ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Máster

MEJORAS BASADAS EN MACHINE LEARNING SOBRE UN SISTEMA BLE PARA LA ZONIFICACIÓN Y LOCALIZACIÓN EN INTERIORES

(Improvements based on Machine Learning on a BLE system for indoor zoning and location)

Para acceder al Título de

Máster Universitario en Ingeniería de Telecomunicación

Autor: Claudia González Oliva

Septiembre - 2020

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE MASTER

Realizado por: Claudia González Oliva
Director del TFM: Alberto Eloy García Gutiérrez
Título: "**Mejoras basadas en Machine Learning sobre un sistema BLE para la zonificación y localización en interiores**"
Title: "Improvements based on Machine Learning on a BLE system for indoor zoning and location"

Presentado a examen el día: 18 de Septiembre de 2020

para acceder al Título de

MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Pablo Pedro Sánchez Espeso
Secretario (Apellidos, Nombre): Héctor Posadas Cobo
Vocal (Apellidos, Nombre): Ramón Agüero Calvo

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFM
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Máster Nº
(a asignar por Secretaría)

Resumen

Este trabajo tiene como objetivo la mejora de los resultados obtenidos en trabajos anteriores de un sistema de posicionamiento para interiores basado en balizas Bluetooth [1] [2].

Para cumplir dicho objetivo, el desarrollo de este trabajo consiste en analizar y aplicar diferentes algoritmos y modelos de Machine Learning sobre los conjuntos de datos de las distintas medidas.

Este trabajo se ha centrado en las medidas de las balizas Bluetooth que fueron obtenidas en el segundo trabajo citado anteriormente mediante un sistema de módulos Bluetooth cuyas señales son recibidas y almacenadas por un terminal móvil para poder procesar los datos después y determinar correctamente su posición.

Estas medidas han sido utilizadas como datos de entrenamiento con los distintos algoritmos y se han evaluado varias rutas posibles en la zona de estudio.

Mediante el uso del lenguaje R se han implementado siete algoritmos distintos y debido a esta variedad de modelos estudiados se ha podido obtener una gran variedad de resultados y determinar el modelo que mejor funciona en este problema.

También, se han realizado los mismos pasos sobre otros conjuntos de datos, uno solo de medidas Wi-Fi y otro utilizando ambas medidas, Bluetooth y Wi-Fi.

La aplicación de estas técnicas de Machine Learning ha conseguido que, pese a que en algunos casos los resultados sean bajos, se mejoren los resultados obtenidos en los trabajos anteriores, por tanto, se ha cumplido el objetivo propuesto.

Abstract

This paper aims to improve the results obtained in previous works of an indoor positioning system based on Bluetooth beacons [1] [2].

To achieve this objective, the development of this work consists of analyzing and applying different algorithms and Machine Learning models on the datasets of the different measures.

This work is focused on the measurements of the Bluetooth beacons, which were obtained in the second project mentioned above using a system of Bluetooth modules whose signals are received and stored by a mobile terminal in order to be able to process the data later and determine their position correctly.

These measurements have been used as training datasets with the different algorithms and several possible routes have been tested in the study area.

Using the R language, seven different algorithms have been implemented and due to this diversity of models studied, it has been possible to obtain a great variety of results and to determine the model that works best in this problem.

At the same time, the same steps have been performed on other datasets, one of only Wi-Fi measurements and another one consisting of both measurements, Bluetooth, and Wi-Fi.

The application of these Machine Learning techniques has managed that, even though the results are low in some cases, the results obtained in previous works are improved, therefore, the proposed objective has been met.

Índice

Índice de figuras

Índice de tablas

1. Introducción

Este capítulo marca el punto de partida con una corta introducción sobre el problema a tratar en el desarrollo del trabajo, la motivación detrás de este estudio y la organización del documento.

1.1 El problema de la geolocalización en interiores

La geolocalización en escenarios abiertos es posible en la actualidad con diversas soluciones comerciales y gratuitas, y mediante varios sistemas de posicionamiento globales:

- **GPS:** Desarrollado e implementado por el Departamento de Defensa de Estados Unidos y en la actualidad propiedad de la Fuerza Espacial de los Estados Unidos.
- **GLONASS:** Desarrollado por la Unión Soviética y administrado hoy por el Ministerio de Defensa de la Federación Rusa.
- **Galileo:** Desarrollado por la Unión Europea, junto con la Agencia Espacial Europea, y actualmente operado por la Agencia Europea del Sistema Global de Navegación por Satélite.
- **BeiDou:** Desarrollado por China y operado por la Administración Espacial Nacional China.

Sin embargo, estos sistemas únicamente obtienen resultados fiables cuando existe visibilidad directa entre las antenas de referencia y del localizador. Por esto el posicionamiento en interiores sigue siendo objeto de estudio sin haber alcanzado todavía soluciones fiables e implementables en la realidad, tanto por dificultades tecnológicas, en cuanto a las señales electromagnéticas debido a interferencias y atenuaciones, como por el coste económico de crear una infraestructura fija de sensores, nodos y demás partes del sistema, además del coste de mantenimiento.

Una posible forma de implementar el posicionamiento en interiores es mediante el uso de los sensores integrados dentro de los smartphones, los dispositivos de uso más común, para obtener datos relacionados. Estos datos pueden ser analizados de acuerdo con la intención de calcular la posición dentro de un entorno cerrado, como es un edificio, para así poder localizar al usuario portador del teléfono.

A modo de ejemplo, el Virtual Positioning System de Google [3] utiliza los sensores del teléfono y su gran base de datos para analizar, mediante técnicas de Machine Learning, dónde se encuentra el usuario.

Otro ejemplo es el de su empresa rival son los iBeacons de Apple [4] que utilizan Bluetooth Low Energy, siendo su principal aplicación el que permiten localizar al usuario para que este reciba notificaciones según dónde se encuentre. Así, por ejemplo, en el caso de estar en una tienda puede recibir un mensaje de un producto en oferta en el pasillo por el que está pasando, o si el sistema es utilizado en un museo el visitante puede recibir información sobre las obras que estén expuestas.

1.2 Motivación y Objetivos

Este problema ha sido motivo de estudio en varios trabajos anteriores:

1. Trabajo de Fin de Máster de Alberto Gozalo Madrazo de 2017 [5]: "Estudio y desarrollo de una solución efectiva para el problema del posicionamiento en interiores": en este trabajo se estudia el uso de sistemas Bluetooth para la estimación de la posición de un móvil mediante técnicas simples de trilateración.

2. Trabajo de Fin de Máster de Javier Gómez Ortiz de 2018 [1]: "Estudio de la aplicación de Machine Learning a técnicas de posicionamiento en interiores": En este trabajo se estudian diferentes técnicas de inteligencia artificial y se busca mejorar los resultados del trabajo anterior, mediante una nueva visión del problema, incluyendo no solo tecnologías Bluetooth, sino también puntos de acceso WIFI.

3. Trabajo de Fin de Grado de Andrei Pietro Sciddurlo Juaristi de 2019 [2]: "Sistema BLE para la zonificación y localización en interiores": En este caso se parte del trabajo anterior, a partir del cual se modifican las zonas de decisión, así como las posiciones de las balizas Bluetooth de acuerdo con escenarios más reales.

