    attributes = ['received_at', 'uplink_message.session_key_id',
'uplink_message.frm_payload', 'uplink_message.decoded_payload.bytes',
```

```
'gateway_ids.gateway_id','time', 'timestamp', 'rssi', 'channel_rssi',
'snr', 'location.latitude', 'location.longitude',
'location.altitude', 'uplink_token', 'channel_index',
'uplink_message.settings.data_rate.lora.spreading_factor',
'uplink_message.received_at', 'uplink_message.consumed_airtime']


    imputer.fit(dt)
    dt = imputer.transform(dt)
    dt = pd.DataFrame(dt, columns = attributes)
```

Código 25. Fragmento de código que implementa el tratamiento de nulos

Finalmente, una vez tenemos la adecuación de los datos, pasamos a aplicar el modelo de IA [D-EX-IA], que realiza una lectura en tiempo real de los datos entrantes, y toma una decisión de si se considera una anomalía o no [D-EX-D], de forma que se genera un sistema de alertas para el control de las infraestructuras IoT.

```
def on_message(client, userdata, message):
    print("\nMessage received on topic '" + message.topic)
    anomaly_inputs = ['received_at',
'uplink_message.session_key_id'...]

    import pickle
    ruta_modelo = '/content/model_IF.pk1'

    with open(ruta_modelo, 'rb') as archivo:
        modelo_cargado = pickle.load(archivo)

    prediction = modelo_cargado.predict(dt[anomaly_inputs])
    dt['anomaly'] = prediction

    umbral = 0.5  # Define el umbral adecuado para tu caso
    if prediction > umbral:
        print("¡Anomalía detectada!")
    else:
        print("No se detectó ninguna anomalía")
```

Código 26. Fragmento de código que implementa el modelo, junto con la detección en tiempo

8. Aplicación sobre un escenario real

El desarrollo de este trabajo persigue no solo el planteamiento teórico sino también su aplicación práctica, y para ello se ha utilizado el proyecto Smart University como ejemplo de aplicación. El proyecto Smart University es un proyecto coordinado por la Universidad de Alicante para el desarrollo de un sistema que integre y centralice toda la información procedente de los distintos tipos de sensorización que puede generarse en la Universidad. Esta información podrá explotarse a través de herramientas de visualización, análisis y procesamiento mediante técnicas de IA con el objetivo de generar información que facilite la toma de decisiones, de manera que la Universidad sea capaz de hacer una gestión más eficiente de los recursos, infraestructuras y servicios. La propuesta es instanciar el modelo de ADS junto con el software implementado para generar un módulo ADS para el proyecto Smart University. Este módulo será independiente de la plataforma, pero le dotará de la capacidad de detectar anomalías para evitar así que, datos incorrectos o malintencionados, dañen la funcionalidad del sistema. La propuesta será por tanto general un módulo ADS que se inserte justo antes de que los datos IoT lleguen a la plataforma. Smart University utiliza TTN como plataforma IoT, aunque podría haber sido cualquier otra. Este es el esquema de la solución propuesta.

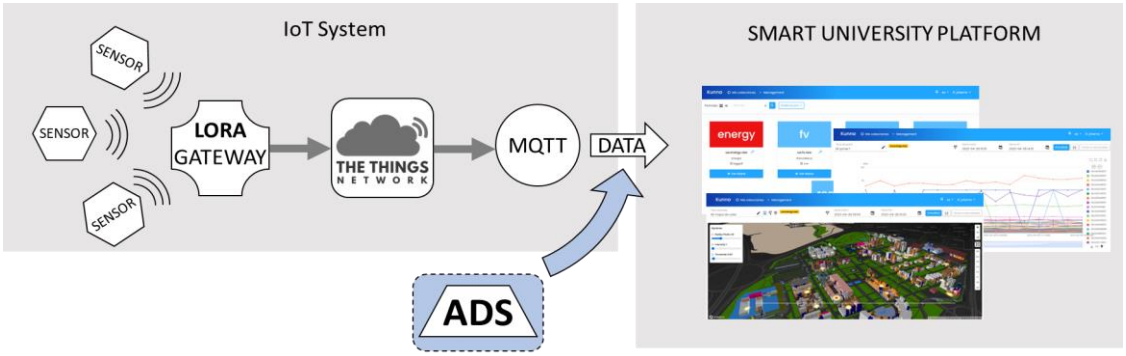


Figura 21. Diagrama de la solución propuesta para Smart University

El sistema de sensorización está formado por dispositivos LORA multisensores (captan y transmiten la medición de CO2, temperatura, presencia, ruido, humedad, composición gases y otros valores) que emiten sus señales periódicamente a la red LORA. Los gateways utilizan también como plataforma IoT la plataforma The Things Network (TTN), ya que al estar basada en un sistema OpenSource y tener detrás una comunidad, es muy adecuada para este tipo de proyectos. Como se puede ver en la figura 21, el control de la infraestructura IoT queda totalmente fuera del control de la plataforma Smart University, de forma que cuando suceden anomalías en dicha infraestructura, la plataforma registra datos que pueden ser incoherentes o erróneos, llegando a producir efectos indeseados, como la parada del sistema de climatización o el cierre de una sala. Para evitar este

efecto, hemos añadido una instancia del modelo ADS en la adquisición de datos de la plataforma, de forma que se puedan recibir los datos saneados, o bien avisos y alarmas para conocer que una anomalía se está produciendo. Es importante resaltar de nuevo que el sistema no detecta valores peligrosos, por ejemplo, como el caso del CO2, sino que las infraestructuras IoT están produciendo datos que pueden ser incorrectos (aunque la medición de CO2 sea un valor normal).

