UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Creación de demos técnica y documentación asociada, para la detección/clasificación de materiales, utilizando un sensor radar a 60GHz

Presentado por Martín Encabo Contreras
en Universidad de Burgos — 16 de enero de 2023
Tutores:
José Francisco Díez Pastor
Pedro Latorre Carmona

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. José Francisco Díez Pastor y D. Pedro Latorre Carmona, profesores del departamento de ingeniería informática, área de lenguajes y sistemas informáticos.

Exponen:

Que el alumno D. Martín Encabo Contreras, con DNI 72897369L, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Creación de demos técnica y documentación asociada, para la detección/clasificación de materiales, utilizando un sensor radar a 60GHz.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 16 de enero de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. José Francisco Díez Pastor	D. Pedro Latorre Carmona

Resumen

Hoy en día, la tecnología basada en radar está siendo utilizada principalmente en ámbitos como el tráfico aéreo o submarino. Existen otros campos poco explorados y que pueden ser más cotidianos como la identificación de objetos en base a las propiedades del material que lo constituye.

El objetivo principal del proyecto es clasificar una serie de objetos comparando su material de fabricación mediante un radar de 60 GHz que utilizará un modelo clasificador entrenado para tal fin.

Para recopilar la información necesaria de cada objeto, se hará uso de la librería proporcionada por el fabricante, con el objetivo de poder conseguir una serie de lecturas, las cuales deberán brindar la suficiente información de todos los objetos sometidos al estudio. Se propone un análisis con un registro de treinta materiales considerando diez lecturas generadas de cada material.

Dado que es necesario realizar un análisis de los datos obtenidos por medio del radar, se propone implementar técnicas de aprendizaje automático para poder reconocer los diferentes materiales a estudiar. Para ello, se decide entrenar varios tipos de clasificadores para generar un modelo capaz de diferenciar entre los distintos materiales.

Descriptores

Aprendizaje Automático, Clasificador, Random Forest, k-NN, SVM, auto-sklearn, TabPFN, Reconocimiento, Python, Sensor, Radar, Acconeer.

Abstract

Today, radar-based technology is mainly being used in areas such as air traffic or underwater. There are other, less explored fields that may be more everyday, such as the identification of objects based on the properties of their constituent material.

The main objective of the project is to classify a series of objects by comparing their material of manufacture using a 60 GHz radar using a classifier model trained for this purpose.

In order to collect the necessary information for each object, use will be made of the library provided by the manufacturer, with the aim of obtaining a series of readings, which should provide sufficient information for all the objects under study. An analysis is proposed with a record of thirty materials considering ten readings generated from each material.

Given that it is necessary to analyse the data obtained from the radar, it is proposed to implement automatic learning techniques to be able to recognise the different materials to be studied. To this end, it is decided to train various types of classifiers to generate a model capable of differentiating between different materials.

Keywords

Machine Learning, Classifier, Random Forest, k-NN, SVM, auto-sklearn, TabPFN, Recognition, Python, Sensor, Radar, Acconeer.

Índice general

Índice de figuras

Índice de tablas

Introducción

La tecnología de radar existe desde la década de 1930 de la mano de Watson-Watt. El término radar proviene del acrónimo inglés RAdio Detection And Ranging. Aplicaciones típicas de los radares de radiofrecuencia incluyen la medida de distancias, altitudes, direcciones y velocidades de objetos. Un ejemplo muy conocido de uso es para «navegación de barcos».

Durante los últimos años han surgido nuevas áreas de aplicación para este tipo de radares. Actualmente se están empleando en monitorizar signos vitales, reconocimiento de gestos, etc. Su creciente aplicación se explica, en parte, por la disminución en su coste de adquisición, por (también, en parte) su implantación en industrias como la del automóvil, convirtiendo estos dispositivos en una opción atractiva en una amplia gama de aplicaciones de bajo coste.

En el presente proyecto se ha documentado, implementado y demostrado el uso de un radar de 60GHz fabricado por *Acconeer*, para el reconocimiento de diferentes materiales u objetos. Mediante el uso de métodos de aprendizaje, se obtendrán diferentes características de tres tipos de materiales, que permitirán clasificarlos de forma automática.

Objetivos del proyecto

El objetivo del proyecto es el desarrollo de una aplicación capaz de implementar la tecnología del radar junto con un modelo clasificador capaz de establecer una comparación entre los diferentes tipos de materiales que se estudiarán.

2.1. Objetivos generales

Los objetivos generales del proyecto se pueden describir en los siguientes puntos:

- El principal, desarrollar una aplicación capaz de comunicarse con el radar y realizar lecturas de objetos para identificar distintos materiales.
- Documentar los diferentes elementos que componen el ensamblado y poner en funcionamiento un radar de 60 GHz fabricado por *Acconeer* para que sea capaz de identificar, mediante un clasificador, el tipo de material al que pertenece un objeto.
- Crear un procedimiento capaz de extraer determinadas características de las lecturas de los objetos para entrenar un modelo de *machine learning*.
- Poder generar una matriz de confusión que muestre la tasa de acierto de los diferentes materiales empleados.
- Establecer una interfaz capaz de poder ser implementada en el sistema operativo *Windows* y *Linux*, para ello la aplicación tiene que poder ser implementada en cualquier dispositivo que cuente con estos sistemas.

2.2. Objetivos técnicos

En este proyecto entran en juego diferentes campos técnicos, entre los que podemos destacar la electrónica, sistemas computacionales, física, mecánica, mecatrónica, por lo cual se presentan los siguientes objetivos técnicos:

- Conocer e identificar las diferentes tecnologías empleadas en esta metodología particular de teledetección, describiendo qué es un radar, cuál es su operación, cuales son sus limitaciones, logrando aterrizar la idea principal del proyecto del empleo de un radar para el estudio.
- Conocer los datos técnicos del radar *Acconeer* el cual será implementado para establecer cuáles serán sus requerimientos en cuanto a alimentación eléctrica, conexión, interacción con los demás componentes que integran el proyecto
- Aplicar diferentes metodología de aprendizaje automático que nos permitan determina cuál es la mejor estrategia simplificando la operación de identificación de los diferentes materiales.
- Establecer las diferentes características a analizar de los materiales buscando poder diferenciarlos con el empleo del sensor radar *Acconeer*, para establecer la base de datos generada.
- Determinar qué lenguajes serán los óptimos para ser empleados de acuerdo con los componentes y los conocimientos obtenidos durante el análisis de la literatura que integra este proyecto.
- Establecer las bases para futuras investigaciones las cuales deseen implementar este tipo de tecnología en sus proyectos, sirviendo como referencia y apoyo a los diferentes desarrolladores.

Conceptos teóricos

Para poder construir un modelo capaz de capturar con precisión las características ofrecidas del resultado de leer cada objeto, es necesario comprender el origen y la estructura de la señal recibida. En este apartado se introducen algunos conceptos fundamentales del sistema de radar.

3.1. Radar

El radar o mejor conocido por su acrónimo del inglés, *radio detecting and ranging*, es un sistema electrónico que puede emitir y recibir radiación en un cierto rango del espectro electromagnético, y cuyo procesamiento permite la estimación de distancias, altitudes, direcciones y velocidades de objetos. El sistema de radar tiene su uso muy extendido en los campos de la meteorología, el control del tráfico aéreo y terrestre y múltiples usos militares [1].

Se han desarrollado múltiples avances tecnológicos los cuales han hecho posible que en la actualidad se puedan emplear los radares en diferentes y diversas áreas con seguridad y eficiencia. Los constantes avances tecnológicos en el desarrollo de materiales y su implementación en los circuitos integrados, junto con el desarrollo de la informática, son factores importantes que han convertido a los sistemas de radares en instrumentos de bajo tamaño con una gran precisión y calidad.

Principio de funcionamiento del Radar

El radar es un sistema de teledetección activa[1] que emite ondas de microondas, con frecuencias comprendidas entre 1 GHz y 100 GHz. Estos valores también se pueden considerar como longitudes de onda con valores entre los 3 mm y 30 cm [2].

Las ondas de microondas tienen diferentes ventajas:

- Son capaces de atravesar los cúmulos de nubes, la lluvia y la nieve.
- Permiten trabajar en condiciones de oscuridad debido a que no dependen de la iluminación solar o artificial.
- Miden características de los materiales como la rugosidad y la humedad, entre otras características.

Los sistemas radar de apertura sintética, también conocidos como SAR, cuentan con un sensor radar capaz de emitir un pulso de ondas de microondas dirigida hacia la superficie del terreno u objeto. En cierto momento ese pulso retorna en dirección al sistema SAR que se encuentra ubicado en un satélite donde la información conseguida es medida por el sensor radar.

Los pulsos empleados por el sistema SAR, típicamente poseen longitudes de onda que pueden variar desde los 3 cm de la banda X, hasta los 24 cm de la banda L, por lo que cuanto mayor sea la longitud de onda, mayor será su capacidad de penetración en zonas vegetadas, sin embargo, habría que considerar que será peor su resolución espacial.

Los sistemas SAR pueden ser transportados mediante un satélite, un avión o permanecer en una plataforma terrestre, en donde se emiten y reciben la señal del pulso de microondas emitido por el sensor radar, generando imágenes complejas de alta resolución espacial de la superficie del terreno.

Podrían mencionarse diferentes casos en los cuales se generan imágenes complejas de alta resolución espacial, por ejemplo:

- El satélite *ERS-1/2* [3] junto con él satélite *ENVISAT* (pertenecientes a la Agencia Espacial Europea) poseen un sensor que emite ondas en la banda C, esta onda tiene una longitud de onda de 6 cm, que generan imágenes SAR, éstas pueden llegar a tener una resolución espacial de $4m \times 20m$.

[1] Proceso de teledetección que funciona con su propia fuente de emisión o luz.

- El satélite *TERRASAR X* [4] es otro ejemplo de sistema SAR, en el que las ondas de microondas son generadas en la denominada *banda X*. Permite obtener imágenes de resoluciones de $1 \times 3m$.

En la obtención de imágenes SAR una de las partes más importantes del radar es el emisor (instrumento que produce la radiación electromagnética). Se puede considerar que cada píxel es representado con un formato de número complejo que contiene un valor de amplitud, junto con otro valor de fase. El valor de la amplitud está directamente relacionado con el coeficiente de respuesta generado con la información de la superficie del terreno dirigido a la señal de respuesta[5].

Tipos de radar

En el mercado y de acuerdo a su forma de operar existen diferentes tipos de radares. Estos pueden operar de modo primario o de modo secundario, considerando que todos los radares tienen la misma base de operabilidad, ya que todos los radares tienen en común la capacidad de poder transmitir una señal, pudiendo generar la evaluación mediante diferentes técnicas de procesamiento de las señales recibidas para así poder obtener el parámetro buscado [6].

Los sistemas de radar por lo general se encuentran seccionados en diferentes categorías de operación las cuales están basadas en los diferentes métodos de transmisión de la señal. Por lo general estos métodos corresponden a la transmisión de pulsos, la onda continua y la frecuencia modulada. El método de transmisión de pulsos es el método más común para implementarse en la transmisión de la energía del radar.

El método de onda continua se basa en el principio del *«El Efecto Doppler»*, este principio es capaz de generar la detección de la presencia y velocidad de un objeto en movimiento en dirección al radar o en dirección opuesta al radar. Suele ser empleado en los sistemas de control de fuego o disparo en el campo militar, para poder rastrear objetos que presentan movimientos rápidos en un rango cercano.

3.2. Teledetección

La teledetección, también conocida como detección a distancia o detección remota, es la técnica que permite recopilar información a distancia de los

objetos sin que exista un contacto físico, para ello es necesario tener una interacción entre los objetos a estudiar y un sensor situado en una plataforma.

En la teledetección el radar es un sensor activo de microondas desplazado a bordo de una plataforma explorando la tierra, por lo general se encuentra emitiendo pulsos de energía con dirección a la superficie terrestre, pudiendo almacenar las señales obtenidas de retorno [7].

Debido a que tienen que trabajar con haces de energía emitidos artificialmente, los sistemas de radar permiten controlar las condiciones de la adquisición basadas en la frecuencia, la polarización y la geometría de la observación. Pueden recolectar datos en cualquier condición ya sea de día o de noche, el realizar este tipo de trabajo resulta muy ventajoso en las regiones polares, lugares donde los prolongados períodos de oscuridad dificultan la adquisición de imágenes convencionales.

Un sistema de teledetección por radar transmite diferentes pulsos de microondas que van barriendo la superficie terrestre, la porción de energía que es reflejada o retrodispersada retorna de vuelta hacia el sensor del radar. En el momento que el radar recibe la señal el sistema registra la intensidad de la señal de retorno conocida como radiación retrodispersada, donde el retardo en tiempo, entre la transmisión y recepción de cada pulso de energía, es el tiempo que se relaciona con la distancia de los objetos observados.

La distancia a la cual debe estar situado un sensor para poder ser considerado remoto puede variar ya sea desde pocos decímetros hasta miles de kilómetros.

La teledetección es un flujo de radiación emitido por los objetos o materiales hacia un radar o sensor. El origen del flujo puede venir de distintas ubicaciones:

- Radiación emitida por el sensor y reflejada por los objetos
- Radiación terrestre emitida por los objetos
- Radiación solar reflejada por los objetos

El grupo de datos adquiridos por medio de los diferentes procedimientos empleados en la teledetección son generados desde plataformas como aviones o satélites, estos contienen por lo general tres tipos de información (espacial, espectral y temporal) [8].

Una propiedad de los datos obtenidos de la teledetección, es el poder permitir dar seguimiento al desarrollo de las enormes extensiones forestales,

existentes en la superficie de la tierra, al tener una perspectiva general de las causas de los efectos que se producen debido a las grandes catástrofes. Como pueden ser las eternas sequías en las regiones desérticas del Sahara de África, en este caso, es posible ubicar determinados fenómenos generados por la contaminación a gran escala afectando al cielo y el ambiente en el mar.

Mediante la tecnología radar se pretende actualizar el conocimiento de las riquezas naturales nacionales e internacionales, ubicadas en las áreas agrícolas, áreas forestales, áreas hidrológicas, áreas mineras o cualquier áreas que represente una fuente de recursos para la humanidad [9].

De tal manera que la vigilancia debe ser constante para el medio ambiente, debido a que esta actividad podrá generar una importante reducción en el impacto de degradación causado por los humanos hasta este momento. Por lo que los datos obtenidos del análisis de teledetección se convierten en una importante fuente de información, siendo un gran papel en el cumplimiento de los objetivos del cuidado del medio ambiente.

En el caso español, considerando que las acciones más imprescindibles deben ser enfocadas en la calidad del agua, debido a que el agua es una gran riqueza para la Península Ibérica, básicamente es necesaria para mantener la vida y el desarrollo del ser humano; otra acción importante sería la detección de incendios.

La implementación de la teledetección impacta de forma positiva en la preservación y el mejoramiento de la calidad del agua de los ríos y embalses, debido a que el uso de estas técnicas permite vigilar su situación para poder actuar y manejar cualquier vertido contaminante que se llegara a producir [10].

Es por ello por lo que las técnicas de radar impactan positivamente en la generación de un pronóstico, obteniendo la detección y generando un monitoreo constante de posibles desastres naturales en las diferentes áreas del planeta; como la instalación de una estación de monitoreo de mareas por el *Centro para Alarmas por Tsunamis en el Pacífico*, conocido como PTWC (por sus siglas en inglés), además del *Servicio Nacional de Estudios Territoriales* (SNET), la cual permite obtener información sobre el nivel del mar y los cambios que se originan en las costas del Pacífico. Es así como la obtención de información podría evitar daños por la generación de un tsunami en la región de este océano que podría afectar a Centroamérica.