En todos los casos se observaron ligeras mejoras en los resultados, aunque muy dependientes de los escenarios planteados, siendo éstos más discretos conforme más realistas eran las situaciones. La aplicación de técnicas de Inteligencia Artificial, en concreto, Machine Learning (ML), ha demostrado ser una opción prometedora, que solo ha sido planteada de forma muy simple. Una selección más cuidada de los métodos y algoritmos ML puede ser determinante para la mejora de los resultados obtenidos en los trabajos anteriores, ya que se han observado comportamientos muy diferenciados según los conjuntos de datos manejados, tanto sobre tecnologías diferenciadas (Bluetooth, y Wi-Fi), como de forma combinada

Con todo ello, el principal objetivo de este trabajo es el estudio comparativo del comportamiento de diferentes metodologías ML sobre los mismos conjuntos de datos obtenidos en los trabajos de referencia. De acuerdo con los resultados se propondrá una solución de compromiso para su aplicación en la resolución del posicionamiento de un móvil en escenarios sin cobertura de sistemas GPS.

1.3 Organización del documento

A continuación, se estructura brevemente el contenido de este trabajo, para, después de esta primera parte de introducción, exponer conceptos teóricos básicos, primero los sistemas de posicionamiento en el capítulo 2, y seguidamente, sobre Machine Learning en el capítulo 3. Para finalizar con la teoría, el capítulo 4 expone el desarrollo teórico de los algoritmos utilizados de forma específica en este trabajo.

El capítulo 5, dedicado al desarrollo práctico del Trabajo, dedica un primer apartado al lenguaje de programación utilizado, denominado R, al paquete utilizado para implementar los métodos de Machine Learning, denominado Caret, y a las funciones concretas de los algoritmos utilizados. Para finalizar el capítulo, se describen los datos de partida del estudio y se detallan los pasos seguidos.

En el capítulo 6, se exponen los principales resultados obtenidos para los tres tipos de datos: solo Bluetooth, solo Wi-Fi y ambos combinados, incluyendo las diferentes comparativas realizadas.

Finalmente, se dedica el capítulo 7 a las conclusiones y posibles líneas futuras de trabajo.

2. Sistemas de posicionamiento

Los sistemas de posicionamiento permiten localizar un objeto en un espacio y tiempo determinados basándose en la propagación de señales electromagnéticas. Al conocer la potencia de transmisión del emisor y su posición, se puede estimar la posición del receptor al calcular la distancia entre ambos. En la práctica los sistemas se deben adaptar a la variabilidad del medio, los obstáculos y todos los efectos sobre la señal que dificultan la transmisión.

2.1 Posicionamiento en exteriores

Los sistemas de posicionamiento globales por satélite comenzaron durante la carrera espacial y fueron motivados, como muchos avances tecnológicos, por motivaciones bélicas, en este caso, la Guerra Fría. En la introducción se han nombrado cuatro sistemas de posicionamiento global, los dos primeros, GPS [6] y GLONASS fueron creados por los Estados Unidos y la Unión Soviética, respectivamente, mientras que el sistema Galileo y BeiDou son más recientes y creados por la Unión Europea y China. Al ser el GPS el más extendido, es este sistema el que se describe a continuación.

El Sistema de Posicionamiento Global permite determinar la posición de cualquier objeto en el planeta con una precisión de pocos metros mediante el uso de la trilateración utilizando un mínimo de cuatro satélites.

La infraestructura consiste en una red de un mínimo de 24 satélites orbitando la Tierra a 20200 kilómetros de altura, lo que los sitúa en la órbita media terrestre, MEO, con un periodo de 12 horas, distribuidos para que desde cualquier punto de la tierra haya cuatro satélites visibles. Cada uno de estos satélites emite de forma continua una señal a 50 bits por segundo a 1575,42 MHz y 1227,6 MHz. Cada transmisión lleva 1500 bits de datos codificados con una secuencia pseudoaleatoria distinta para cada uno, de esta manera los receptores GPS pueden decodificar la señal y distinguir entre los distintos satélites.

Para determinar la localización de un aparato, éste debe localizar estos cuatro satélites y recibir las señales que informan sobre la hora, identificación y constelación de cada uno. Así, el receptor sincroniza su reloj con el sistema GPS y es capaz de calcular la latencia en la transmisión de la señal entre el satélite y el equipo, para medir la distancia entre ambos. Por último, mediante la trilateración inversa, obtiene su propia posición.

La trilateración es un método matemático para obtener la posición relativa de un objeto mediante la geometría de triángulos utilizando las localizaciones de varios puntos de referencia y la distancia entre el objeto a localizar y cada punto de referencia [7].

- Cada satélite indica la situación del receptor en un punto en la esfera, con centro el satélite y radio la distancia del receptor.
- Con la información de dos satélites más, se determina una zona en la que se intersecan las tres esferas en el punto donde está el receptor.
- Para proporcionar también la altitud es necesario contar con un cuarto satélite.

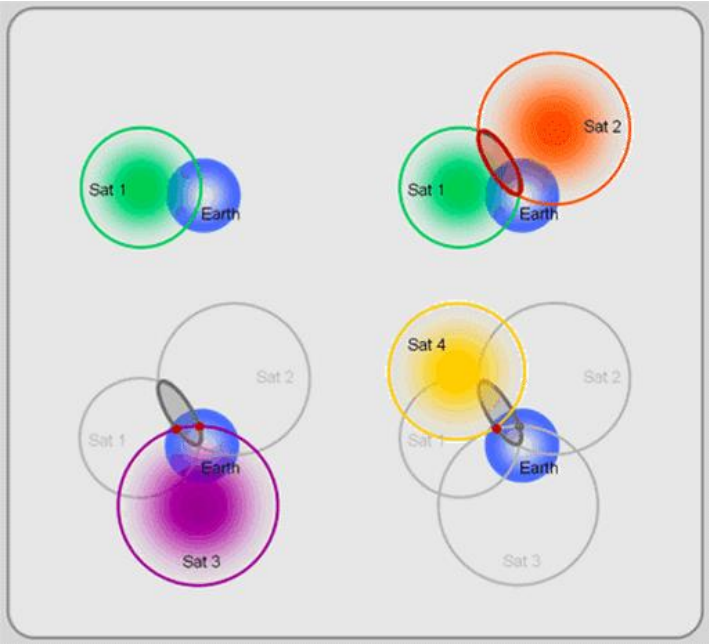

Figura 1. Trilateración satelital. [7]

Es posible utilizar la trilateración también con las estaciones base de redes móviles, ya que al enviar y recibir información entre el móvil y la estación base, se obtiene información a partir de la cual se puede calcular su posición relativa de forma similar al método GPS, con las estaciones base tomando el rol de la red satelital. Aunque este método tiene un peor resultado que el GPS, ya que entre el terminal móvil y las estaciones base hay más obstáculos, que perjudican la señal.

Aunque estos dos sistemas funcionan en exteriores, no pueden utilizarse en interiores, ya que la señal se atenúa demasiado por las paredes y el mobiliario que hay en el interior de los edificios. Además, la distribución del interior hace que haya múltiples reflexiones y refracciones en el trayecto de las señales. Cabe destacar que el comportamiento en general del sistema se ve afectado también porque estos efectos en las señales no afectan a todas por igual.

2.2 Posicionamiento en interiores

Dado que no se pueden utilizar las técnicas de posicionamiento en exteriores para los escenarios en interiores, una opción puede ser desarrollar infraestructuras en los edificios que funcionen de forma similar, ya que el sistema de posicionamiento global tiene una precisión tan buena sobre la localización.