Para crear la instancia del modelo ADS en nuestro sistema, se deben concretar cada uno de los procesos descritos. La mayoría de ellos son acciones simples que solo requieren utilizar las tecnologías concretas para realizar su función. Debido a que la implementación se ha realizado en base a TTN, las adaptaciones necesarias son pocas y realmente el ADS puede casi implementarse de inmediato y ponerse en marcha. La ventaja de haber utilizado TTN para consumir el prototipo base es que podemos utilizar el entrenamiento ya formulado para nuestro sistema de Smart University, ya que la base de infraestructuras es la misma, y por tanto el procesamiento de datos, el estudio de relevancia y el entrenamiento de la IA es válido. Por tanto, se lanza a ejecutar el proceso de detección.

8.1. Resultados de la detección

Al ejecutar la detección sobre el sistema de sensorización se han detectado algunas anomalías. Una vez más recordamos que no se detecta la anomalía concreta ocurrida, sino el hecho de que "algo anómalo está ocurriendo" siendo necesario un análisis posterior. Pasamos a contar las anomalías detectadas. El resultado al ejecutar nuestro módulo instanciado en el proyecto Smart University, sería algo similar a la siguiente figura (en realidad solo se muestran las anomalías cuando está en producción):

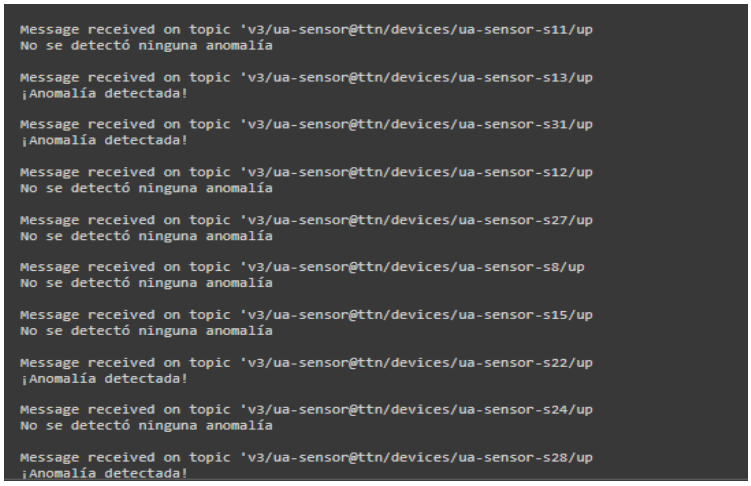


Figura 22. Salida por pantalla de la detección de anomalías en tiempo real.

8.1.1. Caída en la potencia de la señal

Una de las primeras anomalías que detectó el sistema fue el cambio repentino de potencia de la señal de los sensores. Esta cuestión en un dispositivo que es estable en su potencia no era habitual, por lo que el sistema marcó una anomalía en ese instante, y tras el análisis de esa detección, se pudo localizar que había habido un cambio en la posición de la antena que provocó un apantallamiento de la señal. La figura 23 muestra un instante en el que se puede apreciar este cambio repentino de potencia de señal detectado por el ADS.

Figura 23. Cambio brusco de potencia de la señal

8.1.2. Detección de gateways no habituales

Otra de las anomalías que suele detectar el sistema, es cuando un gateway diferente a los usuales recibe el paquete de datos. Al ser algo fuera de lo común, se establecen dos posibilidades, la sonda ha cambiado de ubicación y con ella los gateways a su alrededor, o un nuevo gateway se ha activado dentro de su rango, sin ser uno de los que tienen configurados desde el grupo de Smart University. Tras el análisis del paquete tras la detección de la anomalía, mediante la herramienta de TTN Mapper [44], nos dimos cuenta de que una de las sondas había sido no sólo movida de una habitación, sino sacada del área de la universidad, sin que nadie avisara de que se llevarían una sonda de su lugar.