Todos los elementos existentes en la naturaleza cuentan con una respuesta espectral denominada *signatura espectral*, esta es capaz de estudiar las

variaciones espectrales, las variaciones espaciales y las variaciones temporales de las ondas electromagnéticas, poniendo de manifiesto las relaciones existentes entre las variaciones junto con las características que tienen los diferentes materiales terrestres.

Es por ello que su objetivo básico se centra en la identificación de los diferentes materiales de la tierra y los diferentes fenómenos que en ella se generan a través de su característica espectral. La teledetección como herramienta ofrece enormes posibilidades para la generación de avances en el conocimiento e interpretaciones de condiciones físicoambientales, constituyendo una fuente de información y desempeñando un papel significativo en el campo del conocimiento geográfico [11].

En este sentido la teledetección se define como el proceso de análisis de la energía reflejada por los objetos, es decir, es una técnica que puede aportar una información muy valiosa para distintos campos de intervención.

Tipos de teledetección

Se llama **teledetección pasiva** a la técnica aplicada por los sensores que miden las variaciones de la energía procedente de los objetos sin intervenir en el campo natural.

Los **sensores pasivos** o también conocidos como ópticos miden (en unidades de radiancia) la energía electromagnética solar reflejada por la Tierra, y además la radiación térmica emitida por la Tierra en distintas bandas espectrales o en diferentes intervalos del espectro electromagnético. Aquí la cantidad de radiación reflejada va a depender de las condiciones de iluminación presentes. La proporción entre la radiación incidente y reflejada, tienen una magnitud denominada reflectancia (propiedad específica de la superficie reflectante).

Se denomina **teledetección activa** a aquellos que generan un campo de energía artificial (su propia fuente de radiación de microondas), utilizado para registrar y medir el efecto que en él producen los objetos, donde las características técnicas del sensor influyen en la calidad de los datos y en la posibilidad de recibir información en distintas longitudes de onda [12].

Los **sensores activos** sin imagen son los que incluyen los altímetros y escatómetros, por los cuales en la mayoría de los casos son dispositivos de perfil (toman medidas en una dimensión), es lo opuesto a la representación bidimensional de los sensores de imagen. Los sensores en la teledetección activa se pueden dividir principalmente en dos categorías, en *con imagen* y

en *sin imagen*. Debido a que los sensores activos con imagen son basados en técnicas de radar o de SAR para poder generar imágenes en dos dimensiones.

3.3. Radar *Acconeer*

El radar utilizado en el proyecto está fabricado por *Acconeer* y su modelo tiene la referencia *A111*.

El radar *A111* es un sistema de radar basado en tecnología de radar coherente pulsado, también conocido como PCR[2], el cual está estableciendo un nuevo punto de referencia para el consumo de energía y la precisión de la distancia, es totalmente integrado en un paquete pequeño de 29 mm^2 [13].

El sistema de radar *A111* es de 60 GHz, el cual está optimizado para operar a una alta precisión, con una potencia ultra baja. Por lo general se suministra como una óptima solución de un solo paquete con banda base integrada, interfaz de Radio Frecuencia (RF) y Antena en el Paquete (AiP), esto permitirá una fácil integración en cualquier dispositivo portátil que funcione con batería.

El radar *A111* se basa en una tecnología de sensor patentada de vanguardia con resolución de tiempo de pico de segundo, es capaz de medir distancias absolutas con precisión milimétrica hasta un rango de 2 m, ya que el rango de 2 m está garantizado para un tamaño de objeto, forma y propiedades dieléctricas correspondientes a un reflector esférico de esquina de 5 cm de radio, además de contar con una tasa de actualización configurable.

El radar de 60 GHz no se ve afectado por ninguna fuente natural de interferencia, como lo es el ruido, el polvo, el color o la presencia de luz directa o indirecta.

[2]Pulsed Coherent Radar

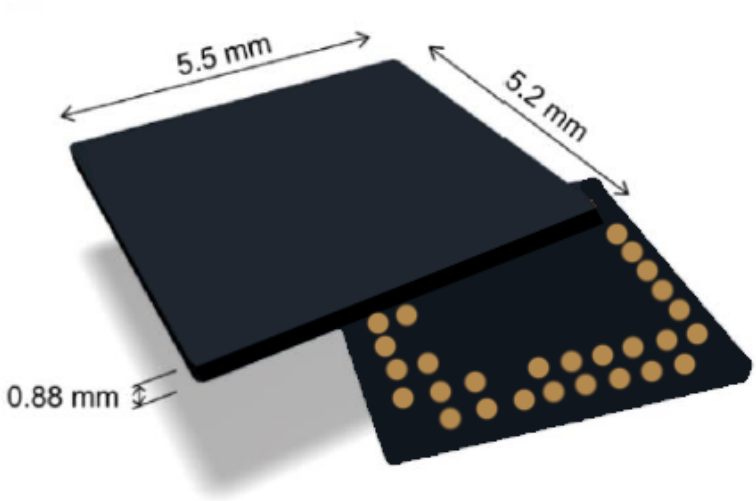

Figura 3.1: Radar *A111*.

Aplicaciones:

- Puede generar mediciones de distancia de alta precisión (del orden del mm) y alta tasa de actualización.
- Puede realizar detección de proximidad con alta precisión y posibilidad de definir múltiples zonas de proximidad.
- Puede generar detección de movimiento y detección de velocidad.
- El radar permite la detección de material.
- Se puede generar el seguimiento de objetos de alta precisión, que permite el control por gestos.
- Es posible desarrollar seguimiento de alta precisión de objetos 3D.
- Es posible que controle los signos vitales de la vida, como la respiración y el pulso.

Algunas de las características principales con las que se cuenta y que se pueden describir del radar son:

- Cuenta con un alcance y movimientos precisos de la distancia, ya que puede:

  - Medir el rango absoluto hasta 2 m o precisión absoluta en mm.
  - Tiene una precisión relativa en micrómetros.
  - Cuenta con la posibilidad de reconocer movimientos y gestos para varios objetos.
  - Admite modo de barrido continuo y único.
  - Tienen ancho de haz de media potencia (HPBW[3]) de 80 (plano H) y 40 grados (plano E).
- Su fácil integración es debida a que cuenta con:
  - Una solución de un chip con banda base y RF integrados.
  - Puede integrarse detrás de plástico o vidrio sin necesidad de una apertura física.
  - Cuenta con componente refluible único.
  - Fuente de alimentación única de 1.8 V, habilitada con *Power on Reset* (PoR[4]).
  - Una entrada de reloj para cristal o reloj de referencia externo, 20-80 MHz.
  - Una interfaz *Serial Peripheral Interface* (SPI[5]) para transferencia de datos, soporte de reloj SPI de hasta 50 MHz.

El radar *Acconeer A111* es un sensor de radar optimizado de baja potencia y alta precisión de 60 GHz con banda base integrada, una interfaz de RF y una antena en paquete (AiP).

El sensor se basa en la tecnología de radar coherente pulsado (PCR), que presenta una solución patentada de vanguardia con resolución de tiempo de picosegundos. El *A111* es la elección perfecta para implementar sistemas de detección de alta precisión y resolución con bajo consumo de energía.

El radar de silicio *A111* esta dividido en cuatro bloques funcionales: *Power*, *Digital*, *Timing* y radio *mmWave*.

[3]Ancho angular total entre los dos puntos que están 3dB abajo del pico del haz principal.

[4]Microcontrolador generador de reinicio de encendido.

[5]Bus de interfaz estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

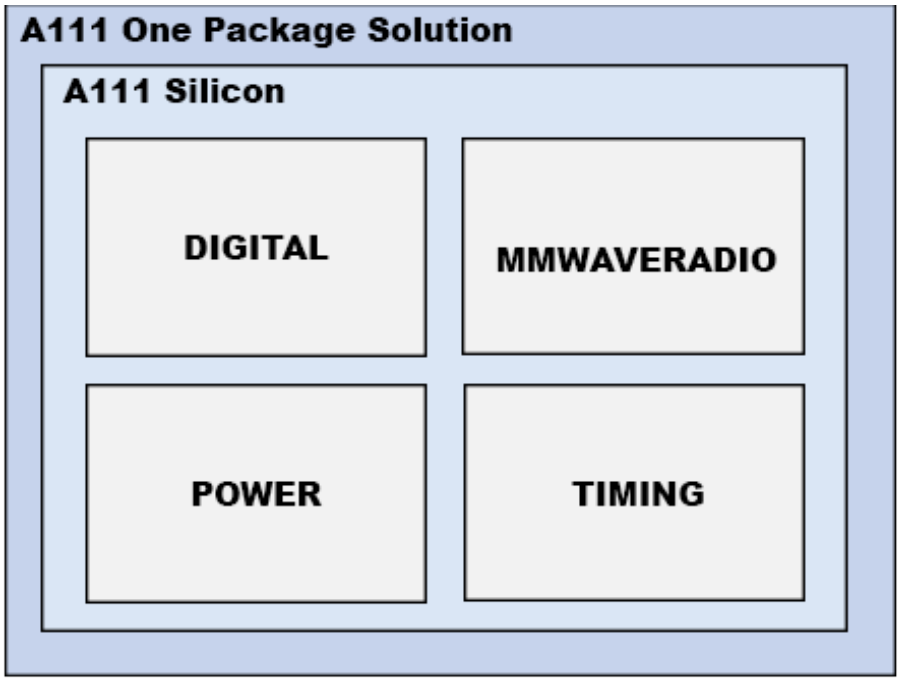

Figura 3.2: Diagrama de bloques del sensor *A111*.

La figura 3.1 muestra un diagrama de bloques del sensor *A111*. La señal se transmite desde la antena de transmisión (Tx) y es recibida por la antena de recepción (Rx), ambas integradas en la capa superior del sustrato del paquete *A111*. Además de la radio *mmWave*[6], el sensor consta de administración de energía y control digital, cuantificación de señales, memoria y un circuito de temporización.

El bloque funcional *Power*[7] incluye *Low Dropout* (LDO[8]) y un bloque Power on Reset (PoR). Cada LDO crea su propio dominio de voltaje. El bloque PoR genera una señal de reinicio en cada ciclo de encendido. El host conecta el bloque funcional de alimentación del sensor a través de una fuente de alimentación única de 1,8 V.

El bloque funcional digital incluye control por sensor. La memoria de datos almacena los datos de barrido del radar del conversor de señal analógica

[6]Rango de Frecuencia 2 (FR2)
[7]Bloque de conexión eléctrica del radar.
[8]Reguladores de voltaje de caída baja.

a digital (ADC). El host interactúa con el sensor a través de una interfaz SPI, un reloj (XIN, XOUT) y una señal de INTERRUPCIÓN[9].

El bloque de temporización incluye los circuitos de temporización.

El bloque funcional de radio *mmWave* genera y recibe pulsos de radar e incluye transmisor (TX), receptor (RX) e interfaces hacia las antenas integradas.

El software *Acconeer* se ha escrito en lenguaje *C* y es portátil para cualquier sistema operativo y plataforma *HardWare* (HW[10]). El software *Acconeer* se ejecuta en *Host Unidad de Control Multipunto* (MCU[11]) y se entrega como binarios, a excepción del software de integración que se entrega como código fuente.

El RSS (software del sistema de radar) proporciona salida en dos niveles diferentes, servicio y detector. RSS proporciona una API (interfaz de programación de aplicaciones) para la utilización de aplicaciones de varios servicios y detectores.

La salida de servicio son datos de sensor preprocesados en función de la distancia. Por ejemplo, datos de envolvente (amplitud de los datos del sensor), datos de la bandeja de potencia (datos de amplitud integrados en intervalos de rango predefinidos), datos modulados por IQ[12] (representación en cartesiano), etc. Los detectores se basan en datos de servicio como entrada, y la salida es un resultado.

El cliente puede utilizar el detector *Acconeer* o desarrollar su propio procesamiento de señales basado en los datos del servicio. *Acconeer* proporciona varias aplicaciones de ejemplo para respaldar el desarrollo de aplicaciones propias del cliente. Además, se proporcionan pautas para el cliente para el desarrollo de aplicaciones utilizando la API RSS de *Acconeer*.

Acconeer proporciona varios controladores de referencia como código fuente, p.ej. soporte para *MCU Cortex M4* y *Cortex M7*. El software de integración implementará las funciones definidas en un archivo de definiciones proporcionado en la oferta de *Acconeer Software*. Esto incluye el manejo de *SPI, ENABLE e INTERRUPT*, así como posibles funciones del sistema operativo.

[9]Señal recibida por el procesador de una computadora, que indica que debe interrumpir el curso de ejecución actual.

[10]Equipo o soporte físico

[11]Dispositivo de red que se usa como puente en conexiones de audioconferencia y videoconferencia

[12]Servicio *In-phase and Quadrature*

En el siguiente enlace se puede acceder a los productos y características que ofrece *Acconeer* (https://www.acconeer.com/products).

El sensor se puede ejecutar en uno de los siguientes servicios básicos de la tabla 3.1.

Servicio	**Tipo de dato**	**Ejemplo de uso**
Envelope	Amplitud	Distancia absoluta y presencia estática
IQ	Amplitud y fase	Detección de obstáculos, respiración y distancia relativa
Sparse	Amplitud instantánea	Velocidad, detección de presencia y detección de gestos

Tabla 3.1: Servicios del radar *A111*

3.4. Servicio *In-phase and Quadrature*

El servicio *In-phase and Quadrature* (IQ[13]) utiliza la coherencia de fase[14] del radar pulsado *Acconeer* para producir componentes estables en fase y en cuadratura. Este servicio se puede utilizar para la detección de presencia frente al sensor, la detección de la frecuencia respiratoria, la detección de obstáculos y, en nuestro caso, para diferenciar materiales.

Los componentes en fase y en cuadratura se representan como números complejos, lo que genera un conjunto complejo de N_D muestras representadas como $x[d]$, dónde d es el índice de demora de la muestra.

Los datos obtenidos a través del servicio IQ proporcionan un método para examinar la reflectividad a diferentes distancias del sensor de radar.

Las técnicas de análisis de imágenes se han venido desarrollando debido al gran número de aplicaciones donde se emplean, detección y análisis de cambios sobre la superficie terrestre, mediciones de campos, determinación de extensiones de hielo, glaciares, etc.

Es posible el aplicar algoritmos en los que mediante la definición de umbrales adecuados se diferencian las regiones, estableciéndose la presencia de bordes, por ejemplo, usando la diferencia entre píxeles, o en el cociente entre los valores de los píxeles [14].

[13] En fase y cuadratura.

[14] Se dice que dos puntos de una onda son coherentes cuando guardan una relación de fase constante.

El problema que se presenta en este método es la aparición de un ruido multiplicativo conocido como *speckle*, en donde este ruido se debe a la interferencia que se produce al sumar las señales coherentes provenientes de los elementos dispersores presentes en la región asociada al píxel considerado. Por lo que se hace necesario el uso de herramientas matemáticas más sofisticadas para la detección de bordes, donde las imágenes SAR son ofrecidas en distintos formatos y niveles de procesamiento, siendo la más básica de las imágenes la denominada SLC (*Single Look Complex*).

Una adquisición SAR en polarización simple, vertical u horizontal, tiene dos bandas de datos que corresponden a los canales de adquisición I (*in phase*) y Q (*in quadrature*), los cuales pueden ser interpretados como la parte real e imaginaria, respectivamente, de una señal compleja.