Para copiar esta infraestructura, se debería situar un determinado número de balizas en una posición concreta dentro del edificio que emitan información de su posición e información temporal. El problema en espacios pequeños es que la latencia apreciada por el sistema al captar las señales es muy baja, ya que las señales llegan casi de forma instantánea. Otro problema son las reflexiones de la señal debido a la distribución del interior, que puede causar errores en el cálculo de la posición.

Por estos motivos se debe descartar la adaptación de los sistemas GPS, aunque las técnicas de trilateración pueden servir si en lugar de tener en cuenta la latencia en la transmisión se tiene en cuenta la atenuación.

En el caso del posicionamiento en interiores, el cálculo de la posición se basa en la atenuación de la señal, que, aunque puede variar por interferencias o reflexiones, es un método sencillo de implementar y además en el caso de señales Wi-Fi y Bluetooth no necesita la creación de una nueva infraestructura, ya que los controladores de los dispositivos proporcionan la fuerza de la señal recibida.

Últimamente se han buscado nuevas formas de posicionamiento en interiores basándose en nuevas tecnologías, como por ejemplo la realidad virtual, VR, y la realidad aumentada, AR, con sistemas que pueden determinar la posición y el movimiento del usuario para incluirlos en los videojuegos.

Uno de los dispositivos de realidad virtual más populares es Oculus Rift, desarrollado por Oculus VR que depende de Facebook, mediante el uso de cámaras infrarrojas para detectar leds con los que calcular la posición y la orientación del jugador.

Otro dispositivo de la competencia es HTC Vive [8], desarrollado por HTC y Valve, basado en telemetría láser. Este sistema de realidad virtual compuesto por las gafas y los controles tiene acelerómetro y giroscopio en estos tres componentes y sensores ópticos para detectar la luz infrarroja que emiten las cajas que se colocan en puntos opuestos de la sala donde se juega para cubrir toda zona. Éstas emiten 60 pulsos por segundo y después de ese pulso un láser barre la habitación. Para obtener la posición del usuario, las gafas y los controles comienzan un contador cuando reciben la luz infrarroja y éste se detiene al recibir el láser.

Aunque estos métodos de realidad virtual son muy precisos y relativamente asequibles, no son prácticos a gran escala, ya que se necesita una gran infraestructura en los edificios y se requiere que los usuarios lleven un equipo receptor. Además, cualquier obstáculo inesperado baja su rendimiento.

Hay que destacar que, hoy en día, una gran mayoría de personas ya contamos con un dispositivo inteligente asequible con múltiples sensores y una gran capacidad de obtener información, el smartphone, y que se prevé que en el futuro este número de usuarios siga creciendo.

Una solución de posicionamiento en interiores relacionada con los dispositivos móviles es el mapeado de redes Wi-Fi, ya que es habitual tener un router Wi-Fi en los interiores de los edificios, para generar un mapa de las señales que pueda utilizarse para triangular la posición en un smartphone. Para obtener dicho mapa se recorre el edificio con un sistema que analice las redes en cada posición y su intensidad. Es una técnica relativamente sencilla, pero depende mucho de la estructura del edificio, la red de routers y la posición de estos, ya que esto haría variar el mapa de las señales.

Se debe concretar que la denominación Wi-Fi [9] hace referencia al conjunto de protocolos basados en las especificaciones 802.11 de IEEE [10], que tratan la conexión de equipos mediante radiofrecuencias en la banda de 2,4 GHz o 5 GHz en una WLAN, Wireless Local Area Network.

Es una técnica que ya está en uso, en Google Maps se pueden recibir notificaciones de los establecimientos dentro de un centro comercial. Aunque la forma en que Google calcula el posicionamiento de sus usuarios no es pública, el método debe utilizar las señales Wi-Fi que detecta el dispositivo móvil, y no requiere ninguna infraestructura extra en dichos establecimientos ni por parte del usuario.

Otra forma similar al mapeado de Wi-Fi es el uso de beacons de Bluetooth, normalmente Bluetooth Low Energy que sirve para reducir el consumo en las conexiones, que emiten una señal Bluetooth, más débil que Wi-Fi, lo que hace que sean más localizadas en una zona y que para recibirlas se deba estar más cerca. Este sistema utiliza la banda ISM de 2,4 GHz en una WPAN, Wireless Personal Area Network.

Este trabajo se basa en una infraestructura de beacons Bluetooth y en un mapeado de redes Wi-Fi, que han sido utilizados para recopilar los datos de las redes. En el trabajo se aplican múltiples técnicas de Machine Learning sobre estos datos y se analiza su rendimiento.

3. Inteligencia Artificial

El término Inteligencia Artificial, junto con Machine Learning, es en la actualidad un foco de interés por parte de estudios e investigaciones debido a la mejora tecnológica en la capacidad de los sistemas para poder ser entrenados con el objetivo de alcanzar una simulación de la mente humana más precisa. Esta mejora ha tenido lugar en tres partes clave: la potencia de cálculo de los dispositivos, la disponibilidad de grandes cantidades de datos y la comprensión teórica y práctica de las técnicas de inteligencia artificial.

La inteligencia artificial consiste en el comportamiento inteligente de las máquinas, su capacidad de percibir datos del ambiente y actuar según ellos. Lo que esto busca es simular la capacidad humana de aprender, razonar y resolver problemas. Una aplicación básica es que un dispositivo pueda escanear el lugar con obstáculos donde se encuentre e interpretarlo, para saber por dónde puede moverse.

Para que los aparatos puedan actuar así se necesita una gran cantidad de información para comprender cómo las personas reaccionan al entorno. Los dispositivos necesitan saber qué relaciones hay entre objetos, la conexión entre causa y efecto y demás componentes que para el pensamiento humano son naturales y adquiridas durante años.

En el proceso de modelado los datos del entorno son procesados y analizados por programas que devuelven una solución a un problema. Este modelo debe ser calibrado de forma iterativa para lograr que la solución que se obtiene se corresponda con una solución humana.

El rango de acción posible en inteligencia artificial se puede dividir en dos grandes grupos:

- **Soft Artificial Intelligence [11]:** Se trata de algoritmos basados en encontrar una solución correcta a partir de unos datos de entrada y únicamente capaces de realizar esta función. Únicamente resuelven una tarea, o un conjunto de tareas, pero su inteligencia está limitada. La inteligencia del sistema está en encontrar, durante el periodo de entrenamiento, la función adecuada, que, a partir de unos determinados datos de entrada, devuelve una salida correcta. Es el tipo de Inteligencia Artificial que se utiliza actualmente para temas como las sugerencias de anuncios basados en compras o visitas de páginas anteriores, el reconocimiento facial, chatbots de atención al cliente, detectar amenazas de seguridad o fraude, etc.
- **Hard Artificial Intelligence:** Se trata de un sistema cuya inteligencia no se limita al periodo de entrenamiento, ni a buscar una solución a un único problema, sino que tiene la capacidad de pensar igual que un humano, incluso llegando a tomar conciencia de sí misma. Este tipo actualmente no existe y se considera que en el futuro cercano continúe siendo inviable.

En este trabajo, al igual que en todas las aplicaciones actuales de esta tecnología, el uso del término Inteligencia Artificial se refiere únicamente a Soft Artificial Intelligence.

Machine Learning y Deep Learning son ambos subconjuntos con los que se puede crear un sistema de inteligencia artificial.

3.1 Deep Learning

Deep Learning, también denominado aprendizaje profundo, utiliza redes neuronales artificiales para procesar los datos y llegar a una solución. Es un subconjunto de las técnicas de Machine Learning que utiliza múltiples capas con unidades de procesamiento no lineal para extraer variables y transformarlas, con cada capa utilizando las salidas de la capa anterior como entrada, con aprendizaje autónomo en cada capa.