```
  "received_at": "2023-05-25T08:20:19.470798102Z",
  "uplink_message": {
    "session_key_id": "AYhRpFbAR2yqyuWPHM5L4g==",
    "f_port": 1,
    "f_cnt": 121,
    "frm_payload": "AP9/RwD/f0cA/39HAP9/RwD/f0cA/39HAGDMRA==",
```

```
  "decoded_payload": {
    "bytes": [0, 255, 127,71,0,...]
  },
  "rx_metadata": [
    {
      "gateway_ids": {
        "gateway_id": "molukas-daimus",
        "eui": "A84041FFFF211964"
      },
      "time": "1980-01-06T00:00:49.431Z",
      "timestamp": 1987300995,
      "rssi": -107,
      "channel_rssi": -107,
      "snr": 2.2,
      "location": {
        "latitude": 38.97518313182293,
        "longitude": -0.1409554481506348,
        "altitude": 30,
        "source": "SOURCE_REGISTRY"
      },
```


Código 27. Fragmento del mensaje son un gateway_id no habitual

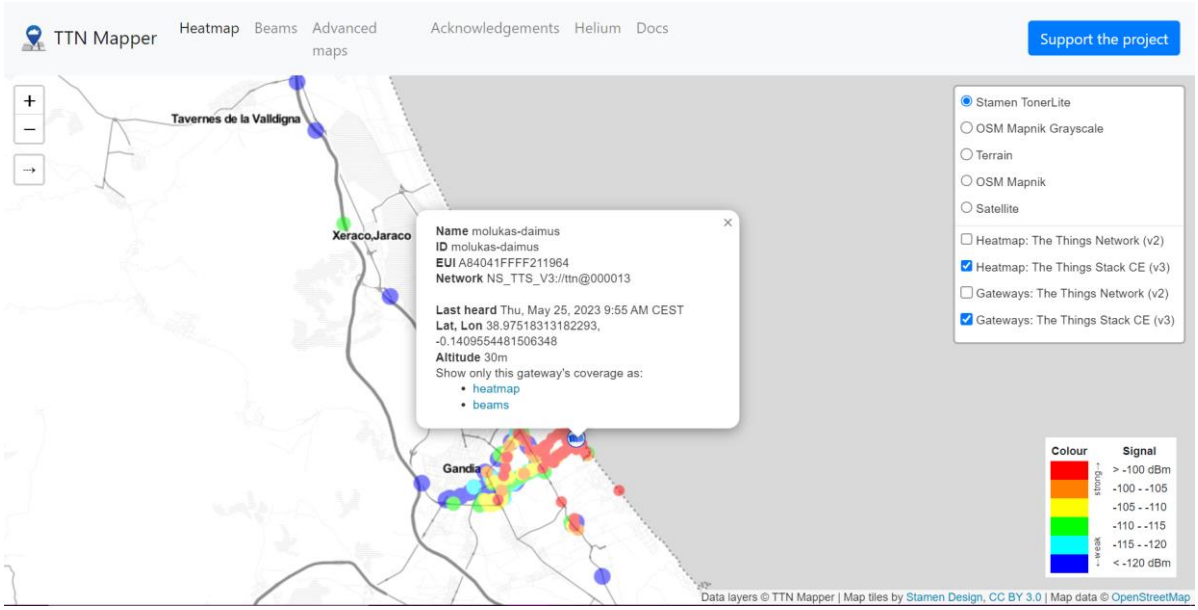


Figura 24. Localización del gateway detectado desde TTN Mapper

8.1.3. Recepción de mensajes sin paquete de datos

Por último, uno de los ejemplos más útiles es cuando una sonda deja de enviar datos a la plataforma de Kunna. Este caso se da cuando un sensor no funciona correctamente, pero la placa a la que está conectada al sensor, junto con su antena LoRa, sí que se encuentran en buen estado, de manera que los paquetes llegan a TTN, pero no hay un mensaje codificado en su interior. Un ejemplo del mensaje recibido en la plataforma de TTN es el siguiente:


```
"uplink_message": {
      "session_key_id": "AYhLfwAKaF77V51mXJozVg==",
      "f_cnt": 1206,
      "rx_metadata": [
        {
          "gateway_ids": {
            "gateway_id": "smartua-iot-r01",
            "eui": "3133303726006500"
          },
          "time": "2023-05-23T13:38:04.234059Z",
          "timestamp": 2645589695,
          "rssi": -95,
          "channel_rssi": -95,
          "snr": 6.75,
          "location": {
            "latitude": 38.3850472441428,
            "longitude": -0.5168026685714723,
            "altitude": 20,
            "source": "SOURCE_REGISTRY"
          },
          "uplink_token":
"Ch0KGwoPc21hcnR1YS1pb3QtcjAxEggxMzA3JgBlABC//cHtCRoMCNjCuKMGEMjx+doC
IJi0vMv/h5EC",
          "channel_index": 5,
          "received_at": "2023-05-23T13:38:16.700461759Z"
        }
```


Código 28. Fragmento del mensaje sin el campo decoded_payload": { "bytes": [ ] }

Figura 25. Muestra del flujo de envío de datos hasta su corte repentino

Como se puede observar en la imagen, a partir de las 13:38 aproximadamente, el sensor dejó de emitir valores cuantitativos, ya fueran C02, GPS, o su correspondiente sensor en ese dispositivo.