El *speckle* ha sido descrito teóricamente mediante modelos de distribución en regiones homogéneas, estos modelos establecen que para imágenes SLC de polarización simple (por ejemplo áreas rurales), los datos de ambos canales I y Q, tienen una distribución Gaussiana para las amplitudes.

Mientras que la fase tiene una distribución prácticamente uniforme y la intensidad de una distribución exponencial (o equivalentemente Gamma de orden 1), el módulo de la amplitud (que es proporcional a la raíz cuadrada de la intensidad) responde a una distribución de Rayleigh.

Considerando esto, se han usado diferentes técnicas que tienen en cuenta la correspondiente distribución de la potencia que se recibe de cada píxel y se han calculado las distribuciones para los gradientes de intensidad asociados a los píxeles, con esto se ha conseguido una mejor identificación de los bordes.

Un radar *Doppler* tiene etapas de detección síncrona, es decir, que están en sincronía con el oscilador de transmisión, lo que define un receptor coherente, debido a que las señales *Doppler* contienen no sólo información de amplitud sino también de fase, es crucial conocer estas componentes para una correcta demodulación[15].

Una forma de obtenerlas es indirectamente, conociendo la amplitud de la señal en fase (I) y la amplitud de la señal en cuadratura (Q), donde estas señales se obtienen al mezclar la señal recibida con las señales obtenidas de un oscilador sincronizado que genere una en fase y la otra desfasada 90º [15].

[15]Conjunto de técnicas utilizadas para recuperar la información transportada por una onda portadora, que en el extremo transmisor fue modulada con dicha información.

Una forma de obtener estas componentes es utilizando receptores enteramente digitales, aunque el receptor analógico tuvo muchos años de vigencia, las técnicas actuales permiten realizar receptores enteramente digitales en un sólo chip.

Es importante notar que la detección de la información se puede realizar en banda base o en una frecuencia intermedia, debido a que la señal de la antena y la señal del oscilador local tendrían la misma frecuencia en el caso del receptor homodino.

Un receptor digital consiste en una fuente analógica, en este caso la señal recibida por la antena, la cual es previamente filtrada y amplificada por un amplificador de bajo ruido (LNA), para a continuación la señal es digitalizada por un conversor análogo-digital (ADC) y luego es multiplicada digitalmente por una señal de referencia para obtener la frecuencia intermedia o la señal en banda base.

Además de la señal de entrada RF se obtienen dos señales, la primera es producto de la multiplicación por la referencia, y la otra es producto de la multiplicación por la referencia desfasada 90º, donde el objetivo es obtener información no sólo de magnitud sino también de fase respecto a la referencia.

Cabe señalar que el receptor digital concentra su selectividad en el filtro digital, donde este filtro, para el caso del radar *Doppler*, suele ser un filtro adaptado (*matched filter*) con el fin de obtener la mejor relación de señal a ruido.

3.5. Aprendizaje automático

El aprendizaje automático o *machine learning* es una disciplina del campo de la Inteligencia Artificial (IA), consiste en dotar a una computadora de capacidad para que mejore en la realización de una tarea a partir de datos de ejemplo o de la experiencia. Se puede visualizar en áreas como los procesos estadísticos, ya que tiene muchas ventajas y reduce el tiempo de demora de los procesos, además de emplear menos personal para realizar las tareas.

Es un área que estudia cómo construir programas de computadoras que mejoren su desempeño en alguna tarea gracias a la experiencia, debido a que esta se basa en las ideas de diversas disciplinas, como inteligencia artificial, estadística y probabilidad, teoría de la información, psicología y neurobiología, teoría de control y complejidad computacional. Estudia la construcción de sistemas capaces de aprender a partir de datos, lo que incluye

una gran variedad de sistemas, desde sistemas de visión por ordenador hasta sistemas para detectar correo no deseado (*spam*).

Este sistema proporciona una gran ventaja a la hora de examinar de manera más efectiva una mayor cantidad de datos, debido a que en la actualidad los avances tecnológicos han permitido crear nuevas experiencias tecnológicas en diferentes áreas tales como: empresas, educación, sociedad, entre otros.

La inteligencia artificial es un concepto de creación de máquinas inteligentes que estimula el comportamiento humano, mientras que el aprendizaje automático es un subconjunto de la inteligencia artificial que permite que la máquina aprenda de los datos sin ser programada.

La diferencia entre el software informático normal y el aprendizaje automático es que un desarrollador humano no ha dado códigos que le indiquen al sistema cómo reaccionar ante la situación, sino que está siendo entrenado por una gran cantidad de datos.

La mayoría de información generada a diario en Internet, ya sea por medio de las redes sociales, por medio de diferentes transacciones comerciales o datos generados por distintos dispositivos, en lugar de almacenarla y que solo ocupe espacio en los servidores, es aprovechada por procesos que emplean los datos para poder generar un análisis que establezca comportamientos capaces de identificar las tendencias futuras [16].

En ciertas ocasiones se reúne demasiada información, por lo que es posible determinar con anticipación y de forma segura el comportamiento futuro de un grupo de personas que interactúan con equipos electrónicos, ya que la técnica del *Machine Learning* (elemento fundamental de la Ciencia de Datos) utiliza métodos para realizar las predicciones de datos y su presentación [17].

Los diferentes dispositivos que cuentan con la conocida inteligencia artificial, son capaces de ejecutar distintos procedimientos análogos con el comportamiento humano. Es el caso de una devolución a una respuesta por cada petición de entrada, parecido o similar a un tipo de reflejos que se encuentran en los seres vivos, estableciendo un estado especifico entre los diferentes estados posibles según una determinada acción, con la solución de problemas empleando una lógica formal.

Cuando se les es otorgada a un determinado tipo de dispositivos la habilidad de poder aprender, se genera la posibilidad de que puedan discernir. Por lo tanto, se convierten en entidades que razonan con capacidades semejantes

a las de un superhombre, dado que su velocidad de procesamiento es prácticamente imposible para un humano promedio. Las maquinas no tienen la necesidad de descansar para poder desarrollar sus funciones correctamente. Son algunas ventajas que ubican a las maquinas sobre los seres vivos en este contexto.

En este sentido es posible encontrar tres grupos de algoritmos en *Machine Learning*.

- **Los algoritmos supervisados**, algoritmos que emplean un conjunto de datos de entrenamiento etiquetados o también conocidos como preclasificados. Los datos son procesados para poder realizar predicciones sobre ellos, existiendo la posibilidad de corrección cuando son incorrectas [18].
- **Algoritmos semi-supervisados** combinan tanto datos etiquetados como datos no etiquetados para poder generar una función determinada o clasificada. Estos tipos de modelos son los que tienen que aprender las estructuras para poder organizar los datos, es así como pueden realizar predicciones.
- **Los algoritmos no supervisados** están diseñados para clasificar datos de los que no existe información sobre su etiqueta, por lo que no se cuenta con un resultado conocido. Por ello es necesario deducir las estructuras que se encuentran presentes en los datos de entrada, lo cual puede ser conseguido a través de un proceso matemático para poder bajar la redundancia sistemáticamente tratando de organizar los datos por similitud.

Dentro de esta clasificación se pueden encontrar diferentes números de algoritmos específicos con las diferentes características para poder generar el tratamiento de los datos, entre los algoritmos más relevantes podemos encontrar el *Deep Learning*, consiste en el empleo de algoritmos para poder hacer representaciones abstractas sobre la información y facilitar el aprendizaje automático.

Otro algoritmo es *Active Learning*, es un caso de aprendizaje semi-supervisado, el algoritmo de aprendizaje puede interactuar con un usuario, además de otra fuente de información para poder obtener los resultados deseados.

El clasificador *Support Vector Machines (SVM)* busca generar la maximización de la distancia de las muestras de cada clase a la llamada «frontera

de decisión» (también conocida como plano). En el caso de que las muestras obtenidas no sean linealmente separables se emplea una transformación llamada *kernel*.

En todos los casos, un sistema que aprende debe ser capaz de generalizar, es decir, de encontrar patrones y regularidades en los datos que le permitan desempeñarse bien en datos que no ha observado previamente.

Tipos de aprendizaje automático

Aprendizaje supervisado

El algoritmo de aprendizaje conocido como supervisado emplea un programa capaz de recibir datos de entrada etiquetados y los datos de salida esperados. Obtiene los datos de los datos de entrenamiento que contienen conjuntos de ejemplos.

Generan dos tipos de resultados:

- Clasificación: notifican la clase de los datos que se presentan.
- Regresión: esperan que el producto produzca un valor numérico.

El aprendizaje supervisado, se da cuando se entrena un algoritmo de *Machine Learning* aportándole las preguntas, las cuales son características, y una serie de respuestas conocidas también como etiquetas, por lo que así en el futuro el algoritmo sea capaz de hacer una predicción bajo el conocimiento de las características. Generalmente en este tipo de aprendizaje hay dos algoritmos, son el algoritmo de clasificación y el algoritmo de regresión.

Los algoritmos de clasificación son los que esperamos que por su naturaleza nos indiquen a qué grupo pertenece un determinado elemento en estudio. Por lo general, el algoritmo tiene la capacidad de encontrar patrones en los datos que le proporcionamos pudiendo clasificarlos en grupos, ya que luego se comparan con nuevos datos obtenidos, para después poder ubicarlos en uno de los grupos, siendo de esta manera como puede predecir de que se trata de objeto de estudio [19].

Aprendizaje no supervisado

El aprendizaje sin supervisión es un conjunto de metodologías que por lo general se emplean principalmente en análisis exploratorios, debido a que puede identificar automáticamente la estructura en los datos.

En el caso de los algoritmos de aprendizaje no supervisado, no es necesaria de la intervención humana para poder desarrollar un conjunto de datos, los cuales son previamente clasificados para poder representar el algoritmo de aprendizaje, dado que el objetivo de la enseñanza no supervisada es el poder encontrar modelos interesantes teniendo en cuenta la distribución con la compilación de los datos que se presentan. Algunos ejemplos que se pueden mencionar sobre técnicas de aprendizaje no supervisado son las técnicas que se aplican de agrupación [20].

Aprendizaje semisupervisado

El algoritmo de aprendizaje semisupervisado emplea el método del aprendizaje que es automático, el cual evita la gran cantidad de datos etiquetados, mejorando este tipo de limitación, ya que aprende a clasificar a partir de un número, casos de ejemplos etiquetados junto con otros no etiquetados. Los datos etiquetados se usan para poder aprender modelos que caractericen cada clase o categoría, y los ejemplos sin etiqueta se usan para refinar los límites entre las clases [21].

El llamado *self-training* la cual es una de las formas más simples de clasificación semisupervisada, donde primero se construye un clasificador usando los datos etiquetados, para después emplear el clasificador que categoriza a los datos no etiquetados, ya que sólo los nuevos ejemplos etiquetados con una confianza que supere cierto umbral se anexan al conjunto etiquetado, es cuando el proceso de aprendizaje se repite.

Por obvias razones los algoritmos de aprendizaje semisupervisado se encuentran entre los de aprendizaje supervisado y los de aprendizaje sin supervisión, dado que en los algoritmos de aprendizaje semisupervisado es posible dividir los datos en dos partes, las cuales están integradas por un grupo de datos clasificados y por un grupo de datos no clasificados, por lo consiguiente se llama aprendizaje semisupervisado estándar.

Para diferentes investigadores, el aprendizaje semisupervisado es más útil cuando hay más datos no clasificados comparado con los datos clasificados, dado que especialmente cuando es necesario mucho esfuerzo para poder obtener datos clasificados se tiende a tardar tiempo aumentando el costo, y obtener datos no clasificados generalmente es más barato.

Algunos ejemplos donde se aplican las técnicas de aprendizaje semisupervisado es en las técnicas de máquinas de vectores de soporte transductivo, otro ejemplo es la maximización de las expectativas, además de poder mencionar algunas aplicaciones donde se aplican las técnicas de aprendizaje

semisupervisado mencionadas, estas podrían ser la clasificación de páginas web, el reconocimiento de voz y la secuencia de la proteína.

Aprendizaje por refuerzo

El aprendizaje por refuerzo es un campo dentro del aprendizaje automático, los problemas que se tratan en esta área requieren de la toma de un conjunto de decisiones secuenciales para poder conseguir un objetivo determinado, dado que, en estas situaciones, una entidad denominada *agente* es la que interacciona con el entorno, debido a que aprende de forma autónoma las acciones que debe llevar a cabo con el fin de maximizar una señal numérica escalar, conocida como recompensa [22].

El algoritmo de aprendizaje por refuerzo puede aprender observando el mundo que lo rodea, la información de entrada es la retroalimentación que recibe del mundo exterior en respuesta a sus acciones, por lo que el sistema aprende sobre la base de pruebas y errores. El algoritmo interactúa con el entorno y por lo general se puede plantear un método de seguimiento, donde es posible encontrar un error lo suficientemente apto para encontrar el mejor resultado basado en la experiencia. Por lo general, en un principio el agente no conoce la dinámica del entorno en el que se encuentra, tampoco conoce los detalles explícitos de la tarea que debe realizar, por lo que la experimentación es la única forma de la que se dispone para poder desarrollar el entrenamiento.

Dado que las dos cualidades implementadas, la experimentación y la recompensa con retraso, forman parte esencial de los problemas de aprendizaje reforzado, son estas las cualidades distintivas las que no pertenecen a ningún otro campo del aprendizaje automático.

Se puede decir que el objetivo de este aprendizaje es el poder aprender la función de valor, la cual ayuda al agente inteligente para poder maximizar la señal de recompensa, buscando optimizar sus políticas para comprender el comportamiento del entorno y así tomar las decisiones correctas, logrando cumplir con sus objetivos formales.

Podemos decir que entre las implementaciones desarrolladas se encuentra *AlphaGo*, programa de Inteligencia Artificial, desarrollado por *Google DeepMind* para poder jugar el juego de mesa *Go*, podría resultar interesante mencionar que en marzo de 2016, *AlphaGo* ganó el partido ante el jugador profesional Lee Se-Dol, el cual tiene la categoría novena «Dan» con 18 títulos

mundiales. Uno de los algoritmos utilizados es el «árbol de búsqueda de Monte Carlo»[16] que utiliza el aprendizaje profundo con redes neuronales.

Aprendizaje multitarea

El aprendizaje multitarea, también conocido como *Multitask Learning* (MTL), esta basado en la resolución de varias tareas al mismo tiempo. Este aprendizaje aprende tanto de las características comunes como de las diferencias entre ellas, por lo cual, el modelo estudia en paralelo todas las tareas, al mismo tiempo comparte la información aprendida con cada una de las tareas para poder resolver las otras tareas, por lo general esta técnica es especialmente útil cuando las tareas están relacionadas. Se ha demostrado que puede ser muy poderoso para tareas no relacionadas [23].

El aprendizaje multitarea es la aplicación de los métodos de aprendizaje que emplean el conocimiento previamente aprendido por el sistema, se emplean cuando se enfrentan a problemas similares a los ya vistos, lo que implica generar una solución simultánea de diferentes tareas. El aprendizaje de una tarea se mejora para poder complementarla con el aprendizaje común de otras tareas relacionadas con esa tarea.

Unas de las ventajas más importantes al emplear aprendizaje multitarea son las que comprenden la automatización, la precisión, la personalización, la rapidez, y la escalabilidad.