Las "neuronas de entrada" reciben los datos entrantes, las "neuronas ocultas" o "neuronas intermedias" realizan cálculos intermedios en la red y las "neuronas de salida" contienen el resultado. Lo habitual es que todas las neuronas en cada capa estén conectadas con la siguiente capa y dichas conexiones tienen unos pesos determinados.

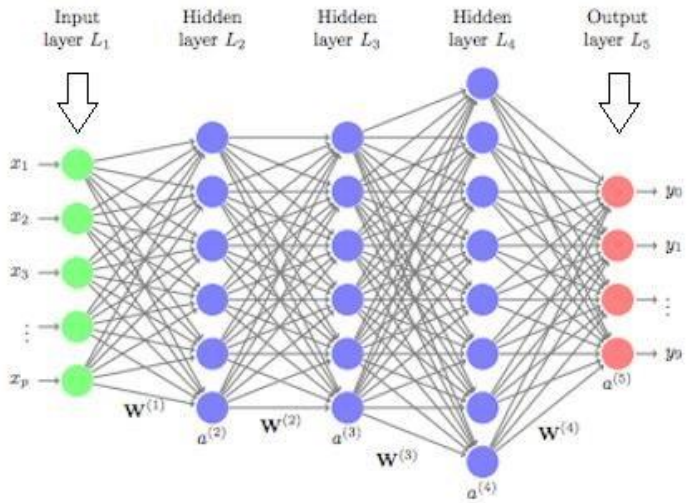

Figura 2. Diagrama de Deep Learning. [12]

El funcionamiento del sistema consiste en productos y sumas. Además, se tiene la función de activación, o sigmoide, cuya función es hacer que todos los valores de las neuronas del sistema se encuentren entre 0 y 1 para evitar realizar grandes operaciones.

Para caracterizar una red hay que decidir el número de capas ocultas, cuántas neuronas se deben poner en cada capa oculta y los pesos entre capas. Los pesos se pueden obtener mediante el entrenamiento. El objetivo es conseguir un sistema lo suficientemente complejo para resolver los problemas de forma satisfactoria y a la vez lo más sencillo posible para que cueste menos hacer las operaciones.

Se puede saber el error de cada neurona de la capa de salida comparando la respuesta obtenida con la esperada y, debido a que ese valor está formado por operaciones con las neuronas anteriores, se puede calcular cuánto contribuye al error cada neurona de la red y variar los pesos para hacer que afecte menos.

Hay otras técnicas de Deep Learning basadas en redes de neuronas recurrentes, que no tienen capas sino solo conexiones entre las neuronas, o bien, las redes de neuronas convolutivas, que mantienen las capas, pero cada neurona no debe estar conectada con todas las de la capa anterior.

Deep Learning se puede utilizar en la actualidad para resolver múltiples problemas como el análisis de imágenes, el reconocimiento de voz o la identificación de emociones faciales.

3.2 Machine Learning

El Machine Learning [13], también denominado aprendizaje autónomo, busca encontrar soluciones a partir del análisis de datos. Primero se generan predicciones mediante el análisis de los datos y seguidamente éstas se utilizan para tomar una decisión. Los resultados se pueden usar para mejorar las futuras predicciones. También se puede usar para optimizar funciones y para extraer estructuras de los conjuntos de datos al clasificarlos.

El campo de Machine Learning se puede clasificar en las siguientes categorías:

- **Aprendizaje supervisado:** Se trata del método en el que, a la hora de entrenar el algoritmo, dentro del conjunto de datos se encuentra también la salida. Si los datos de entrada van junto a la salida correcta, se denominan datos etiquetados. Estos datos de salida pueden ser un valor numérico o una etiqueta de clase. Su objetivo es generar una función que permita predecir la salida correcta después de analizar los datos de entrenamiento.
- **Aprendizaje no supervisado:** A diferencia del anterior, en éste se desconoce el valor de salida correcto según los datos de entrada, ya que no están etiquetados. Este tipo busca encontrar estructuras en los datos proporcionados para poder caracterizarlos. Es una forma curiosa de aprendizaje autónomo porque, aunque se puedan encontrar estructuras que no aporten ninguna información valiosa y resulte en falsos positivos, a veces, este método puede detectar relaciones que, a simple vista, y mediante técnicas de análisis tradicionales, no pueden encontrarse.
  Debido a que no requiere del etiquetado de datos, es una forma de aprendizaje fácil de aplicar.
- **Aprendizaje por refuerzo:** Pese a que las principales categorías de ML son las dos anteriores, hay que destacar este último. Se trata de un método parecido al aprendizaje supervisado, con la diferencia de que, en lugar del valor de salida correcto, lo que se tiene es el error cometido. Su funcionamiento consiste en proporcionar los datos de entrada en el periodo de aprendizaje, dar una respuesta y recibir una crítica sobre el acierto de esa respuesta.
  Resulta un tipo de aprendizaje muy útil para los casos en los que la respuesta correcta no es conocida, problemas de complejidad NP y NP-Hard, ya que sirve para encontrar aproximaciones a la solución en un tiempo finito.

Estos tres tipos de aprendizaje sirven para resolver los problemas de inteligencia artificial, que debido a las múltiples variantes que tienen, pueden dividirse en nueve grupos [14]:

- **Clasificación:** El problema consiste en determinar el grupo, denominado clase, al que pertenece un conjunto de datos a partir de los datos de entrenamiento. Por ejemplo, determinar si un correo electrónico es spam según el texto del correo, diagnosticar una enfermedad a un paciente a partir de sus características, etc.

- **Regresión:** El problema es similar a uno de clasificación, sin embargo, la salida es un número, en lugar de una clase. Por ejemplo, la predicción del precio de una vivienda según múltiples variables, la predicción del estado de las acciones en bolsa, etc.
- **Identificación de similitudes:** Sirve para los problemas de sistemas de recomendación, para poder agrupar a los usuarios según sus similitudes de búsqueda y productos adquiridos.
- **Clustering:** Es útil para agrupar individuos, igual que el grupo anterior, pero sin un propósito determinado.
- **Agrupar coocurrencias:** El problema consiste en asociar atributos que sucedan al mismo tiempo. Por ejemplo, averiguar qué productos suelen comprarse juntos para poder hacer promociones de ambos.
- **Profiling:** El problema se trata de determinar el comportamiento de un usuario o grupo de usuarios determinado. Por ejemplo, comprobar el consumo de datos móviles de un segmento de clientes. Estos comportamientos típicos se pueden utilizar como referencia si sucede una anomalía para bloquear transacciones sospechosas. El caso de que si una persona suele pagar una serie de gastos con su tarjeta bancaria y en un corto periodo de tiempo se detectan grandes compras se puede lanzar una alerta de seguridad.
- **Predicción de vínculos:** Sirve para predecir conexiones entre elementos como, por ejemplo, sugerencias de contactos en una red social.
- **Reducción de datos:** El problema es tener una gran cantidad de datos que hace falta reducir para poder trabajar con ellos, garantizando una buena relación entre la pérdida de información y la mejora de dicha reducción. Un problema que puede suceder en bases de datos de preferencia de consumo.
- **Modelado Causal:** El problema trata de detectar cuánto influye un hecho sobre otro. Es importante conocer, por ejemplo, si una campaña de marketing hace que aumenten las ventas, para continuar con dicha campaña o no.