9. Resultados

A través de este proyecto se han generado varios resultados de interés.

En primer lugar, se ha propuesto un modelo conceptual para el desarrollo de ADS sobre infraestructura IoT. Una de las características más destacadas de estas infraestructuras es precisamente no tener el control sobre las mismas, utilizando servicios e incluso infraestructuras intermedias abiertas (como en nuestro caso TTN). Esto hace que nos enfrentemos al desconocimiento de información, que en infraestructura de red tradicional es muy útil para detectar anomalías, como IPs, MACs o protocolos. Nuestro modelo lo que pretende es la creación de detectores de amplio espectro, es decir, detectar que hay paquetes que no son correctos, para posteriormente analizarlos y extraer la anomalía. Para ello se proponen los procesos y el flujo de trabajo para su creación.

A partir del modelo desarrollado, se ha realizado una implementación basada en infraestructura IoT The Thing Networks, que a su vez utiliza el software open source The Thing Stack como base. Este software es ampliamente utilizado por la comunidad IoT. La implementación creada sirve de base o plantillas para crear instancias para otros sistemas. Lo importante en este código es la ejecución de los procesos definidos en el modelo. Será necesaria su adaptación para cada proyecto, pero la base es la misma, aunque haya procesos que incluso no contengan nada porque de base la tecnología utilizada igual ya resuelve ese aspecto. Además, aprovechando la implementación realizada, se ha instanciado un módulo para el proyecto Smart University, que se basa en TTN también. Esta instancia del módulo ha permitido poner en marcha un ADS sobre la plataforma que desde el primer momento ha detectado anomalías que antes eran filtradas a mano.

Por último, este trabajo ha sido presentado y aceptado en el congreso *International Conference on IoT Based Control Networks and Intelligent Systems - ICICNIS 2023*, bajo el título *Proposal of a general model for creation of anomaly detection systems in IoT infrastructures*, y cuyas actas del congreso se publican por la editorial Springer.

10. Conclusiones y trabajo futuro

Para finalizar con este proyecto, se pueden extraer las siguientes conclusiones.

En primer lugar, el poner en práctica el proceso para la creación del modelo, ha cumplido con todos los objetivos propuestos en un inicio, puesto que se ha logrado obtener una metodología de adquisición, procesado y utilización de los datos, que puede ser genérica para cualquier tipo de infraestructura en la que se quiera implementar, de manera que ha completado con éxito la fase correspondiente a la obtención de los datos y creación del dataset especializado para la infraestructura del proyecto.

Por otra parte, al elegir a conciencia los parámetros que se iban a considerar anomalías tras el estudio de correlación, unido a la propia experiencia adquirida al trabajar con los paquetes entrantes en la plataforma de TTN, se centra y acota mucho la dimensión en la que se pueden detectar esos valores inesperados, siendo este un factor que ha ayudado a mejorar la calidad de las anomalías detectadas.

Por último, la propia implementación del modelo en un sistema en tiempo real para la plataforma de Smart University, de manera que se han podido detectar posibles fallos en las infraestructuras y situaciones inesperadas en cuanto al comportamiento de los sensores, ha permitido poder tomar medidas al respecto, siento un punto de apoyo importante para la monitorización general del proyecto. Es decir, la propuesta es válida, es útil y está en marcha.

En cuanto a las posibles mejoras y trabajos futuros, se va a mejorar el rendimiento del modelo obtenido mediante el entrenamiento con más datos, pero debido al tiempo de ejecución durante el procesamiento que se llevaba al utilizar un archivo CSV con cientos de miles de valores, se decidió no utilizar archivos tan grandes desde el principio, ya que era un retraso considerable en la planificación del proyecto.