3.6. Algoritmos *machine learning* considerados

Machine Learning, en castellano aprendizaje automático, es una disciplina científica que se enfoca en crear sistemas que aprenden de datos, los cuales son capaces de realizar predicciones e identificar patrones complejos en ellos. Se puede establecer que por definición estos sistemas tienden a mejorar de forma autónoma, el científico informático Tom Mitchell establece una definición moderna de lo que es *Machine Learning*, *«Se dice que un programa de ordenador aprende de la experiencia E con respecto a alguna clase de tareas T y la medida de rendimiento P, si su desempeño en tareas en T, medido por P, mejora con la experiencia E.»*, dada esta definición es posible identificar dos tipos de aprendizajes, estos son el aprendizaje supervisado y aprendizaje no supervisado [24].

[16] Algoritmo de búsqueda heurístico realiza un proceso de toma de decisiones sobre todo relativo con los juegos.

La principal diferencia entre ambos está en el conjunto de datos que usaremos para entrenar, si usamos datos etiquetados o con un valor real estaremos hablando de aprendizaje supervisado, mientras que, si el conjunto de datos no está etiquetado, nuestro algoritmo deberá buscar un patrón en estos, por lo que estaremos ante un problema de aprendizaje no supervisado.

Los lenguajes de programación más usados en *Machine Learning* son: *Python, R, C++, Java y Scala. R* se usa debido a que es un lenguaje muy orientado al análisis estadístico y cuenta con muchas librerías desarrolladas capaces de analizar un conjunto de datos. *Python* destaca por su sintaxis intuitiva y una gran variedad de librerías para el desarrollo de aplicaciones de *Machine Learning*, las más relevantes son:

- *Pandas*: Nos ofrece una forma versátil de recoger estos datos en *streaming* para poder procesarlos posteriormente en *dataframes*.
- *SciPy*: Herramientas de matemáticas, ciencia e ingeniería.
- *Statsmodel*: Nos provee herramientas dedicadas para el análisis de series temporales.
- *Scikit-learn*: Para algoritmos de *Machine Learning*.
- *Numpy*: Para vectores y matrices.
- *Keras*: Al igual que *TensorFlow*, es una librería destinada al desarrollo de redes neuronales.

Los algoritmos que se han propuesto para ser utilizados en el planteamiento del desarrollo del proyecto son *Random Forest*, *k-NN*, *SVM*, *Auto-Sklearn* y *TabPFN* los cuales se describirán a continuación.

Random Forest

Ramdom Forest conocido en castellano como Bosques Aleatorios, es un algoritmo de clasificación supervisado, consiste en crear muchos árboles para luego usarlos en la predicción de la variable de interés. Para poder establecer un algoritmo basado en *Random Forest* (bosques aleatorios), es necesario describir en qué consiste un árbol de decisión, ya que en él podemos encontrar la combinación de los diferentes casos.

El árbol de decisión es una de las herramientas de aprendizaje supervisado más implementadas, este modelo consta de una serie de nodos, en los cuales

se toman decisiones lógicas de forma secuencial. Por ejemplo, si quisiéramos conocer si a alguien le deberíamos conceder un crédito, podríamos ver si en primer lugar obtiene más o menos de X ingresos [25]. En caso que fuera afirmativo podríamos querer saber si lleva más de Y años en su trabajo, o si tiene algún aval, dado que un árbol de decisión no es más que una acumulación de este tipo de decisiones. Empezamos en el nodo conocido como raíz, pasamos por los nodos intermedios y nuestro resultado es proporcionado por un nodo terminal, que serían las hojas del árbol.

Al final, los árboles de decisión lo que hacen es atribuir un valor constante al *output*[17] dentro de regiones que suelen ser rectángulos definidos en el espacio de los *inputs*[18]. La ventaja principal de estos métodos en la sencillez de interpretación, ya que podemos encontrar un caso con dos variables, donde el dominio de la variable *X1* se ha dividido en tres regiones, mientras que el de *X2* se descompone en función del valor de la primera variable.

Los árboles se crean siguiendo el algoritmo:

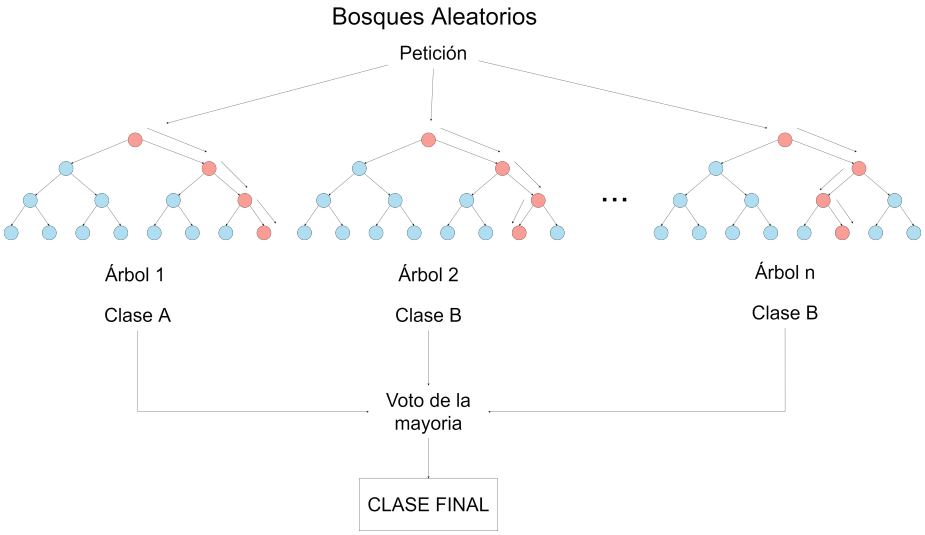

Figura 3.3: *Random Forest*

- Sea N el número de casos de prueba, M es el número de variables en el clasificador.
- Sea m el número de variables de entrada, m menor que M

[17] Resultado de procesar un conjunto de datos.
[18] Conjunto de datos que se introducen en el sistema.

- Se elige un conjunto de datos para el entrenamiento del árbol y el resto de los casos se utilizará para estimar el error.

- Para cada nodo del árbol, es posible elegir aleatoriamente m variables en las cuales basar la decisión, y así poder calcular la mejor partición del conjunto de entrenamiento a partir de las m variables.

La idea es que a medida que se vaya separando la muestra de entrenamiento a través de los diferentes nodos, se vayan formando grupos cada vez más homogéneos, hasta que las decisiones no generen grupos más uniformes. El método fragmenta el espacio formado por los datos obtenidos asignando un valor como resultado para todos los subespacios.

k-NN

k-NN (k vecinos más cercanos) también conocido como *k-nearest neighbors*, es un algoritmo de tipo simple, se basa en un control de parecido. *k-NN* se emplea para obtener el reconocimiento de patrones, también es empleado para obtener la estimación estadística, ya que es una técnica no paramétrica, y no hace suposiciones con los datos subyacentes. Su funcionalidad es poder buscar un número anteriormente definido de un conjunto de datos de entrenamiento que esté cercano al nuevo punto, pudiendo predecir la etiqueta a partir de los datos.

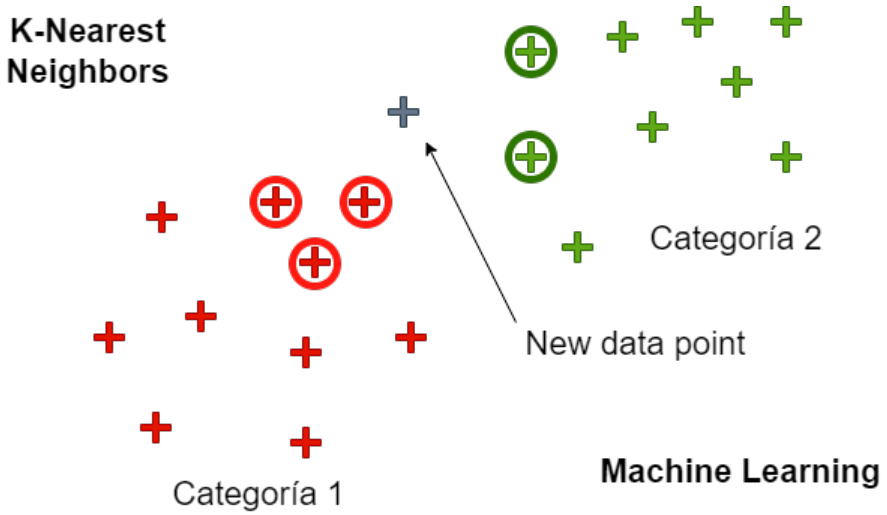

Figura 3.4: *k-nearest neighbors*

La cantidad de datos son una constante definida por los usuarios conocida como aprendizaje del vecino más cercano K, es posible que varíe de acuerdo con la cantidad local de puntos, donde la distancia es cualquier medida, por lo que la distancia euclidiana estándar es la más empleada. Dado que el dato más cerca de K es un algoritmo de aprendizaje automático basado en técnicas de aprendizaje supervisado, se asume el parecido entre el nuevo caso con los datos, ya que los casos disponibles para poder establecer el nuevo caso en su categoría deberá ser más similar a las categorías existentes [26].

El algoritmo *k-NN* es capaz de almacenar los datos disponibles, clasificando un punto en función de la similitud, por lo que cuando aparecen datos nuevos, es posible clasificarlos fácilmente en una categoría empleando el algoritmo *k-NN*. Puede ser conocido como algoritmo de aprendizaje perezoso debido a que almacena el conjunto de datos.

Los algoritmos *k-NN* cuentan con dos parámetros, la K que representa el número de vecinos y la k que es un parámetro de suavizado, donde mientras más grande sea k más pequeño es el ruido, y d es a cantidad de retardos para cada elemento.

SVM

Support Vector Machine o *SVM* es uno de los algoritmos de aprendizaje supervisado más populares, que se utiliza tanto para problemas de clasificación como de regresión. Sin embargo, principalmente, se utiliza para problemas de clasificación en aprendizaje automático.

El objetivo del algoritmo es encontrar un hiperplano en un espacio N-dimensional que clasifique claramente los puntos de datos. Para ello *Scikit-learn* tiene una funcionalidad integrada, el método *GridSearchCV* encargado de seleccionar hiperparámetros. Los hiperparámetros son variables que rigen el proceso de entrenamiento de un modelo, como el tamaño del lote o la cantidad de capas ocultas de una red neuronal profunda.

GridSearchCV se encarga de buscar hiperparámetros óptimos y, por lo tanto, mejorar los resultados de precisión/predicción. [27]

Ventajas del algoritmo:

- Eficaz en espacios de altas dimensiones.
- Efectivo en casos donde el número de dimensiones es mayor que el número de muestras.

- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores de soporte), por lo que también es eficiente en memoria.

auto-sklearn

Auto-sklearn es un conjunto de herramientas de *Machine Learning* de código abierto, además de ser un reemplazo directo para los clasificadores de *scikit-learn*.[28]

Podemos decir que se libera al usuario de elegir los algoritmos y el ajuste de los hiperparámetros, ya que proporciona aprendizaje automático supervisado basándose en la optimización bayesiana, el meta-aprendizaje y la construcción de conjuntos.

Auto-sklearn envuelve un total de 15 algoritmos de clasificación, 14 algoritmos de preprocesamiento de funciones y se ocupa del escalado de datos, la codificación de parámetros categóricos y los valores faltantes. Se puede ampliar fácilmente con nuevos métodos de clasificación, regresión y preprocesamiento de funciones. Para hacerlo, un usuario debe implementar una clase contenedora y registrarla para *auto-sklearn*.

TabPFN

Se ha decidido incluir en el proyecto un nuevo método clasificador llamado *TabPFN*. Es un transformador[19][29] capacitado que puede realizar una clasificación supervisada para pequeños conjuntos de datos en menos de un segundo, no necesita ajuste de hiperparámetros y es competitivo con los métodos de clasificación más avanzados.

En la web donde se presenta *TabPFN* se indica que el método supera claramente a los árboles potenciados con un aumento de velocidad de hasta 70 veces.

Este método al ser bastante novedoso por ahora limita los problemas hasta 1000 ejemplos de capacitación, 100 funciones y 10 clases. En un futuro el clasificador irá aumentando las limitaciones que presenta a día de hoy.

[19]Tipo de algoritmo de Deep Learning

3.7. Proceso de lectura y procesamiento de la señal del lector

Se ha realizado la extracción de características de los objetos utilizados para el entrenamiento de los modelos clasificadores del presente proyecto. Las características y demás atributos nos darán la posibilidad de identificar los distintos materiales por los que están compuestos los objetos. Estos datos están representados por números complejos, por lo que serán extraídos de las lecturas y separados en módulo y fase para poder realizar cálculos correctos.

Inicialmente se realizaron pruebas de extracción desde la herramienta facilitada por *Acconeer*, figura 3.5.

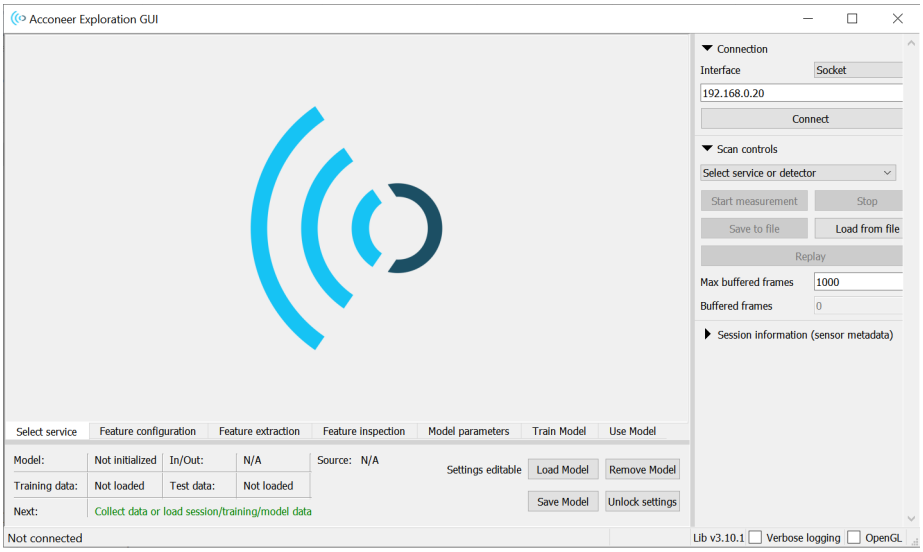

Figura 3.5: Herramienta *Acconeer*

Ejemplo de como se muestra una lectura en la interfaz, figura 3.6.

Figura 3.6: Lectura realizada en la herramienta de *Acconeer*.

La aplicación *Acconeer Exploration Tool* se puede descargar desde el repositorio de este proyecto (*acconeer-python-exploration*). En esta interfaz se conecta utilizando el modo *socket* a la IP de la *Raspberry Pi 4*.

Una vez ejecutada la aplicación hay cuatro servicios y varias funcionalidades. Los servicios son:

- *Power Bins*: según https://acconeer-python-exploration.readthedocs.io/en/latest/services/pb.html, calcula la energía de diferentes distancias, su objetivo es la medición de objetos grandes a distancias cortas como un sensor de parking. Al ser un método muy sencillo con poca cantidad de datos y enfocado a grandes objetos como una pared no se ha contemplado para el estudio.

- *Envelope*: según https://acconeer-python-exploration.readthedocs.io/en/latest/services/envelope.html, es igual que Power Bins pero utilizando un espectro continuo de los datos. Su caso de uso típico es la detección estática, por esto se ha contemplado para el estudio ya que la basura estará quieta.