Cuando se busca resolver un problema, dependiendo de la forma en que se oriente, se buscará un algoritmo u otro, el que resulte más apropiado. Una forma de determinar qué algoritmo puede servir para el problema en cuestión es consultar los siguientes cinco puntos:

- Determinar si un dato es de la clase A o de la clase B. Clasificación.
- Identificar datos atípicos o inesperados. Detección de anomalías.
- Determinar un valor numérico según datos conocidos. Regresión.
- Determinar la estructura de los datos. Clustering.
- Determinar el comportamiento en base a datos. Aprendizaje por refuerzo.

Para la realización de este trabajo se parte de una serie de medidas cuyas características son el identificador de la baliza Bluetooth emisora de la señal, la potencia recibida y la clase, o zona, a la que pertenece la medida.

En el caso de los datos Wi-Fi, se cuenta con el identificador del punto de acceso Wi-Fi, la potencia recibida y la clase, o zona, a la que pertenece la medida.

En el capítulo 5, donde se especifican los aspectos prácticos del trabajo, se detallará la estructura y características de estos datos.

En ambos casos son datos etiquetados, por lo que se ha determinado utilizar técnicas de aprendizaje supervisado y clasificación. El problema consiste en determinar en qué zona está situado el usuario a partir de los datos de potencia recibida de cada identificador. Como hay múltiples zonas, el problema es de clasificación múltiple.

Después de identificar el tipo de problema, se pueden estudiar diversos algoritmos de clasificación múltiple que puedan ser implementados para poder analizar su rendimiento.

4. Modelos de Machine Learning

En este capítulo se tratan los siete modelos de aprendizaje autónomo que se han estudiado para el desarrollo de este trabajo:

- LDA Linear Discriminant Analysis
- CART Classification and Regression Tree
    - Recursive Partitioning
    - Bagged CART
    - Random Forest
- KNN k-Nearest Neighbours
- SVM Support Vector Machine
- Multinomial Logistic Regression

A continuación, se dan algunos detalles importantes sobre cada uno de estos modelos.

4.1 LDA Linear Discriminant Analysis

El Análisis Discriminante Lineal es una técnica de clasificación de datos mediante aprendizaje supervisado cuya finalidad es reducir la dimensionalidad de los datos. El método sirve para encontrar una combinación lineal a partir de la cual poder diferenciar entre distintas clases. La idea principal de este algoritmo es maximizar la separación entre las clases para poder clasificar los datos correctamente [15].

La técnica fue desarrollada inicialmente en 1936 por Ronald A. Fisher [16] con el nombre de Discriminante Lineal y aplicable para casos de 2 clases. Aunque la versión multiclase fue generalizada por C. R. Rao en 1948 [17] como Análisis Discriminante Múltiple se suelen denominar ambas Análisis Discriminante Lineal.

Este tipo de análisis está relacionado con el Análisis Factorial y con el Análisis de Componente Principal (PCA), y se diferencia de este último en que LDA realiza clasificación de datos de forma supervisada y el PCA funciona de forma no supervisada, por lo que ignora las etiquetas de clase y pasa a analizar una combinación de atributos de los datos de entrada que se denominan Componentes Principales.

Otras técnicas relacionadas son el Análisis de la Varianza (ANOVA) y el Análisis de Regresión, ya que estas también utilizan combinaciones lineales, pero se diferencian en el tipo de variables utilizadas, debido a que LDA utiliza variables independientes continuas y una variable dependiente de etiqueta de clase, mientras que ANOVA hace lo contrario, tiene variables independientes categóricas y una variable dependiente continua.

La técnica LDA funciona cuando los valores de las variables independientes de cada observación son números continuos, si se trata de variables independientes categóricas, se utiliza el Análisis Discriminante de Correspondencia (DCA).

Los sets de datos y los vectores de test se pueden transformar de dos formas.

- Transformación dependiente de clase: Para maximizar el ratio entre la varianza entre clases y la varianza de dentro de la clase.

- Transformación independiente de clase: Para maximizar el ratio entre la varianza global y la varianza de dentro de la clase.

Para el caso original de la clasificación de dos clases, el conjunto de entrenamiento consiste en una serie de observaciones, $\vec{x}$, para cada medida de una clase, y. Se debe encontrar un buen estimador para la clase dada una observación.

Se asume que la probabilidad de cada clase sigue una distribución normal $p(\vec{x}\,|y=0)$ y $p(\vec{x}\,|y=1)$ con parámetros de media y covarianza $(\overrightarrow{\mu_0}|\Sigma_0)$ y $(\overrightarrow{\mu_1}|\Sigma_1)$. También se asume que las covarianzas de las clases con iguales, homocedasticidad, y tienen rango completo. Además, el poder de predicción puede disminuir si las variables tienen una alta correlación y se asume que los datos han sido obtenidos de forma aleatoria.

Se pueden determinar los pasos que siguen las operaciones matemáticas de este algoritmo:

- Determinar los conjuntos de datos y test.
- Calcular la media de cada conjunto y la de la totalidad de los datos. Para ello se calcula la media de ambos conjuntos por separado y se multiplica cada una por la probabilidad de las clases a priori, en este caso se asume que es 0.5, por lo que la media de todos los datos es:

$$\mu_2 = \mu_0 * p_0 + \mu_1 * p_1$$

- Las medidas de dispersión son relevantes para la separabilidad de las clases. La dispersión de dentro de la clase es la covarianza esperada de cada clase.

$$S_w = \sum_j p_j * cov_j = p_0 * cov_0 + p_1 * cov_1$$

$$cov_j = (x_j - \mu_j) * (x_j - \mu_j)^T$$

- La dispersión entre clases se calcula con la siguiente ecuación.

$$S_b = \sum_j (\mu_j - \mu_2) * (\mu_j - \mu_2)^T$$

- Como se ha comentado anteriormente existen 2 enfoques de LDA, para el tipo dependiente de clase, los factores de optimización de cada clase son:

$$criterion_j = inv(cov_j) * S_b$$

- Para el tipo independiente de clase, el factor de optimización es:

$$criterion = inv(S_w) * S_b$$

- Un eigenvector de una transformación representa un subespacio de 1 dimensión donde se aplica la transformación. Cualquier espacio vectorial se puede representar como una combinación lineal de los eigenvectores. En LDA, las transformaciones son la matriz de eigenvectores de los factores de optimización.
- La región de decisión es una línea que separa los datos transformados.

- Para el tipo dependiente de clase, la transformación de los datos es:

$$transformed\_set\_j = transform\_j^T * set\_j$$

- Para el tipo independiente de clase, el factor de optimización es:

$$transformed\_set = transform\_spec^T * data\_set^T$$

- Después de completar las transformaciones se utiliza la distancia Euclídea para clasificar los puntos. En esta fórmula, n es el número de la clase, x es el vector de test y μ_{ntrans} es la media del conjunto de datos transformados.

$$dist\_n = transform\_n\_spec^T * x - \mu_{ntrans}$$

- La menor distancia que se tenga entre las n clases clasificará al vector en la clase n.