Por otra parte, se desea implementar diferentes módulos de Machine Learning para la detección de anomalías tanto en un sentido más amplio, como en la identificación de estás. Es decir, por un lado, buscaremos la comparación de distintos algoritmos que permitan detectar más situaciones de anomalías, para evitar que puedan “colarse” paquetes incorrectos. Incluso se plantea el poder tener varios algoritmos procesando en paralelo y generando una agregación de sus resultados. Pero por otro lado también se busca la identificación. Una vez que podemos discriminar paquetes y ponerlos en un estado de observación, a través de otros algoritmos especializados podemos generar una catalogación que identifique anomalías de distintos tipos como puede ser: fallos de

transmisión, sustracción de dispositivos, mal funcionamiento del dispositivo, implantación de dispositivos inesperada (un problema muy común en IoT, el que parezcan sensores que no se sabe de donde surgen). Entonces la encadenación de sistemas de detección de anomalías con la identificación de anomalías puede ayudar a tener sistemas flexibles y modulares. Por ejemplo, se puede plantear una infraestructura donde nuestro ADS detecte anomalías, una vez apartados estos paquetes del flujo de datos, podemos probar con distintos módulos de identificación hasta que demos con aquellos que identifican el problema. Esto hace que se pueda construir sistemas más eficientes ya que sólo incorporaría los módulos necesarios, pudiendo automatizar incluso que el sistema busque, entre una posible colección de módulos de identificación, aquellos que requiere.

Referencias

1. Canva. (s. f.). Free Design Tool: Presentations, Video, Social Media | Canva. https://www.canva.com/
2. Fundación Telefónica. (2017). Recuperado el Marzo de 2020, de Smart Cities: un primer paso hacia la internet de las cosas. (s. f.). Google Books. https://books.google.es/books?hl=es&lr=&id=wZLmCgAAQBAJ&oi=fnd&pg=PA1&dq%20=Smart+City+telefonica&ots=Y_c0FeW_FL&sig=w4Xj7TkJ5sVsLeYDi0hPhLtiWQs#v%20=onepage&q&f=false
3. Jesús, G. A. M. (2020). Los servicios de una ciudad inteligente : Smart Cities. http://hdl.handle.net/11531/41643
4. Khan, M. A., & Salah, K. (2018). IoT security: Review, blockchain solutions, and open challenges. Future generation computer systems, 82, 395-411.
5. Oh, S. R., & Kim, Y. G. (2017, February). Security requirements analysis for the IoT. In 2017 International Conference on Platform Technology and Service (PlatCon) (pp. 1-6). IEEE.
6. Perwej, Y., Parwej, F., Hassan, M. M. M., & Akhtar, N. (2019). The internet-of-things (IoT) security: A technological perspective and review. International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN, 2456-3307.
7. OWASP Top 10 de vulnerabilidades del Internet de las cosas (IoT) – FGC Security. (s. f.). https://www.fgcsecurity.com/owasp-top-10-de-vulnerabilidades-del-internet-de-las-cosas-iot/
8. Pedro, C. P. (s. f.). Sistema automatizado de detección de ataques en redes IoT - Archivo Digital UPM. https://oa.upm.es/70611/
9. Fundamentals of (IPS) Intrusion Prevention System. (2022, 25 marzo). [Vídeo]. Cisco. https://www.cisco.com/c/es_mx/products/security/what-is-network-security.html
10. Kiravuo, T., Särelä, M., & Manner, J. (2013). A Survey of Ethernet LAN Security. IEEE Communications Surveys and Tutorials, 15(3), 1477-1491. https://doi.org/10.1109/surv.2012.121112.00190

11. Protocolos Ethernettip según INCIBE. https://www.incibe.es/incibe-cert/blog/protocolo-ethernetip-analizando-sus-comunicaciones-y-medidas-seguridad

12. Conceptos básicos que te ayudarán a entender LoRa y LoRaWAN en minutos. (2023, 27 abril). Becolve digital. https://becolve.com/blog/conceptos-tecnicos-basicos-que-te-ayudaran-a-entender-lora-y-lorawan-low-power-wide-area-network-en-pocos-minutos/#:~:text=De%20forma%20an%C3%A1loga%20a%20una,dispositivos%20en%20la%20red%20Ethernet

13. De Catalunya, U. O. (2017, 13 junio). Estudio de la arquitectura y el nivel de desarrollo de la red LoRaWAN y de los dispositivos LoRa. http://hdl.handle.net/10609/64365

14. Lavric, A., & Petrariu, A. (2018). LoRaWAN communication protocol: The new era of IoT. https://doi.org/10.1109/daas.2018.8396074

15. Singhal, D. & Swarup, K. (2011). Electricity price forecasting using artificial neural networks. International Journal of Electrical Power & Energy Systems. Elsevier, 33 (3), pp. 550-555.