- *IQ*: según https://acconeer-python-exploration.readthedocs.io/en/latest/services/iq.html, utiliza la coherencia de fase del radar que detecta movimiento a nivel fino. Tiene cinco modos, el primero es el que ha sido relevante al detectar mejores reflejos de los datos y estar optimizado para distancias muy cortas.

- *Sparce*: según https://acconeer-python-exploration.readthedocs.io/en/latest/services/sparse.html, se basa en la señal está más cuantizada y su uso principal es el análisis del movimiento y no de situaciones estáticas, por ello no se ha contemplado su uso para el estudio.

Para la creación del modelo se han seleccionado 30 materiales divididos en, 10 de plástico, 10 de cristal y 10 de cartón. De cada material se han realizado 10 lecturas, de varias caras, girando el objeto.

Para unificar el proyecto y no depender de la herramienta de *Acconeer* se ha decidido realizar la extracción de características de materiales desde un cuaderno de *Jupyter* (*LecturasRadar.ipynb*), estableciendo una configuración y un estándar en el radar para un correcto tratamiento de datos.

Se ha establecido el radar con los siguientes parámetros:

- Servicio *IQ*
- Perfil 2 (distancias de unos 20 cm)
- Tasa de lectura de 30 Hz
- Normalización desactivada (solo se recomienda activada para representar datos)
- Ganancia medio punto, la ganancia es la intensidad de la señal.

Llegamos a una colección de 300 lecturas exportadas cada una en ficheros con formato *numpy* (.npy) donde están almacenadas las características en vectores y matrices. Un porcentaje de estos datos conformarán la red de entrenamiento y otro porcentaje servirán para testear la red. En el directorio *GitHub* del proyecto se incluye una carpeta llamada «Materiales» que contiene los 300 archivos originales.

Cada instante de tiempo de la lectura comprende 291 atributos, de cada fichero obtenemos del orden de 300 instancias. Una instancia son estadísticas de los 291 atributos calculadas a partir de los 300 instantes de tiempo establecidos.

3.8. Transformación de las características

Una vez extraídas las características de los materiales se han realizado una serie de transformaciones en las lecturas, estas devuelven datos mostrados como números complejos.

Se creó una función que a partir del nombre del fichero *npy* cargue los datos, los redimensione y los devuelva.

Función *get_data* devuelve un array de 2 dimensiones (2D) al que llamaremos *datos_bruto*.

```
def get_data(url):
    try:
        diccionario = np.load(url, allow_pickle=True).item()
        data = diccionario.get('sweep_data').get('data')
        data = data.reshape(data.shape[0],data.shape[2])
    except:
        data = np.load(url)
    return data
```

Figura 3.7: Función *get_data*

Una función *get_modulo_fase* que a partir del array anterior, obtenga un array 2D, con el doble de anchura. Por ejemplo, si *datos_bruto* es de $300x291$, *modulo_fase* será de $600x291$.

Esa función obtendrá el módulo del número complejo y la fase del número complejo. El módulo será una matriz de $300x291$ y la fase también. Será necesario concatenarlas «horizontalmente».

```
def get_modulo_fase(data):
    modulo = abs(data)
    fase = []
    for i in data:
        fila = []
        for j in i:
            fila.append(phase(j)) #Fase
        fase.append(fila)
    fase = np.asarray(fase)
    modulo_fase = np.concatenate((modulo, fase), axis=0)
    return np.asarray(modulo_fase)
```

Figura 3.8: Función *get_modulo_fase*

A partir de los datos del modulo y la fase se obtiene la media. La comprobación sería que devuelva un array de 1 dimensión de tamaño 582 ($291x2$).

```
def get_media(modulo_fase):
    traspuesta = np.transpose(modulo_fase) #Obtenemos la matriz traspuesta(291x600)

    #Separamos la traspuesta en modulo y fase
    t_modulo = traspuesta[:,:int(traspuesta.shape[1]/2)]
    t_fase = traspuesta[:,int(traspuesta.shape[1]/2):]

    media = []
    for i in t_modulo:
        media.append(i.mean())
    for j in t_fase:
        media.append(j.mean())

    return np.asarray(media) #Devuelve medias modulo y fase
```

Figura 3.9: Función *get_media*

Una vez se han transformado las característica se decide utilizar un método de selección de estas.

3.9. Métodos de selección de características

En esta sección se indican los métodos de selección de características que se han utilizado.

Validación cruzada

La validación cruzada, o *cross validation*, es una técnica para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre los datos de entrenamiento (train) y prueba (test).

Este método lo utilizaremos para validar y comparar los modelos de aprendizaje automático elegidos.

El procedimiento elegido para crear las particiones de entrenamiento y test como indica la validación cruzada es el siguiente.

Se ha decidido crear una función (*particiones*) encargada de realizar automáticamente las particiones separadas en entrenamiento (*part_train*) y test (*part_test*). El desarrollo del código lo podemos encontrar en el fichero *GenerarParticiones.ipynb*.

Podemos decir que se realiza una validación cruzada dejando uno fuera. Se crean diez particiones que se corresponden con las diez vistas de cada

material. Por lo que en la partición N de entrenamiento se deja fuera la N vista de cada material, y estas vistas excluidas para entrenamiento pasarán a la partición de test.

Resumen de la creación de particiones:

```
Partición 1
Entrenamiento: [Carton Vista2-Vista10] + [Plastico Vista2-Vista10]
+ [Cristal Vista2-Vista10]
Test: [Carton Vista1] + [Plastico Vista1] + [Cristal Vista1]

Partición 2
Entrenamiento: [Carton Vista1,Vista3-Vista10] + [Plastico V1, Vista3-Vista10]
+ [Cristal Vista1,Vista3-Vista10]
Test: [Carton Vista2] + [Plastico Vista2] + [Cristal Vista2]

...

Partición 10
Entrenamiento: [Carton Vista1-Vista9] + [Plastico Vista1-Vista9]
+ [Cristal Vista1-Vista9]
Test: [Carton Vista10] + [Plastico Vista10] + [Cristal Vista10]
```

Bucles utilizados para crear las particiones:

```
for x in objetos: #Bucle para crear las 10 particiones
    print("Particion:", z)
    train = [] #conjunto entrenamiento
    test = [] #conjunto de pruebas
    for y in materiales: #Bucle de los materiales = 3
        for i in objetos: #Bucle de los objetos = 10
            #obj = [] #Lista de las vistas de un objeto
            for j in vistas: #Bucle de las vistas = 10
```

Figura 3.10: Bucle función *particiones*

Una vez tenemos las diez particiones creadas procedemos a entrenar los algoritmos. Los algoritmos utilizados en el proyecto *Random Forest*, *k-NN*, *SVM* y *auto-sklearn* se han entrenado mediante validación cruzada. Para entrenar *TabPFN* a parte de la validación cruzada se han tenido que

reducir los atributos a 100 (número máximo que permite *TabPFN*) mediante *Pipeline* junto con *SelectFromModel*.

Pipeline

Pipeline se ha empleado para reducir los atributos que se han empleado para entrenar el clasificador *TabPFN*. Este clasificador actualmente limita los problemas hasta 1000 ejemplos de capacitación, 100 funciones y 10 clases. En un futuro el clasificador irá aumentando las limitaciones que presenta a día de hoy.

SelectFromModel se usa para extraer las mejores características de los conjuntos de datos y requiere un estimador, en este caso la clase*LogisticRegression* es el modelo escogido para que *Pipeline* seleccione las 100 características para entrenar a *TabPFN*.

```
tabPFNClassifier_attsell = Pipeline([('att_sel', SelectFromModel(estimator=LogisticRegression(solver='newton-cg', max_iter=10000),
                                                 threshold=-np.inf,
                                                 max_features = 100)),
                        ('clasificador', TabPFNClassifier(device='cpu', N_ensemble_configurations=300))])
```

Figura 3.11: Pipeline

Usamos $threshold = -np.inf$ para que ignore el valor de la importancia de las características y use el número establecido en $max\_features$ (número máximo de características).

Técnicas y herramientas

4.1. Técnicas y metodologías

Metodología SCRUM

Se trata de una metodología de trabajo ágil que tiene como finalidad dividir en periodos de tiempo el flujo de trabajo. Estos periodos son conocidos como *sprints*, al finalizar cada *sprint* se realizan revisiones y reuniones donde se deciden las tareas de los próximos *sprints*.

Cliente de control de versiones

- Herramientas consideradas: *GitHub Desktop* y *Gitkraken*
- Herramienta elegida: *GitHub Desktop*

GitHub Desktop utilizado para la gestión ágil del proyecto.

Hosting del repositorio

- Herramientas consideradas: *GitLab* y *GitHub*
- Herramienta elegida: *GitHub*

GitHub ofrece una gran cantidad de facilidades para mantener el proyecto en la nube y debido a que no cobra por sus servicios lo convierte en la mejor opción posible. Nos permite alojar nuestro repositorio central del proyecto usando el control de versiones *Git*.

Git es un sistema de control de versiones distribuido de código abierto. El control de versiones nos permitirá retornar a algún punto anterior del desarrollo de nuestra aplicación en caso de sufrir errores.

Url de la herramienta: https://github.com/

4.2. Lenguajes y bibliotecas

Python

El lenguaje de programación *Python* se diferencia por su código el cual es legible y limpio esto hace que sea uno de los lenguajes de iniciación de muchos programadores. Además, se trata de un lenguaje de multiparadigma y multiplataforma muy utilizado en la técnica del *BigData*.

Url de la herramienta: https://www.python.org/

NumPy

NumPy es una biblioteca utilizada en la programacion con *Python* para crear vectores y matrices grandes multidimensionales.

La funcionalidad principal de *NumPy* es su estructura de datos *«ndarray»*, para una matriz de n dimensiones.

El uso de *NumPy* en *Python* brinda una funcionalidad comparable a *MATLAB*.

Url de la herramienta: https://numpy.org/

Pandas

Pandas es una biblioteca de código abierto especializada en el manejo y análisis de estructuras de datos flexible y fácil de usar, construida sobre el lenguaje de programación *Python*.

Su objetivo es ser el bloque de construcción fundamental de alto nivel para realizar análisis de datos prácticos del mundo real en *Python*.

Pandas está construido sobre *NumPy* y está destinado a integrarse bien dentro de un entorno informático científico con muchas otras bibliotecas de terceros.

Url de la herramienta: https://pandas.pydata.org/

acconeer.exptool

acconeer.exptool es la biblioteca facilitada por la empresa *Acconeer*. Con ella pondremos en funcionamiento el radar de 60GHz además de recopilar los datos necesarios.

Url de la herramienta: https://github.com/acconeer/acconeer-python-exploration

Sklearn

Scikit-learn es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación de *Python*. En esta biblioteca encontramos varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, *Gradient boosting*, *K-means*...

Url de la herramienta: https://scikit-learn.org/

Auto-sklearn

Auto-sklearn es un kit de herramientas de aprendizaje automático automatizado de código libre y creado a partir de *Scikit-learn*.

En esta biblioteca encontramos un total de 15 algoritmos de clasificación, 14 algoritmos de preprocesamiento de funciones además se ocupa del escalado de datos, la codificación de parámetros categóricos y los valores faltantes.

Url de la herramienta: https://automl.github.io/auto-sklearn/master/

TabPFN

TabPFN método clasificador de software libre que puede realizar una clasificación supervisada para pequeños conjuntos de datos en menos de un segundo.

Url de la herramienta: https://www.automl.org/tabpfn-a-transformer-that-solves-small-tabular-classification-problems-in-a-second/

Tkinter

Tkinter es una adaptación de la biblioteca gráfica *Tcl/Tk* [30] para el lenguaje de programación *Python*. Mediante *Tkinter* se desarrollará la interfaz gráfica para comparar distintos tipos de materiales.

Url de la herramienta: https://docs.python.org/es/3/library/tkinter.html

cmath

Módulo incorporado en *Python* que usaremos para tareas matemáticas. Las funciones de este módulo aceptan números enteros, números de coma flotante o números complejos como argumentos.

Url de la herramienta: https://docs.python.org/3/library/cmath.html

Socket

Es una librería que proporciona acceso a la interfaz *BSD*[20] *socket*. En el presente proyecto se utilizará para realizar las comunicaciones con el radar.

Url de la herramienta: https://docs.python.org/es/3.10/library/socket.html

Paramiko

Paramiko es un modulo de *Python* utilizado para control remoto. En este proyecto nos ayudaremos de esta biblioteca para ejecutar comandos en el radar.

Url de la herramienta: https://www.paramiko.org/

matplotlib

Matplotlib es una biblioteca para la generación de gráficos a partir de datos almacenados en listas o arrays. Se empleará para visualizar gráficamente los resultados de la clasificación.

Url de la herramienta: https://matplotlib.org/

joblib

Joblib es un conjunto de herramientas encargadas de realizar una canalización ligera en *Python* mejorando la rapidez del tratamiento de los datos.

[20]Distribución de software Berkeley.

Url de la herramienta: https://docs.python.org/3/library/cmath.html

4.3. Herramientas de desarrollo

Jupyter Notebook

Jupyter Notebook es el entorno de trabajo utilizado en el proyecto que permite desarrollar código en *Python* de manera dinámica. Nos ofrece integrar en un mismo archivo bloques de código, texto, gráficas o imágenes. Utilizado ampliamente en análisis numéricos y estadísticos.

Jupyter admite más de 40 lenguajes de programación, incluidos *Python, R, Julia* y *Scala*.

Url de la herramienta: https://jupyter.org/

Colab

Colab o *Colaboratory* es un producto de *Google Research*. Permite a cualquier usuario escribir y ejecutar código arbitrario de *Python* en el navegador, es una alternativa a *Jupyter Notebook*.

Se ha utilizado para ejecutar *auto-sklearn* y *TabPFN*. Debido a que para utilizar *auto-sklearn* se necesita un crea un entorno en *Ubuntu* y *TabPFN* estaba dando problemas de instalación en el equipo utilizado.

Url de la herramienta: https://colab.research.google.com/

4.4. Herramientas de documentación

LaTeX

LaTeX es un sistema de elaboración de documentos basado en comandos. El proyecto está desarrollado mediante una plantilla creada en LaTeX.

Url de la herramienta: https://jupyter.org/

MiKTeX

MiKTeX es una distribución de LaTeX encargada de gestionar los componentes y paquetes. Tiene la capacidad de actualizarse así mismo descargando nuevas versiones de componentes.

Url de la herramienta: https://miktex.org/

Texmaker

Texmaker es el editor de documentos LaTeX utilizado para funcionar necesita *MiKTeX*.

Url de la herramienta: https://www.xm1math.net/texmaker/

BibItNow!

BibItNow! es la extensión del navegador que utilizamos para citar páginas web en el formato *Bibtex*, formato soportado por LaTeX.

Url de la herramienta: https://chrome.google.com/webstore/detail/bibitnow/bmnfikjlonhkoojjfddnlbinkkapmldg

Aspectos relevantes del desarrollo del proyecto

En este apartado se va a recoger el ciclo de vida del proyecto, detallando los aspectos más relevantes que se han tratado y como se han resuelto las diferentes dificultades encontradas a lo largo de su desarrollo. Se irán presentando diferentes secciones que concuerdan con el orden cronológico seguido en el proyecto y muestran la justificación de las decisiones tomadas.

5.1. Propuesta del Proyecto

La propuesta de este proyecto consistía en crear un modelo clasificador integrado en una iterfaz para el reconocimiento de materiales que se dividía en las siguientes 3 tareas principales:

- **Montaje del sensor:** Puesta en funcionamiento del radar junto con sus componentes. Siendo capaces de extraer las características del las lecturas realizadas.