Otra forma de abarcar el algoritmo es verlo partiendo de la solución óptima de Bayes, prediciendo que un punto pertenece a la segunda clase si no se supera un valor umbral, T, que se considera el Análisis Discriminante Cuadrático, ADC.

$$(\vec{x} - \vec{\mu}_0)^T \Sigma_0^{-1} (\vec{x} - \vec{\mu}_0) + \ln|\Sigma_0| - (\vec{x} - \vec{\mu}_1)^T \Sigma_1^{-1} (\vec{x} - \vec{\mu}_1) - \ln|\Sigma_1| > T$$

Al asumir homocedasticidad y covarianzas de rango completo, se cancelan varios términos y se convierte en:

$$\vec{w} \cdot \vec{x} > c$$

$$\vec{w} = \Sigma^{-1}(\vec{\mu}_1 - \vec{\mu}_0)$$

$$c = \frac{1}{2}(T - \vec{\mu}_0{}^T \Sigma_0^{-1} \vec{\mu}_0 + \vec{\mu}_1{}^T \Sigma_1^{-1} \vec{\mu}_1)$$

La observación formará parte de una clase si la observación está situada en el hiperplano perpendicular a $\vec{w}$ cuya localización la define el umbral c.

Se debe destacar que el documento original de Fisher es algo distinto al ADL, ya que no supone la distribución normal de las clases o la homocedasticidad.

Se determina que la combinación lineal de $\vec{w} \cdot \vec{x}$ tiene medias $\vec{w} \cdot \vec{\mu}_i$ y varianzas $\vec{w}^T \cdot \Sigma_i \cdot \vec{w}$. La separación entre las 2 clases se puede realizar por la proporción de la varianza entre las clases y dentro de las clases.

$$S = \frac{\sigma_{entre}^2}{\sigma_{dentro}^2} = \frac{(\vec{w} \cdot \vec{\mu}_1 - \vec{w} \cdot \vec{\mu}_0)^2}{\vec{w}^T \cdot \Sigma_1 \cdot \vec{w} + \vec{w}^T \cdot \Sigma_0 \cdot \vec{w}} = \frac{(\vec{w} \cdot (\vec{\mu}_1 - \vec{\mu}_0))^2}{\vec{w}^T \cdot (\Sigma_1 + \Sigma_0) \cdot \vec{w}}$$

Si las distribuciones son similares, el hiperplano entre las proyecciones de las dos medias se puede elegir umbral.

$$c = \frac{1}{2}\vec{\mu}_1{}^T \Sigma_1^{-1} \vec{\mu}_1 - \frac{1}{2}\vec{\mu}_0{}^T \Sigma_0^{-1} \vec{\mu}_0$$

Si existen múltiples clases, se extiende este Discriminante de Fisher, con la generalización de C. R. Rao suponiendo que cada clase tiene una media distinta y la misma covarianza.

La dispersión entre las clases se obtiene con la siguiente ecuación:

$$\Sigma_b = \frac{1}{C}\sum_{i=j}^{C}(\mu_j - \mu) * (\mu_j - \mu)^T$$

Y la separación entre clases es:

$$S = \frac{\vec{w}^T \cdot \Sigma_b \cdot \vec{w}}{\vec{w}^T \cdot \Sigma \cdot \vec{w}}$$

Si $\vec{w}$ es un vector propio de $\Sigma^{-1}\Sigma_b$, la separación es igual a su valor propio.

Hay más técnicas para realizar esta clasificación con múltiples clases, como por ejemplo dividir las clases o agrupar los puntos de una clase y compararlos con todos los demás correspondientes a las otras clases para aplicar ADL.

A la hora de poner esto en práctica, las medias y las covarianzas de las clases no son conocidas, pero se pueden estimar mediante el conjunto de entrenamiento. También puede suceder que las medidas de cada ejemplo superen al número de ejemplos que tiene cada clase, por lo que no se pueda invertir la matriz de covarianza y se tenga que utilizar la pseudoinversa.

Este método de clasificación tiene diversas aplicaciones prácticas:

- Predecir la entrada en bancarrota: Tomando diversas variables financieras y de contabilidad se puede aplicar el LDA para analizar qué empresas son las que entran en bancarrota y las que no.
- Reducir el número de píxeles a analizar en aplicaciones como el reconocimiento de caras: Su funcionamiento es generar una plantilla con las combinaciones lineales de los píxeles tras la reducción de la imagen.
- Analizar la relación entre clientes y los productos de una empresa tomando diversos datos de entrada, como encuestas.
- Determinar la severidad de las enfermedades en estudios clínicos.

4.2 CART Classification and Regression Tree

El aprendizaje basado en árboles de decisión se basa en un árbol de decisión binaria creado mediante la división de un nodo en 2 [18]. Tiene una forma similar a un diagrama de flujo. La función principal de un árbol de decisión es predecir una variable según los valores de entrada.

Si los valores del resultado son discretos se les llama árboles de clasificación y las clases están representadas por las hojas del árbol. Si la variable del resultado es continua, se denomina árbol de regresión.

Se utiliza esta denominación de árbol de decisión para ambos árboles que tienen algoritmos similares con diferencias en su funcionamiento y aplicación, como por ejemplo la división de los nodos.

Existen múltiples métodos de Machine Learning basados en árboles de decisión que han sido utilizados en este trabajo, como Recursive Partitioning, Bagged CART y Random Forest.

La idea principal para generar el árbol es elegir la mejor división del nodo para que las hojas del árbol sean las mejores.

- Si el set de todas las variables es una variable nominal categórica de I categorías, entonces hay $2^{I-1} - 1$ posibles divisiones.
- Si el set de todas las variables es una variable continua u ordinal categórica con K valores distintos, entonces hay $K - 1$ posibles divisiones.

El árbol de decisión se crea desde el nodo raíz siguiendo los pasos en cada nodo:

- Encontrar la mejor división de la predicción: Para el caso continuo y ordinal, se ordenan los valores de menor a mayor y se examina cada punto de división. El mejor punto es el que maximiza el criterio de división. En el caso nominal, se examinan las categorías.
- Encontrar la mejor división del nodo: Aquel que maximiza el criterio de división.
- Dividir el nodo con su mejor división si las reglas lo permiten.

En el nodo t, la mejor división s, se elige para maximizar el criterio de división $\Delta i(s,t)$. Si se puede definir la impureza de un nodo, el criterio de división se corresponde con una disminución de la impureza.

Si el resultado es categórico hay tres criterios de división: Gini, Twoing y Twoing ordenado.

En el nodo t, las probabilidades son:

$$\pi(j) = probabilidad\ a\ priori\ de\ (Y = j)$$

$$w_n = peso\ del\ caso\ n$$

$$f_n = frecuencia\ del\ caso\ n$$

$$N_{w,j} = \sum_{n\ \epsilon h} w_n\ f_n I(y_n = j) \quad I = 1\ si\ y_n = j, \quad else\ I = 0$$

$$N_{w,j}(t) = \sum_{n\ \epsilon\ h(t)} w_n\ f_n I(y_n = j) \quad I = 1\ si\ y_n = j, \quad else\ I = 0$$

$$p(j,t) = probabilidad\ de\ un\ caso\ en\ clase\ j\ y\ nodo\ t = \frac{\pi(j) N_{w,j}(t)}{N_{w,j}}$$

$$p(t) = probabilidad\ de\ un\ caso\ en\ nodo\ t = \sum_j p(j,t)$$

$$p(j|t) = probabilidad\ de\ caso\ en\ clase\ j\ si\ está\ en\ nodo\ t = \frac{p(j,t)}{p(t)} = \frac{p(j,t)}{\sum_j p(j,t)}$$

- **Criterio de Gini generalizado**

La impureza de Gini se mide como:

$$C(i|j) = coste\ de\ error\ al\ clasificar\ la\ clase$$

$$i(t) = \sum_{i,j} C(i|j)p(i|t)p(j|t)$$

El criterio de división se define como

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

$$p_L = p(t_L)/p(t) \qquad p_R = p(t_R)/p(t)$$

La probabilidad de pasar un caso al nodo izquierdo es p_L, y al derecho p_R.