16. M, E. S., Serna, A., & Acevedo, E. (2017). Principios y características de las redes neuronales artificiales. ResearchGate. https://www.researchgate.net/publication/331498946_Principios_y_caracteristicas_de_las_redes_neuronales_artificiales

17. Fernández, M. (2022, 17 noviembre). Inteligencia Artificial: un nuevo hito para la Ciberseguridad. enzyme.biz. https://enzyme.biz/blog/inteligencia-artificial-nuevo-hito-ciberseguridad

18. Common Intrusion Detection Framework (CIDF). [en línea]. Actualizado en septiembre 1999 [consultado en mayo, 2003]. Disponible en <http://www. isi.edu/gost/cidf/>

19. Gómez, D. G. (2003). Sistemas de Detección de Intrusiones. Sistemas De Detección De Intrusiones.

20. Vista de Sistema de detección de intrusos en redes corporativas. (s. f.). https://revistas.utp.edu.co/index.php/revistaciencia/article/view/9105/10161

21. Ramírez, H. (2022). El sistema de detección de intrusiones (IDS). Grupo Atico34. https://protecciondatos-lopd.com/empresas/sistema-deteccion-intrusiones-ids/

22. Méndez, J. R., Riverola, F. F., Díaz, F., & Corchado, J. M. (2007). Sistemas inteligentes para la detección y filtrado de correo spam: una revisión. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 11(34), 63-81.

23. Adrián, L. P. (2023). Desarrollo de un modelo de detección de URLs maliciosos usando aprendizaje automático supervisado. https://hdl.handle.net/10669/88464

24. colaboradores de Wikipedia. (2022). Árbol de decisión. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n

25. colaboradores de Wikipedia. (2023). Random forest. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Random_forest

26. Enrique, L. M. F. (2021). Detección de transacciones fraudulentas en tarjetas de crédito mediante el uso de modelos de Machine Learning. Universidad de los Andes. http://hdl.handle.net/1992/53571

27. https://www.thethingsindustries.com/docs/reference/ttn/. (s. f.-b).

28. Gestiona los proyectos de tu equipo desde cualquier lugar | Trello. (s. f.). https://trello.com/es

29. Clockify. (2017, 1 septiembre). Clockify - Software de control del tiempo GRATIS. https://clockify.me/es/

30. Hawkins, D. M. (1980). Identification of Outliers. En Springer eBooks. https://doi.org/10.1007/978-94-015-3994-4

31. The Things Network. (s. f.). The Things Network. https://www.thethingsnetwork.org/

32. Node.js. (s. f.). Node.js. https://nodejs.org/en

33. MongoDB. (s. f.). MongoDB: The Developer Data Platform. https://www.mongodb.com/

34. Wikipedia contributors. (2023). MQTT. Wikipedia. https://en.wikipedia.org/wiki/MQTT

35. colaboradores de Wikipedia. (2023b). JSON. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/JSON

36. json — Codificador y decodificador JSON. (s. f.). Python documentation. https://docs.python.org/es/3/library/json.html

37. colaboradores de Wikipedia. (2023a). Pandas (software). Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Pandas_(software)

38. csv — Lectura y escritura de archivos CSV. (s. f.). Python documentation. https://docs.python.org/es/3/library/csv.html

39. seaborn: statistical data visualization — seaborn 0.12.2 documentation. (s. f.). https://seaborn.pydata.org/

40. colaboradores de Wikipedia. (2022b). Matplotlib. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Matplotlib

41. colaboradores de Wikipedia. (2020). Scikit-learn. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Scikit-learn

42. colaboradores de Wikipedia. (2023b). NumPy. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/NumPy

43. paho-mqtt. (2021, 21 octubre). PyPI. https://pypi.org/project/paho-mqtt/

44. Meijers, J. (s. f.). TTN Coverage. TTN Mapper. https://ttnmapper.org/heatmap/

45. Xiao, L., Wan, X., Lu, X., Zhang, Y., & Wu, D. (2018). IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?. IEEE Signal Processing Magazine, 35(5), 41-49.

46. Dorsemaine, B., Gaulier, J. P., Wary, J. P., Kheir, N., & Urien, P. (2015, September). Internet of things: a definition & taxonomy. In *2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies* (pp. 72-77). IEEE.

47. Rajendran, G., Nivash, R. R., Parthy, P. P., & Balamurugan, S. (2019, October). Modern security threats in the Internet of Things (IoT): Attacks and Countermeasures. In *2019 International Carnahan Conference on Security Technology (ICCST)* (pp. 1-6). IEEE.

48. Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (IoT) security. IEEE Communications Surveys & Tutorials, 22(3), 1646-1685.

49. Rojas, B. S. C., Rodríguez, C. U. C., Osorio, D. J. E., & Bello, Y. T. G. (2020). Redes neuronales artificiales y estado del arte aplicado en la ciberseguridad. *Revista Matices Tecnológicos*, *12*, 58-63.

50. Matich, D. J. (2001). Redes Neuronales: Conceptos básicos y aplicaciones. *Universidad Tecnológica Nacional, México*, *41*, 12-16.