- **Lecturas de los materiales:** Conseguir información a través de las características de los materiales para crear un modelo capaz de diferenciar materiales como son: cartón, plástico y cristal.

- **Clasificador de materiales:** La tarea principal pedida para realizar este proyecto era la de construir un clasificador de materiales a través de un radar de 60GHz devolviendo la identificación correcta del material.

5.2. Metodologias aplicadas

En el desarrollo de este proyecto se ha intentado emplear en la medida de los posible la metodología ágil de *Scrum*.

Debido a que el proyecto se comenzó en abril de 2021, no se ha podido seguir de forma estricta todas las pautas de la metodología, como las reuniones diarias con todo el equipo.

Las pautas que se han seguido durante el proyecto han sido:

- Desarrollo incremental del proyecto mediante *sprints*.
- *Spints* de 1 a 2 semanas.
- Al finalizar el *sprint*, reuniones para evaluar el proyecto y plantear los pasos a seguir en el siguiente *sprint*.

5.3. Formación

Durante las primeras fases del proyecto, fue necesario aprender el funcionamiento del Radar Sensor. Para ello, se consultó la documentación facilitada por el fabricante *Acconeer*.

A parte de esto se investigó información adicional sobre el uso de estos radares, así como ejemplos de su uso, la información extraída de la red fue escasa o casi nula. Únicamente fue válida la documentación facilitada por *Acconeer*.

5.4. Montaje del radar

En esta sección se indican tanto las necesidades hardware como software para el montaje y puesta en funcionamiento del radar.

Necesidades hardware

Los creadores de *Acconner* han elaborado un vídeo (*EVK 2*) con las instrucciones del montaje del sensor. En él se detallan los diferentes componentes y el montaje de hardware a la *Raspberry Pi 4*.

En un comienzo será necesario disponer de:

- Raspberry Pi 4

- Radar A111
- Placa XR112
- Cable flexible para XR112
- Tarjeta SD
- Teclado USB
- Ratón USB
- Monitor con HDMI
- Cable HDMI
- Adaptador mini HDMI a HDMI

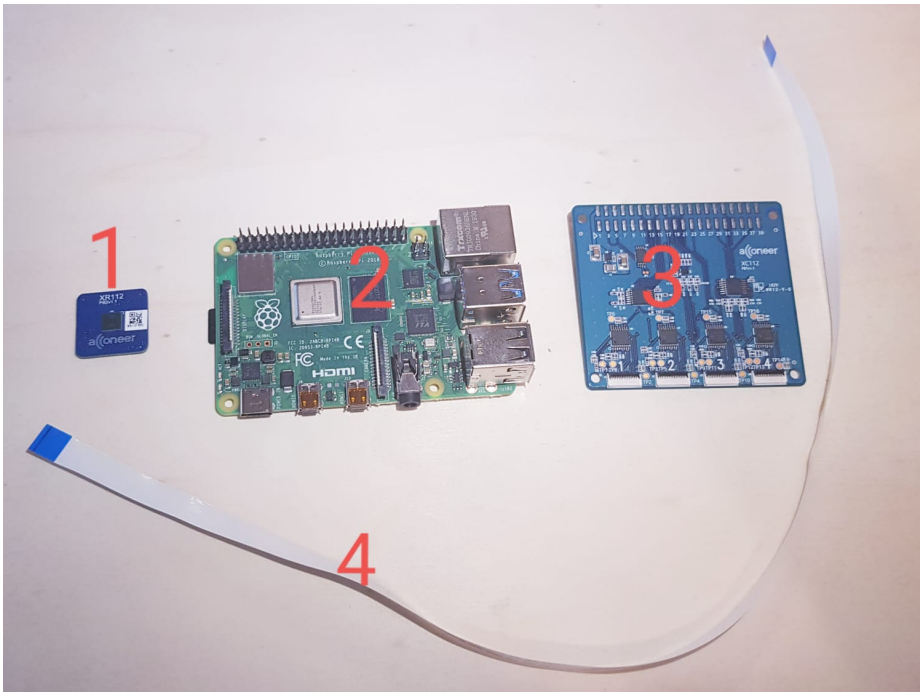

Figura 5.1: Componentes principales

La figura 5.1 muestra los siguientes componentes:

1 - Radar A111

2 - Raspberry Pi 4

3 - Placa XR112

4 - Cable flexible para XR112

La placa se conecta a la *Raspberry* por el puerto *GPIO* quedando por encima del resto.

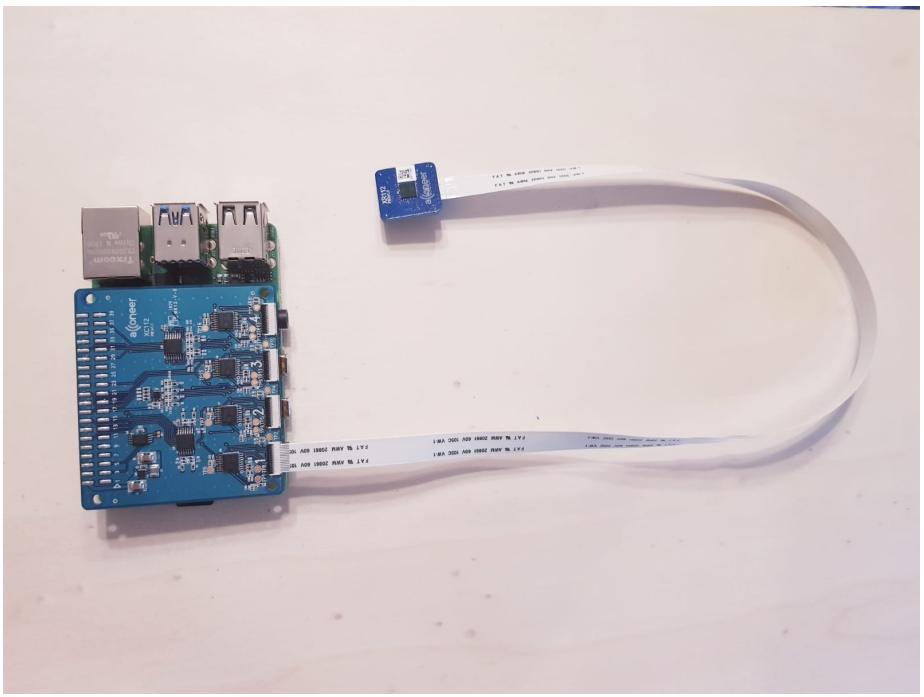

Figura 5.2: Componentes en conexión

Se ha realizado un espacio o cubículo de lectura de objetos de $30x22x25cm$ (ancho, largo y alto) con una incisión en la parte superior, en el medio, del tamaño del sensor. El sensor fue bloqueado a su posición solamente colocándole en la posición de la incisión y sujetado con cinta adhesiva. Esta caja de madera se utiliza poniendo la abertura en la parte lateral, mirando al operario que hace uso del radar.

El sensor tiene sensibilidad a partir de los 10 cms. La sensibilidad del sensor se limitó al rango de 10 a 24 cm, el mínimo de sensibilidad hasta el suelo.

Figura 5.3: Prototipo

Necesidades software

Una vez se había procedido con el montaje del dispositivo se pasó a la instalación y configuración del sistema operativo. Además, se instalaron los diversos programas necesarios tanto para recoger los datos como para poder interpretarlos.

La tarjeta SD facilitada por la UBU ya disponía del sistema operativo necesario para realizar las lecturas, tras esto se debía comenzar a investigar como ejecutar el radar desde un cuaderno de Python.

Para realizar una instalación desde cero del radar serán necesarios los siguiente programa. Además, nos pueden resultar útiles a la hora de trabajar con el Radar. Hay que descargar e instalar:

- *Acconeer SW*: disponible para todos los usuarios (*Windows, Linux, iOS*)
- *Raspbian OS*
- *Etcher*: se usará para instalar el sistema operativo *Raspbian* en la tarjeta SD
- *PuTTY*: se utiliza para conectarse a *Raspberry Pi*
- *WinSCP*: utilizado para transferir archivos a la tarjeta SD

5.5. Lectura de los objetos

Para la creación del modelo se han seleccionado 30 materiales divididos en, 10 de plástico, 10 de cristal y 10 de cartón. De cada material se han realizado 10 lecturas, de varias caras, girando el objeto.

Llegamos a una colección de 300 lecturas exportadas cada una en ficheros con formato *numpy* (.npy) donde están almacenadas las características en vectores y matrices. Un porcentaje de estos datos conformaran la red de entrenamiento y otro porcentaje servirán para testear la red.

Estas lecturas se han realizado desde el cuaderno de Jupyter «Lecturas-Radar».

```
iq = configs.IQServiceConfig()
#envelope = configs.EnvelopeServiceConfig()
iq.range_interval=[0.10, 0.24]
iq.profile = 'PROFILE_2' #Perfil 2
iq.update_rate = 30 # 30 Hz
iq.noise_level_normalization = 0 #Normalizacion desactivada
iq.gain = 0.5 #Ganancia [0-1]
#envelope.range_interval=[0.10, 0.24]
raspi = clients.SocketClient(ip)
raspi.connect()
raspi.start_session(iq) # o envelope
```

Figura 5.4: Configuración inicial.

Se ha establecido el radar con los siguientes parámetros:

- Indicamos el *hostname* (RadarAcconeer) y el modo de conexión mediante *socket*
- Servicio IQ
- Perfil 2 (distancias de unos 20 cm)
- Tasa de lectura de 30 Hz
- Normalización desactivada (solo se recomienda activada para representar datos)
- Ganancia medio punto, la ganancia es la intensidad de la señal.

Una vez establecida la configuración debemos tener el radar encendido y conectado a la misma red que nuestro equipo. Tras esto vamos ejecutando el cuaderno para realizar las lecturas de los objetos. Como el intercambio de los objetos para la lectura es manual la ejecución del cuaderno también la haremos manual y tomando los tiempos oportunos.

Para poder obtener las lecturas debemos ejecutar el fichero indicado en la figura 5.5. Esta obligatoriedad de iniciar el servicio del radar se ha implementado en el desarrollo creado para este proyecto facilitando la ejecución de las lecturas.

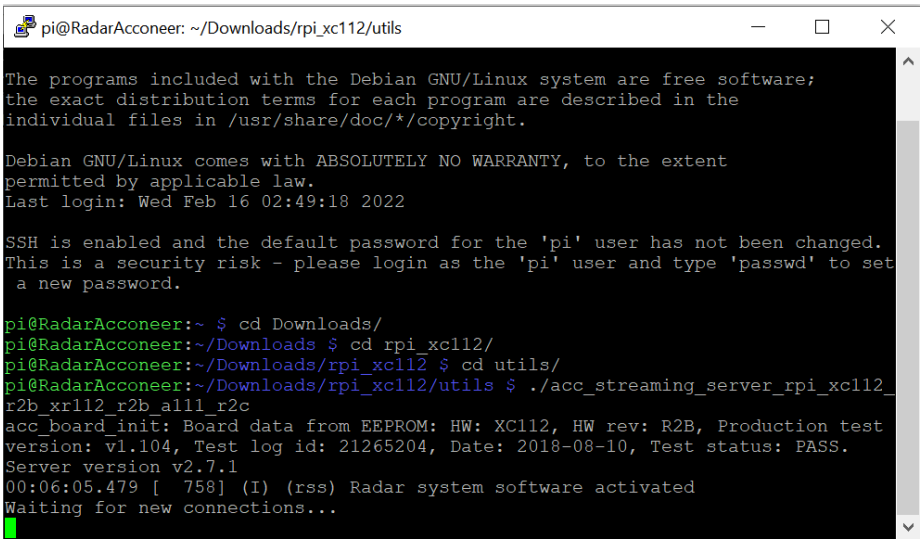

Figura 5.5: Terminal del radar.

Cada instante de tiempo de la lectura comprende 291 atributos, de cada fichero obtenemos del orden de 300 instancias. Una instancia son estadísticas de los 291 atributos calculadas a partir de los aproximadamente 300 instantes de tiempo.

Una vez finalizadas las lecturas se procedió al procesado de los datos de cada lectura para finalmente unirlos en datos de entrenamiento y test.

Tenemos los siguientes datos:

- N es el numero de lecturas (1 objeto se corresponde con 10 vistas)
- M son los instantes de tiempo
- A referencia a los atributos

Se creará la siguiente matriz:

$$Matriz1 = (N \cdot M) \cdot A \tag{5.1}$$

Es una matriz 2D $(N \cdot M, A)$

La extracción y creación de la matriz con los datos necesarios se realizará mediante *Python*:

```
diccionario = np.load('C01_V01.npy',allow_pickle=True).item()
data = diccionario.get('sweep_data').get('data')
data = data.reshape(data.shape[0],data.shape[2])
```

Los datos de la lectura tienen una parte real y otra imaginaria, a partir del array 2D anterior, se obtiene otro array 2D, con el doble de anchura. Por ejemplo, si Matriz1 es de $300x291$, se obtendrá otra matriz que comprende el modulo y la fase de $600x291$. Así se obtiene el módulo del número complejo y la fase del número complejo. El módulo será una matriz de $300x291$ y la fase también.

```
modulo = abs(data)
fase = []
for i in data:
fila = []
for j in i:
fila.append(phase(j)) #Fase
fase.append(fila)
      fase = np.asarray(fase)
      modulo_fase = np.concatenate((modulo, fase), axis=0)
```

A partir de los datos del modulo y la fase se obtiene la media.

```
traspuesta = np.transpose(modulo_fase)
#Obtenemos la matriz traspuesta(291x600) de modulo_fase(600x291)

#Separamos la traspuesta en modulo y fase
t_modulo = traspuesta[:,:int(traspuesta.shape[1]/2)]
t_fase = traspuesta[:,int(traspuesta.shape[1]/2):]
media = []
for i in t_modulo:
```

```
     media.append(i.mean())
for j in t_fase:
     media.append(j.mean())
#media -> medias modulo y fase
```

Una vez obtenidos los datos de las lecturas se comenzó con la creación de cuatro modelos clasificadores que más adelante se usarán para realizar la clasificación de los materiales junto con el radar.

Dentro del repositorio de *GitHub* del proyecto hay un fichero llamado *GenerarParticiones.ipynb* que incluye el desarrollo de todas estas funciones así como comentarios incisivos.

5.6. Modelo clasificador

Para elegir el modelo a usar se tuvieron que tener en cuenta diferentes factores como el peso del modelo, su carga de computo y su precisión.

Se realizaron pruebas con los diferentes modelos *Random Forest*, *k-nearest neighbors (k-NN)*, *Support Vector Machine (SVM)*, *auto-sklearn* y *TabPFN*.

Todos los modelos mantienen el mismo patrón de ejecución realizando una validación cruzada manual.

A continuación se incluyen los resultados obtenidos de entrenar cada clasificador, el código desarrollado para cada modelo lo encontramos en el repositorio de proyecto https://github.com/mecyc/TFG_RADAR_60GHZ/tree/main/scripts.

Random Forest

Con el método *Random Forest* se ha llegado a una tasa de acierto del 86.33 % estableciendo el número de árboles en 100.

En la figura 5.6 se muestra la correspondiente matriz de confusión, junto con con la tasa de acierto promedio después de haber entrenado al modelo. El test se ha realizado en 10 particiones de 30 ficheros cada partición siguiendo la linea de la validación cruzada.