- **Criterio de Twoing**

$$\Delta i(s,t) = p_L p_R \left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2$$

- **Criterio de Twoing ordenado**

Se utiliza solo si es ordinal categórico.

Se separa la clase $C = \{1,\ldots,J\}$ de Y en dos super clases $C_1, C_2 = C - C_1$ con $C_1 = \{1,\ldots,j_1\}\ j_1 = 1,\ldots,J-1$.

Utilizando $i(t) = p(C_1|t)p(C_2|t)$ obtener:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R) = p_L p_R \left[\sum_j |p(j|t_L) - p(j|t_R)|\right]^2$$

Buscar la clase C_1 que maximice $\Delta i(s(C_1),t)$

Si el resultado es continuo el criterio de división se utiliza con las medidas de Desviación de Mínimos Cuadrados, LSD.

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

$$i(t) = \frac{\sum_{m\epsilon h(t)} w_n f_n (y_n - \bar{y}(t))^2}{\sum_{m\epsilon h(t)} w_n f_n}$$

$$p_L = \frac{N_w(t_L)}{N_w(t)} \qquad p_L = \frac{N_w(t_L)}{N_w(t)} \qquad N_w(t) = \sum_{m\epsilon h(t)} w_n f_n \qquad \bar{y}(t) = \frac{\sum_{m\epsilon h(t)} w_n f_n y_n}{N_w(t)}$$

Existen reglas para determinar si el proceso de crecimiento del árbol debe parar o no.

- Si todos los casos del nodo tienen los mismos valores que la variable dependiente, el nodo no se divide.

- Si todos los casos de un nodo tienen los mismos valores para cada predictor, el nodo no se divide.
- Si la profundidad del árbol alcanza la máxima determinada por el usuario, el proceso de aumentar el árbol para.
- Si el tamaño de un nodo es menor que el determinado por el usuario, el nodo no se divide.
- Si la división de un nodo crea un nodo hijo de tamaño menor que el determinado por el usuario, el nodo no se divide.
- Si para la mejor división del nodo, la mejora $\Delta I(s,t) = p(t)\Delta i(s,t)$ es menor que el determinado por el usuario, el Complexity Parameter, el nodo no se divide.

Otra forma de comprender los árboles de decisión es mediante la ganancia de información, la entropía y la ganancia, en el caso de clasificación según los algoritmos ID3 y C4.5.

El primer paso es encontrar la ganancia de información mediante la siguiente fórmula, siendo p y n 0,1 según la clase:

$$I(p,n) = \frac{-p}{p+n} log_2\left(\frac{p}{p+n}\right) - \frac{p}{p+n} log_2\left(\frac{n}{p+n}\right)$$

Seguidamente se consigue la entropía de cada atributo:

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n}(I(p,n))$$

Después se calcula la ganancia:

$$Gain = I(p,n) - E(A)$$

Cuando se realiza este cálculo para todos los atributos, se ordenan las ganancias para que la ganancia máxima sea el nodo raíz, del cual descienden las opciones posibles para ese valor. Se comprueban los valores para ver si se puede seguir aumentando el árbol o si se llega a nodos que estén directamente relacionados con las clases. En caso necesitar otra división de nodos se deben realizar de nuevo todos los pasos anteriores, pero adaptándolo solo a los datos que quedan.

Los árboles de decisión tienen ventajas respecto a otras formas de Machine Learning ya que son fáciles de interpretar y de entender. Además, pueden manejar datos numéricos y categóricos y funcionan bien con grandes cantidades de datos.

Uno de los problemas de este modelo es que obtener un árbol de decisión óptimo es un problema NP-completo. También pueden generarse árboles muy complejos a partir de unos datos de entrenamiento que no funcionen bien después con otros datos.

4.2.1 Recursive Partitioning

El algoritmo implementado en Recursive Partitioning no sigue los métodos Gini generalizado o Twoing, sino que implementa el método de las probabilidades a priori alteradas [19].

Este método consiste en que el riesgo del nodo A es:

$$R(A) = \sum_{i=1}^{C} p_{iA} L(i, \tau(A)) = \sum_{i=1}^{C} \pi_i L\big(i, \tau(A)\big)\,(n_{iA}/n_i)(n/n_A)$$

Siendo $\tau(A)$ la clase asignada a A si fuese el último nodo, n el número de observaciones, $L\big(i, \tau(A)\big)$ la matriz de pérdidas por clasificar de forma errónea y C el número de clases.

Si se asume que para todas las clases:

$$\tilde{\pi}_i \tilde{L}(i,j) = \pi_i L(i,j)$$

Entonces R(A) no cambia. Los valores de $\tilde{\pi}_i$ tienen que utilizarse en el criterio de división si:

$$L(i,j) = \begin{cases} L_i & i \neq j \\ 0 & i \neq j \end{cases} \qquad \tilde{\pi}_i = \frac{\pi_i L_i}{\sum_j \pi_j L_j} \qquad para\ C = 2$$

$$L(i,j) = \begin{cases} L_i & i \neq j \\ 0 & i \neq j \end{cases} \qquad \tilde{\pi}_i = \frac{\pi_i L_i}{\sum_j \pi_j L_j} \qquad L_i = \sum_j L(i,j) \quad para\ C > 2$$

Otra justificación de este método es que el índice de impureza, $I(A) = \sum f(p_i)$ tiene el máximo en $p_1 = p_2 = \cdots = p_C = 1/C$. La técnica de alterar las probabilidades a priori desplaza p_i.

Los otros dos algoritmos utilizados en el trabajo son Random Forest y Bagged CART y ambos se basan en combinar varios árboles de decisión para obtener un mejor rendimiento que con un único árbol, aunque son más difíciles de interpretar, ya que con un único árbol se puede seguir el camino del nodo a las hojas para tomar una decisión.

4.2.2 Bagged CART

El algoritmo Bagged CART se utiliza para reducir la varianza del algoritmo CART. Debido a que los árboles de decisión son muy sensibles a los datos con los que han sido entrenados, puede haber una gran diferencia de árboles resultantes con predicciones muy distintas.

El método resultante consiste en [20]:

- Crear muchos subconjuntos a partir de los datos originales mediante muestreo con reemplazo. Esto significa que algunas observaciones se repiten en los distintos subconjuntos. Se denomina muestra bootstrap y el nombre Bagged tiene su origen en Bootstrap Aggregation.
- Se entrena cada subconjunto con un modelo de árboles de decisión.
- Según la media de las decisiones de cada árbol se calcula la predicción.

En este modelo no es preocupante que los árboles sean muy densos y específicos, aunque es posible que si se tiene un gran número de árboles el programa tome mucho tiempo.

4.2.3 Random Forest

Este método es una extensión del algoritmo Bagged CART ya que además de utilizar los subconjuntos de datos, también selecciona de forma aleatoria las características para hacer crecer los árboles, su parámetro de ajuste [20].

En el caso de clasificación, un buen número es mtry=sqrt(variables de entrada), para regresión, un buen número es mtry=(variables de entrada)/3.

Al igual que en el modelo de Bagged CART, un gran número de árboles relativamente incorrelados mejoran el rendimiento de aquellos individuales, ya que se protegen en conjunto de sus errores individuales. Además, puede seguir funcionando correctamente en caso de que falten datos.

Una gran ventaja de este método es la facilidad de manejar grandes datos y tienen una gran utilidad en motores de búsqueda y clasificación de imágenes.