51. Khoussainov, R., & Patel, A. (2000). LAN security: problems and solutions for Ethernet networks. *Computer Standards & Interfaces*, *22*(3), 191-202.

52. Ordóñez Monfort, I. (2017). Estudio de la arquitectura y el nivel de desarrollo de la red LoRaWAN y de los dispositivos LoRa.

53. Joancomartí, J. H., Alfaro, J. G., & Tornil, X. P. (2004). Aspectos avanzados de seguridad en redes. *UOC Formación de Posgrado*.

54. Castro, F. R. (1997). *Seguridad en redes locales de datos: contribución a la seguridad de redes 802.3/Ethernet extendidas* (Doctoral dissertation, Universitat Politècnica de Catalunya (UPC)).

55. Lavric, A., & Petrariu, A. I. (2018, May). LoRaWAN communication protocol: The new era of IoT. In *2018 International Conference on Development and Application Systems (DAS)* (pp. 74-77). IEEE.

56. Conceptos básicos que te ayudarán a entender LoRa y LoRaWAN en minutos. (2023b, abril 27). Becolve digital. https://becolve.com/blog/conceptos-tecnicos-basicos-que-te-ayudaran-a-entender-lora-y-lorawan-low-power-wide-area-network-en-pocos-minutos/#:~:text=De%20forma%20an%C3%A1loga%20a%20una,dispositivos%20en%20la%20red%20Ethernet.

57. Yagual, D. I. Q., Yagual, C. C., & Suárez, I. C. (2022). Una revisión del aprendizaje profundo aplicado a la ciberseguridad. *Revista Científica y Tecnológica UPSE*, *9*(1), 57-65.

58. Martinez, G. R. S., Ocampo, C. A., & Bermúdez, Y. V. C. (2017). Sistema de detección de intrusos en redes corporativas. *Scientia et technica*, *22*(1), 60-68.

59. Muñoz García, J. A., & Amón Uribe, I. (2013). Técnicas para detección de outliers multivariantes. *Revista en telecomunicaciones e informática*.

60. Puertos Amat, A. (2022). Análisis de las tendencias del mundo de los videojuegos en Twitter según la opinión del usuario.

61. Pérez Sifre, J. (2020). IDS de red para la detección de ataques sobre SSH y FTP.

62. Pérez, D. M., Benavent-Lledo, M., Azorin-Lopez, J., Marcos-Jorquera, D., & Garcia-Rodriguez, J. (2022). Anomaly detection and virtual reality visualisation in supercomputers.

63. Pérez González, G. A. (2021). Deteccion de transacciones fraudulentas en tarjetas de crédito mediante el uso de modelos de Machine Learning.

64. M, E. S., Serna, A., & Acevedo, E. (2017b). Principios y características de las redes neuronales artificiales. ResearchGate. https://www.researchgate.net/publication/331498946_Principios_y_caracteristicas_de_las_redes_neuronales_artificiales

65. Ramírez, H. (2022b). El sistema de detección de intrusiones (IDS). Grupo Atico34. https://protecciondatos-lopd.com/empresas/sistema-deteccion-intrusiones-ids/

66. Serna, E. (2019). Desarrollo e innovación en ingeniería. Medellín, Antioquia, 54-57.

67. OWASP Internet of Things Project - OWASP. (s. f.). https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project

68. Mora Gimeno, F. J. (2010). Modelo para la integración de técnicas heterogéneas de detección de intrusos en sistemas distribuidos. Universidad de Alicante.