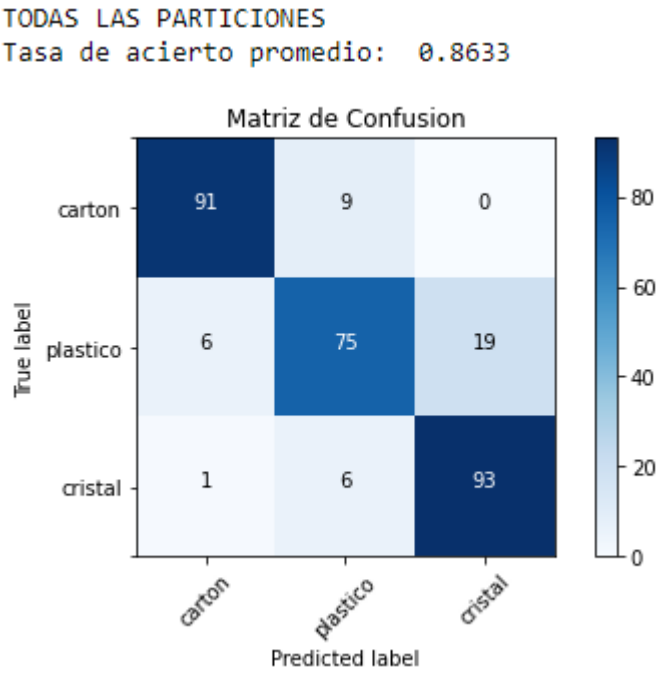

Figura 5.6: Matriz de confusión *Random Forest*.

Podemos ver que la confusión más frecuente son los objetos de plástico predichos erróneamente como cristal (19).

k-NN

En la ejecución del clasificador *k-NN* se ha decidido establecer un rango de 1 a 26 vecinos, rango que debería dar buenos resultados. Este método supone que los vecinos más cercanos nos dan la mejor clasificación y así ha sido, la mejor tasa de ha acierto ha sido con un vecino, una tasa del 87.67 %.

En la figura 5.7 se muestra la correspondiente matriz de confusión para 1 vecino. El rango de vecinos ha sido establecido de 1 a 26, el algoritmo en 1 vecino ha obtenido la mejor tasa de acierto.

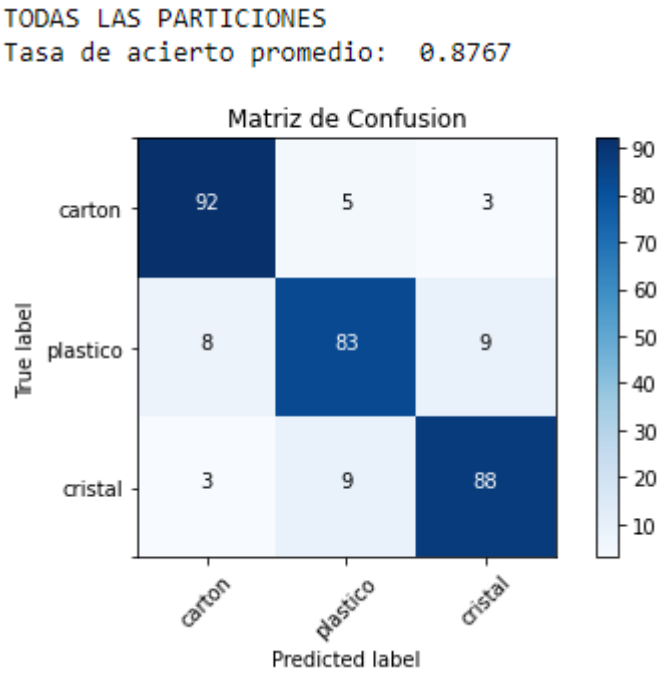

Figura 5.7: Matriz de confusión *k-NN*.

Según la gráfica de confusión obtenida vemos que los resultados son bastante parejos, obteniendo el mayor acierto en el cartón pero las confusiones son parejas entre los tres materiales. No ha sido la mayor tasa de acierto obtenida pero podemos decir que es la matriz de confusión con la diagonal más uniforme.

Gráfico que muestra la evolución del clasificador en el rango de 1 a 26 vecinos:

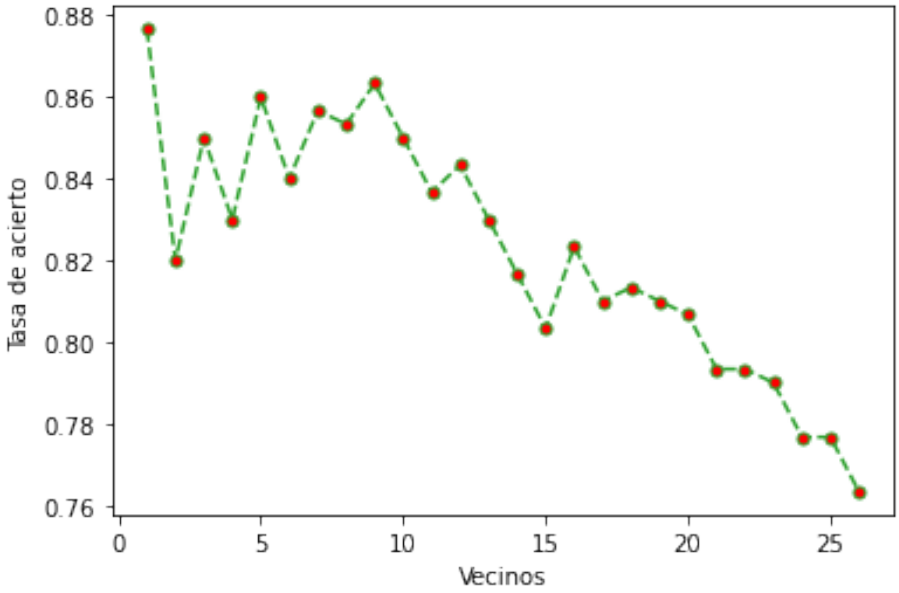

Figura 5.8: Evolución clasificador *k-NN*.

SVM

En la ejecución del clasificador *SVM* se ha decidido utilizar la clase *GridSearchCV* para la búsqueda de parámetros. La tasa de acierto obtenida ha sido del 84.67 %.

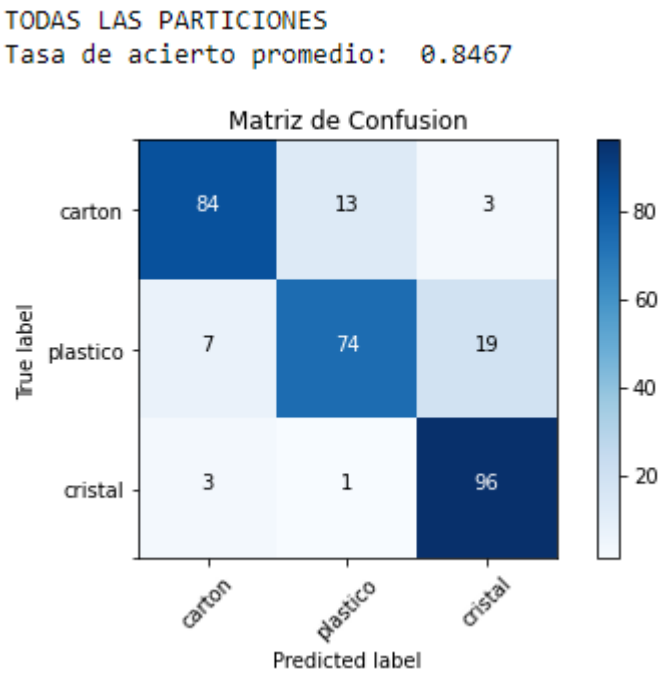

Figura 5.9: Matriz de confusión *SVM*.

SVM nos da la matriz de confusión con menos tasa de acierto de los clasificadores utilizados. Obtiene una gran tasa de acierto en el cristal pero a la vez mantiene una confusión elevada al predecir el plástico (19) y el cartón (13).

auto-sklearn

En la ejecución del clasificador *auto-sklearn* se ha establecido el límite de tiempo para la búsqueda de modelo en 30 segundos y el límite para cada llamada en 120 segundos, ejecutando el modelo aumentando los segundos no ha mejorado la tasa de acierto. La tasa de acierto obtenida ha sido del 88 %

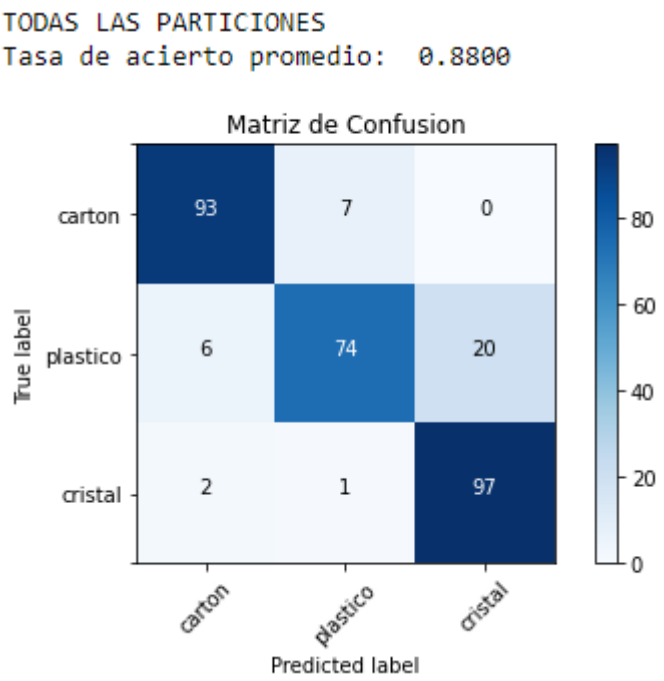

Figura 5.10: Matriz de confusión *auto-sklearn*.

Muestra una matriz de confusión muy parecida a la de *Random Forest*, un acierto muy elevado en el cartón y cristal pero en el plástico mantiene 20 predicciones confusas.

TabPFN

TabPFN ha sido el último modelo incluido en el proyecto así como el más novedoso. Debido a la limitación de 100 características que presenta se ha decidido utilizar *Pipeline* de *Sklearn* para reducir de 582 características a 100. *Pipeline* es una secuencia de mecanismos de procesamiento de datos.

LogisticRegression es el modelo escogido para que *Pipeline* seleccione las 100 características para entrenar a *TabPFN*.

La tasa media de acierto obtenida ha sido la mejor hasta el momento llegando al 90 %.

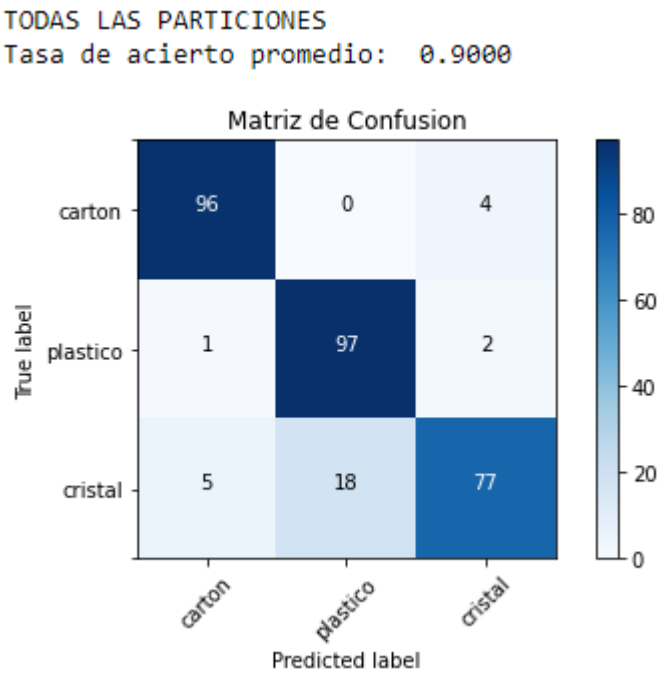

Figura 5.11: Matriz de confusión *TabPFN*.

TabPFN nos ha dado la mayor tasa de acierto pero a decir verdad la matriz de confusión no ha sido la esperada, la confusión de indicar plástico (18) al cristal ha sido demasiado frecuente.

Comparativa de clasificadores

Viendo en la tabla 5.1 se puede decir que en todos los modelos de clasificación hemos obtenido una tasa de acierto entre el 80 % y 90 % sin obtener unos resultados muy dispares.

Modelo	**Tasa de acierto (%)**
Random Forest	86.33
k-NN	87.67
SVM	84.67
auto-sklearn	88
TabPFN	90

Tabla 5.1: Comparativa de modelos clasificadores.

En nuestro caso, el método más sencillo (*k-NN*) ha proporcionado resultados competitivos, hasta el punto de ser el tercer mejor método. *Auto-sklearn* ha obtenido el segundo mejor resultado con el 88 %. El mejor resultado lo ha dado *TabPFN* alcanzando un 90 % de tasa de acierto.

k-NN ha mostrado la matriz de confusión con menor confusión por cada material, es decir con la diagonal más uniforme. Esto no quiere decir que con algoritmos más elaborados y que usan más parámetros obtengamos malos resultado. Con el clasificador *auto-sklearn* se ha obtenido la misma tasa de acierto que con *k-NN*, *auto-sklearn* es un modelo más elaborado que se basa en el auto aprendizaje y ajuste de hiperparámetros.

Por último decir que *TabPFN* cumple las expectativas indicadas por los creadores de este modelo y ha superado en tasa de acierto con el 90 % a todos los clasificadores mostrados en el presente proyecto. *TabPFN* es mejor en general, pero no es tan bueno al clasificar objetos que son de plástico y confunde muchos con cristal.

Según el uso que se quiera dar al sistema habría que elegir entre un clasificador u otro. Por ejemplo, evitar residuos impropios en el contenedor de plástico.

En un principio lo ideal habría sido usar un modelo obtenido a partir del clasificador *TabPFN* por innovación. Pero se ha decidido optar por *k-NN* para generar el modelo que utilizará la interfaz desarrollada. Esta decisión se debe a la falta de compatibilidad de *auto-sklearn* y *TabPFN* (basado en *auto-sklearn*) con el sistema operativo *Windows*. Actualmente *auto-sklearn*

es compatible con sistemas como *Linux* y el desarrollo creado se quiere utilizar tanto en *Windows* como en *Linux*.

5.7. Interfaz

Una vez generado el fichero del modelo a utilizar, se generó la interfaz de la aplicación desarrollada también en *Python*.

Para la generación de la interfaz se ha utilizado la librería *Tkinter*, es una librería dedicada generalmente para el desarrollo de interfaces.

La aplicación generada para el proyecto ha sido llamada ***RadarWave***, debe su nombre a las ondas[21] generadas por los radares.

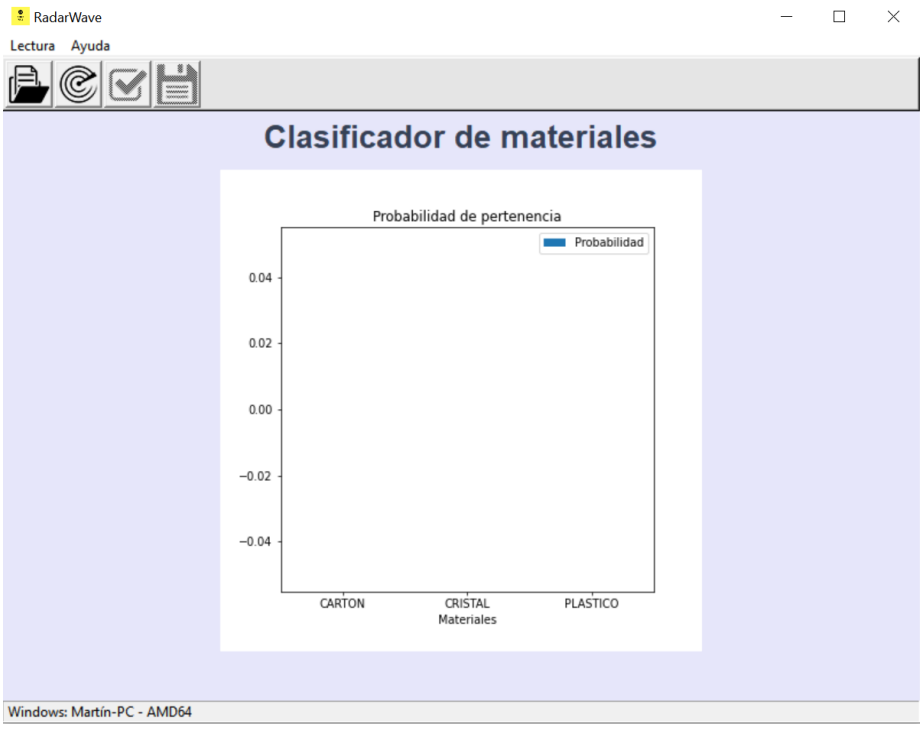

Figura 5.12: Ventana principal de RadarWave.

Como podemos ver en la figura 5.12 la aplicación principalmente se divide en:

- **Zona superior:** están las barras de herramientas donde se encuentran los botones que utilizaremos para las lecturas de los materiales.

[21] Onda en inglés es wave.

- **Zona central:** tenemos la sección donde se mostrara la probabilidad de pertenencia del objeto a un tipo de material.
- **Zona inferior:** se ha decidido incluir una barra de estado con información sobre el equipo o PC. Esta barra de estado se puede utilizar en un futuro para mostrar datos de interés.

Volviendo a las barras de herramientas podemos apreciar dos filas, la superior que consta de dos menús desplegables *Lectura* y *Acerca de*. En la sección *Lectura* encontraremos todas las opciones posibles para clasificar objetos y en *Acerca de* encontramos información sobre la versión de la aplicación.

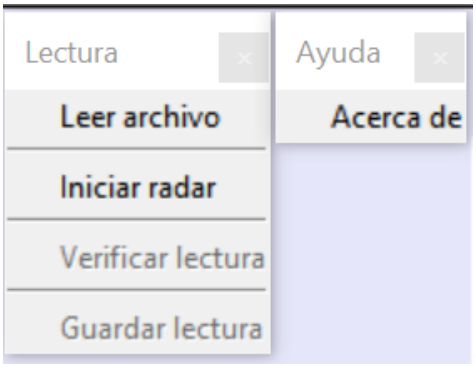

Figura 5.13: Menús desplegables de RadarWave.

Un poco mas abajo llama la atención una barra de herramientas con iconos, esos iconos se corresponden con el menú desplegable de *Lectura*. Cada icono se corresponde con un item del menú. Los botones de izquierda a derecha se corresponden con las funcionalidades de abrir fichero de lecturas ya generadas, iniciar lectura por radar, clasificar objeto y guardar lectura generada.

La aplicación es capaz de conectar con el radar sin tener que acceder a la configuración de este en ningún momento. Los únicos requisitos son tener conectado el radar a la corriente eléctrica y a la red de internet (mediante *Ethernet* o *Wifi*). La comunicación de la interfaz con el radar se ha generado principalmente con las librerías *Socket* y *Paramiko*.

Dentro del código fuente se han incluido la mayoría de las funciones utilizadas para el tratamiento de las lecturas iniciales de las vistas de los objetos, es decir se realiza una lectura in sito con el radar para extraer los

datos del objeto, estos datos se transforman y se tratan con las mismas funciones que fueron encargadas de generar los datos para la creación y entrenamiento del modelo.

El detallado del código de la aplicación se expone dentro del anexo *D - Documentación técnica de programación*.

Trabajos relacionados

En este apartado se muestran las herramientas relacionadas con el proyecto. Hoy en día hay poca información y documentación de este tipo de tecnología de radares en la red, por ello solo muestro una única herramienta.

6.1. *RadarCat*

RadarCat (*Radar Categorization for Input and Interaction*) permite que un dispositivo electrónico pueda reconocer y clasificar distintos materiales y objetos a tiempo real con alta precisión. Además da mucha más información de los objetos, como su estructura interna.

El escaner/radar utilizado envía ondas electromagnéticas al objeto, estas rebotan y vuelven al punto de partida para su procesamiento con aprendizaje automático o *machine learning*.

Es necesario un ordenador para inicializar el software de *RadarCat*, a este equipo se conecta una placa base donde se encuentran las antenas/radares capaces de realizar lecturas de objetos. Por último encontramos una placa que protege el sensor y soporta el objeto a reconocer.

Más allá de la interacción humana con la computadora, *RadarCat* también abre nuevas oportunidades en áreas como la navegación y el conocimiento mundial (por ejemplo, usuarios con baja visión).

Aquí se puede acceder a un vídeo interesante que muestra los múltiples usos de *RadarCat* https://www.youtube.com/watch?v=B6sn2vRJXJ4

En el proyecto *RadarCat* creado por *Google* utiliza un radar del fabricante *Infineon Technologies* al que *Google* llama *Google ATAP Project Soli*, nosotros

para este proyecto estamos utilizando el del fabricante *Acconeer*. Existe una gran diferencia de precios entre los dos, mientras que nuestro radar tiene un precio de 52,73€ el de *Infineon*, según muestra la web del fabricante, llega a valer 284,64€.

6.2. Exploración de interacciones tangibles con sensores de radar

En esta sección se expone un artículo que muestra un ejemplo de uso para sensores de radar y aprendizaje automático para la exploración de interacciones tangibles[22].[31]

El el artículo se proponen usar el radar como una plataforma de detección para rastrear la identidad de un objeto o la cantidad de objetos próximos para habilitar la *Tangible User Interface* (TUI). El radar mediante redes microondas ha sido capaz de reconocer, contar y ordenar objetos y además determinar movimientos (deslizamiento o rotación). Todo ello con elementos comunes, generalmente objetos en gran parte planos.

Para realizar el prototipo, se usa el sensor de radar *Google ATAP Project Soli*, por lo que este proyecto es una ampliación de *RadarCat*. Para mejorar el objetivo buscado se crean unas estructuras 3D, mediante una impresora, para guiar la onda del radar y mejorar la colocación de los objetos sobre este. Según indica el articulo se han creado hasta 12 estructuras distintas (p.ej. soporte para fichas *Lego*, tarjetero y un espacio de discos), cada una para un fin.

El prototipo se basa en el paradigma *Token+Constraint*. En estas interfaces, los *tokens* (fichas) son objetos físicos discretos que representan información digital. Las *constraint* (restricciones) son regiones de confinamiento que se asignan a las operaciones digitales. Estos se incorporan con frecuencia como estructuras que canalizan mecánicamente cómo se pueden manipular las fichas, a menudo limitando su movimiento a un solo grado de libertad. La colocación y manipulación de tokens dentro de sistemas de restricciones se puede usar para invocar y controlar una variedad de interpretaciones computacionales. Se muestra que al agregar un «caso guía» que actua como restriccion, se puede ampliar en gran medida las capacidades de interacción y mejorar la precisión de detección de objetos.

[22]El mundo digital no se observa mediante un monitor o la pantalla de un dispositivo móvil, sino que está mezclado con la realidad.

En el siguiente enlace se puede acceder a un vídeo donde se muestran varios ejemplos de uso del prototipo: https://www.youtube.com/watch?v=zDAJ_0KIovc

Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

En este trabajo se exploraron las posibilidades de utilizar un radar de ondas de 60 GHz para la clasificación de diferentes tipos de materiales. Se descubrió que extrayendo características mediante el radar y combinando esto con un modelo clasificador es posible realizar una clasificación para distinguir tres tipos de materiales comunes en la vida diaria como son PLÁSTICO, CRISTAL y CARTÓN.

Mediante una aplicación en fase Beta se ha conseguido realizar la lectura de las características de varios materiales y su correcta identificación.

Una posible aplicación sería automatizar la diferenciación de los tipos de embalajes en una empresa de envíos para su correcto reparto evitando los golpes.

Respecto a las dificultades encontradas, algunas han surgido por intentar abarcar una gran cantidad de lecturas de distintos objetos tanto en su estructura como en su forma, lo que ha significado tener un gran número de características dificultando la tarea de implementar un modelo clasificador que diese una gran exactitud a la hora de indicar el tipo de material.

7.2. Líneas de trabajo futuras

Posibles líneas de trabajo futuras relativas al uso del radar:

- Combinar el radar utilizado de *Acconeer* con un sensor radar del fabricante *Infineon Technologies*, posiblemente colocados en dos posiciones del habitáculo de lectura (un radar superior y otro lateral).
- Refactorización del código actual para su uso en otros sensores.
- Aplicar otros clasificadores.
- Añadir otros estadísticos.
- Automatizar el proceso de creación y entrenamiento del clasificador mediante una base de datos, así como la ampliación de la capacidad de reconocer materiales nuevos.

Bibliografía

[1] J. L. Alonso Cerpa, "IntroducciÓn a los sistemas radar y arpa," 2015.

[2] H. G. el al., "Herrera, g., tomás, r., lópez sánchez, j. m., oriol, m., cooksley, g., & mulas de la peña, j. (2009). sistemas radar aplicados a la investigación de subsidencia y movimientos de ladera. enseñanza de las ciencias de la tierra, 17(3), 316–324. - buscar con google," 2009.

[3] Dipl.-Ing. f. C. Wolff, "Fundamentos de radar - ERS–1, (2)," Dec. 2022. [Online; accessed 16. Dec. 2022].

[4] "Constelación de Satélites Radar TerraSAR-X/PAZ • Hisdesat," Dec. 2022. [Online; accessed 16. Dec. 2022].

[5] "Rigo ribas, t. (2004). estudio de sistemas convectivos mesoscalares en la zona mediterránea occidental mediante el uso del radar meteorológico. http://cataleg.ub.edu/record=b1691394~s1 %2acat - buscar con google," 2004.

[6] M. Acosta Osorio, *Estudio de técnicas de procesamiento de señal en sistemas de radar para la detección del rango de un objeto/obstáculo para pruebas en ambientes de laboratorio.* PhD thesis, Universidad Pontificia Bolivariana, Mar 2014.

[7] D. Marchionni and F. Cavayas, "La teledetección por radar como fuente de información litológica y estructural: Análisis espacial de imágenes SAR de," *Geoacta*, vol. 39, no. 1, pp. 62–89, 2014.

[8] F. Sacristán Romero, "Sacristán romero, f. (2006a). la teledetección satelital y los sistemas de protección ambiental. aquatic, 5(24), 13–41. https://doi.org/10.22518/16578953.701 - buscar con google," 2006.

[9] F. S. Romero, "Teledetección satelital en la visión territorial y sistemas de protección ambiental urbano-rural (2ª parte)," *Urbano*, vol. 10, no. 15, pp. 84–90, 2007.

[10] J. A. F. Roselló and L. M. Rego, "La teledetección en el manejo de riesgos de desastres. Sus impactos en la sociedad," *stgo*, no. 118, pp. 76–88, 2009.

[11] M. I. Botana and S. E. Fernández, "Teledetección como experiencia de aprendizaje : Una mirada desde Geografía Física I, Geografía de los Espacios Marítimos y Cartografía," *Universidad Nacional de La Plata. Facultad de Humanidades y Ciencias de la Educación. Departamento de Geografía*, 2019.

[12] R. Acevo Herrera, *Sistemas de teledetección activos y pasivos embarcados en sistemas aéreos no tripulados para la monitorización de la tierra.* PhD thesis, Universitat Politècnica de Catalunya, Spain, Apr 2011.

[13] "Radar sensor introduction — acconeer-python-exploration documentation," Jan 2021. [Online; accessed 11. Jan. 2022].

[14] M. A. Ré, G. G. A. Varela, and S. Masuelli, "SEGMENTACIÓN DE SECUENCIAS CON DISTRIBUCIÓN GAMMA," *ANALES AFA*, vol. 30, pp. 36–41, Jul 2019.

[15] C. F. Castillo Plasencia, *Diseño de módulos de generación, conversión de frecuencia, amplificación y sincronización para un radar perfilador de vientos que opera a 445 MHz.* PhD thesis, Pontificia Universidad Católica del Perú, Oct 2016.

[16] L. Sandoval Serrano, "Sandoval serrano, l. (2018). algoritmos de aprendizaje automático para análisis y predicción de datos. revista tecnológica, 11, 36–40. - buscar con google," 2018.

[17] G. S. Cornejo Macias, "DiseÑo de un modelo de aprendizaje automÁtico (machine learning) para clasificar texto en variables pest.," 2021.

[18] "Russo, c., ramón, h., alonso, n., cicerchia, b., esnaola, l., & tessore, j. p. (2016). tratamiento masivo de datos utilizando técnicas de machine learning. xviii workshop de investigadores en ciencias de la computación, 131–134. - buscar con google," 2016.

[19] F. González, "Modelos de aprendizaje computacional en reumatología tt - machine learning models in rheumatology," 2015.

[20] Y. González Pérez and I. I. Kholod, "Uso de sistemas multiagentes para el aprendizaje automático," *Universidad de Los Andes*, 2020.

[21] A. C. Cardoso, M. L. Talamé, M. N. Amor, and A. Monge, "Creación de un Corpus de Opiniones con Emociones Usando Aprendizaje Automático," *RTyC.*, pp. 11–23, Apr 2020.

[22] D. Guerra Ramos, *Cofaseado de espejos segmentados con aprendizaje automático*. PhD thesis, Universidad de La Laguna, 2020.

[23] S. Cerdá Muñoz, *Predicción de efectos de drogas en cáncer mediante Machine Learning y Flux Balance Analysis*. PhD thesis, Feb 2021.

[24] M. J. Rodríguez González, *idUS - Depósito de Investigación Universidad de Sevilla*. PhD thesis, 2018.

[25] A. Albelda Martínez, "Composicion de cartera rÉplica del Índice bursÁtil alemÁn dax ulizando tÉcnicas pertenecientes al machine learning: Comparativa entre lasso, random forest u autoencoder," 2021.

[26] J. I. Jiménez León and J. J. Martínez Vera, *Análisis comparativo entre modelos de Machine Learning para la predicción de fallo en áreas axiales de un recipiente toroidal de sección recta circular para el almacenamiento de GNC*. PhD thesis, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas. Carrera de Ingeniería en Sistemas Computacionales., 2021.

[27] "SVM Hyperparameter Tuning using GridSearchCV | ML - GeeksforGeeks," Aug. 2022. [Online; accessed 25. Nov. 2022].

[28] "auto-sklearn — documentación de AutoSklearn - 0.15.0," Nov. 2022. [Online; accessed 25. Nov. 2022].

[29] S. López, "Deep Learning: Algoritmos de programación que aprenden por sí mismos," *Drauta*, Feb. 2020.

[30] Colaboradores de los proyectos Wikimedia, "Tcl - Wikipedia, la enciclopedia libre," Apr 2020. [Online; accessed 24. Jun. 2021].

[31] H.-S. Yeo, R. Minami, K. Rodriguez, G. Shaker, and A. Quigley, "Exploring Tangible Interactions with Radar Sensing," Dec. 2018.