69. Ávila Pérez , . M. (2020). Implementación de un IDS. Gestión Competitividad E Innovación, 8(1), 11-23. Recuperado a partir de https://pca.edu.co/editorial/revistas/index.php/gci/article/view/91

70. Marktab. (2023, 10 febrero). Preparación de datos para ML Studio (clásico) - Azure Architecture Center. Microsoft Learn. https://learn.microsoft.com/es-es/azure/architecture/data-science-process/prepare-data

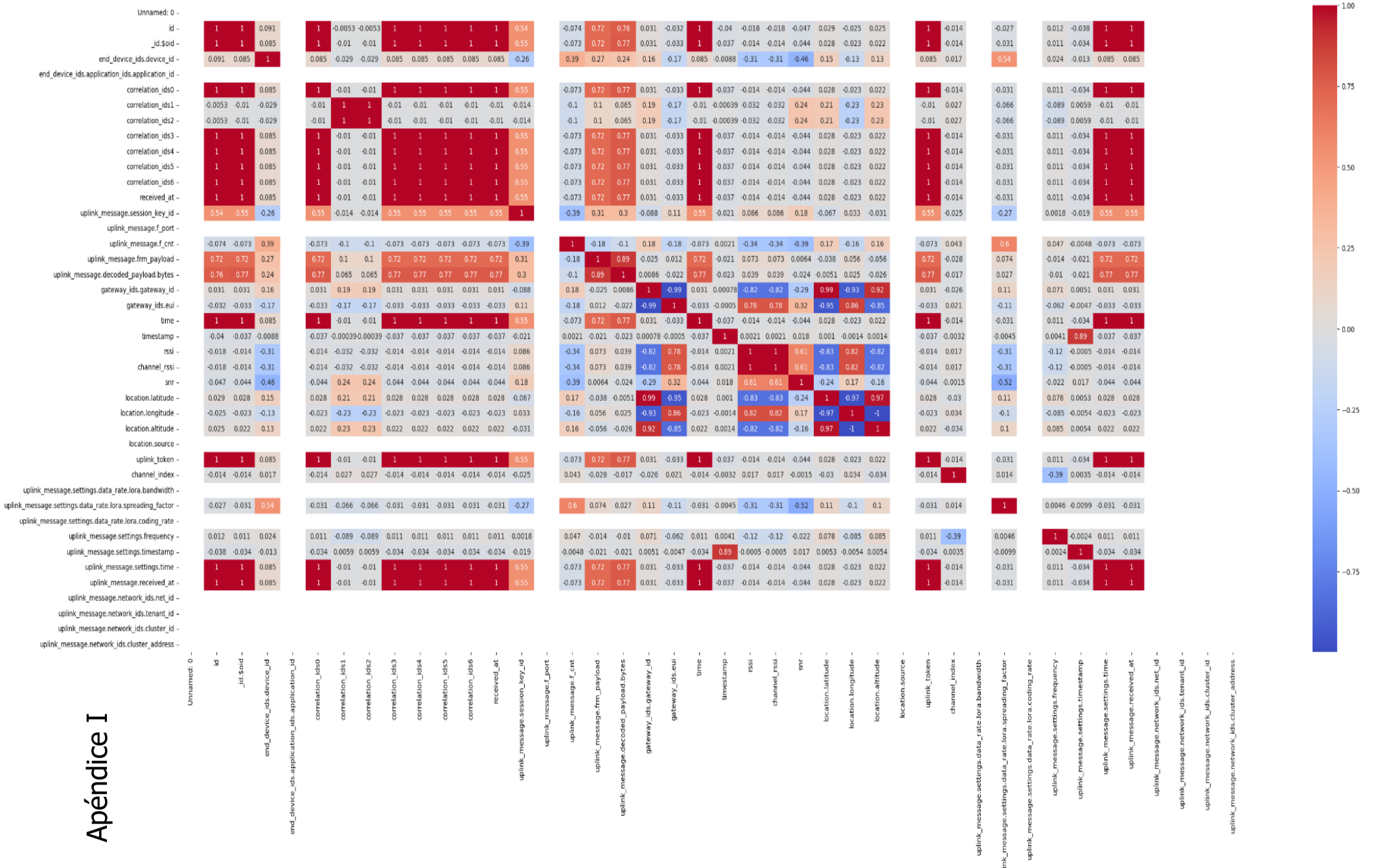
71. Generador APA. (2022, 23 mayo). Formato APA con el Generador APA de Scribbr. https://www.scribbr.es/detector-de-plagio/generador-apa/

72. Iglesias, I. A. ANÁLISIS Y DETECCIÓN DE ANOMALÍAS USANDO TÉCNICAS DE DEEP LEARNING: CASO DE ESTUDIO EN COTIZACIONES DE PFIZER Y MODERNA.

73. *Kunna*. (s. f.). https://smartua.es/

74. Rojas, E. M. (2020). Machine Learning: análisis de lenguajes de programación y herramientas para desarrollo. Revista Ibérica de Sistemas e Tecnologias de Informação, (E28), 586-599.

Apéndice I


Unnamed: 0
id
_id.$oid
end_device_ids.device_id
end_device_ids.application_ids.application_id
correlation_ids0
correlation_ids1
correlation_ids2
correlation_ids3
correlation_ids4
correlation_ids5
correlation_ids6
received_at
uplink_message.session_key_id
uplink_message.f_port
uplink_message.f_cnt
uplink_message.frm_payload
uplink_message.decoded_payload.bytes
gateway_ids.gateway_id
gateway_ids.eui
time
timestamp
rssi
channel_rssi
snr
location.latitude
location.longitude
location.altitude
location.source
uplink_token
channel_index
uplink_message.settings.data_rate.lora.bandwidth
uplink_message.settings.data_rate.lora.spreading_factor
uplink_message.settings.data_rate.lora.coding_rate
uplink_message.settings.frequency
uplink_message.settings.timestamp
uplink_message.settings.time
uplink_message.received_at
uplink_message.network_ids.net_id
uplink_message.network_ids.tenant_id
uplink_message.network_ids.cluster_id
uplink_message.network_ids.cluster_address
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75