4.3 KNN k-Nearest Neighbours

Este algoritmo es una forma de clasificación supervisada que se utiliza para estimar la función de densidad de los predictores en cada clase [21]. Asume que hay cosas similares en su proximidad y se basa en la idea de la proximidad o cercanía.

A diferencia de otras formas de aprendizaje supervisado, que generan un modelo a través de los datos de entrenamiento, el aprendizaje ocurre al probar los datos de test, y se considera un método de aprendizaje vago.

Los pasos que sigue el algoritmo son:

- Cargar los datos e inicializar un número K de vecinos.
- Para cada ejemplo en los datos se calcula la distancia entre el test y los datos de entrenamiento y se apunta de forma ordenada. Normalmente se utiliza la distancia euclídea.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{p} (x_{ri} - x_{rj})^2}$$

- Se ordena esta colección de menor a mayor y se seleccionan los primeros K valores.
- En clasificación se devuelve el tipo de las K etiquetas de clase.

Para seleccionar un valor de K correcto para los datos que se tienen se puede ejecutar el algoritmo con distintos valores y examinar los resultados.

Se debe tener en cuenta que si el valor de K es muy bajo las predicciones son muy poco estables porque puede que el punto más cercano sea uno de otra clase, pero que haya más puntos de la clase correcta situados un poco más lejanos.

A medida que se aumenta el valor de K, si llega a ser demasiado alto se comienzan a ver más errores, por lo que habría que seleccionar el valor de K correcto antes de darse este fenómeno.

Una técnica útil para que los vecinos más cercanos aporten más que los lejanos es darles cargas que dependen de la distancia.

Es también recomendable que este valor de K sea impar, para poder desempatar las clases.

Este método es simple y fácil de implementar y tiene la ventaja de no necesitar muchos parámetros y funcionar para varios tipos de problemas de clasificación y regresión. Aunque, tiene la desventaja de que a medida que la cantidad de ejemplos aumenta, se vuelve más lento.

4.4 SVM Support Vector Machine

Una máquina de soporte vectorial es un clasificador definido por un hiperplano [22]. Dados unos datos de entrenamiento etiquetados, el algoritmo devuelve un hiperplano en un espacio de dimensión N, siendo N el número de características, que clasifica los datos.

En el caso de dos, el hiperplano es una línea que divide el plano en dos lados y en cada uno se encuentra un tipo. Si el número de características es 3, el hiperplano es un plano.

Si una línea no puede separar los dos tipos, se añaden dimensiones de forma que se pueda establecer una nueva separación. Estas transformaciones se denominan kernels.

Si los datos se superponen, se debe establecer un parámetro de regularización y gamma, con los que se pueden obtener resultados con mayor o menor exactitud.

Los parámetros básicos de este método son:

- **Kernel:** La ecuación para transformar los datos. Si el kernel es lineal se tiene un clasificador lineal, si se utiliza uno no lineal, se obtiene uno no lineal. Existen diversas funciones de Kernel.
    - Lineal: $K(x, y) = x^T y$
    - Polinomial: $K(x, y) = (x^T y + r)^n$
    - Gaussiano: $K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$
- **Regularización, C:** Cuánto se quiere evitar un error en la clasificación. Si se trata de un valor alto, todos los valores de entrenamiento estarán clasificados correctamente y si es un valor bajo puede tener errores.
- **Sigma:** Cuánto influyen los puntos de entrenamiento.

Una SVM busca alcanzar el mejor margen, el que consigue una gran separación entre los datos y el hiperplano.

Un hiperplano se puede escribir como el conjunto de puntos $\vec{x}$ que cumplen la siguiente ecuación en la que $\vec{w}$

$$\vec{w} \cdot \vec{x} - b = 0$$

La función que permite maximizar el margen es la minimización de:

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x_i} - b)\right)\right] + \frac{1}{C}\|w\|^2$$

Las SVM pueden utilizarse en diversos problemas como categorización de textos, clasificación de imágenes y reconocimiento de caracteres manuscritos.

Sin embargo, tiene varias desventajas ya que necesita el etiquetado de todos los datos de entrada y solo puede aplicarse de forma directa cuando hay dos clases, por lo que para hacerlo funcionar en caso de un problema multiclase la solución más habitual es reducir el problema a uno de clasificación binaria.

4.5 Multinomial Logistic Regression

La regresión logística polinomial es una forma de clasificación que generaliza la regresión logística para problemas multiclase [23] [24].

Es un método que puede predecir las probabilidades de los resultados de una variable dependiente categórica con más de 2 categorías a partir de las variables independientes. Para ello es necesario construir un modelo que las relacione, para que así cuando se tengan las medidas correspondientes a un nuevo test, se pueda predecir la clase correcta.

El modelo asume que cada variable tiene un valor para cada caso y que la variable dependiente no puede obtenerse perfectamente partiendo de las variables independientes. No es necesario que las variables independientes sean estadísticamente independientes, pero se asume que la colinealidad es baja.

La idea principal es construir una función que devuelva un resultado a partir de una serie de coeficientes y los valores de la observación.

En este modelo, el predictor lineal, dicho resultado es la probabilidad de una observación acabando en un determinado resultado.

$$f(k,i) = \beta_{0,k} + \beta_{1,k}x_{1,i} + \beta_{2,k}x_{2,i} + \cdots + \beta_{M,k}x_{M,k} \qquad \leftrightarrow \qquad f(k,i) = \beta_k \cdot X_i$$

β_k son los coeficientes de regresión del resultado k, y x_i son las variables de la observación i.

Si se considera que en un modelo multiclase se pueden tener K resultados, y se opta por seleccionar uno de ellos como base contra los demás, se obtiene para cada resultado la siguiente expresión, escrita para el primer caso:

$$\ln\frac{\Pr(Y_i = 1)}{\Pr(Y_i = K)} = \beta_1 \cdot X_i \qquad \leftrightarrow \quad \Pr(Y_i = 1) = \Pr(Y_i = K)e^{\beta_1 \cdot X_i}$$

Ya que la suma de todas las probabilidades es 1, se puede llegar a las siguientes expresiones que como en el caso anterior la que está escrita del resultado 1 es similar a la de los demás resultados hasta $K - 1$:

$$\Pr(Y_i = K) \,?\, 1 - \sum_{k=1}^{K-1} \Pr(Y_i = k) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot X_i}}$$

$$\Pr(Y_i = 1) = \frac{e^{\beta_1 \cdot X_i}}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot X_i}}$$

Normalmente los coeficientes se estiman mediante la estimación Máximo a Posteriori, MAP.

En el caso log-lineal, que es el que utiliza el modelo implementado en este trabajo, se añade un factor de normalización, el logaritmo de la función de partición para asegurar que la suma de todas las probabilidades es 1:

$$\ln \Pr\,(Y_i = 1) = \beta_1 \cdot X_i - \ln Z$$

El usar la suma de ese factor de normalización en lugar de su multiplicación se debe a que se utiliza el logaritmo de las probabilidades.

Al quitar los logaritmos de ambos lados el resultado es la medida de Gibbs:

$$\Pr(Y_i = 1) = \frac{1}{Z} e^{\beta_1 \cdot X_i}$$

De igualar la suma de todas las probabilidades a 1, se obtiene el valor de Z y las probabilidades finales:

$$Z = \sum_{k=1}^{K} e^{\beta_k \cdot X_i}$$

$$\Pr(Y_i = 1) = \frac{e^{\beta_1 \cdot X_i}}{\sum_{k=1}^{K} e^{\beta_k \cdot X_i}}$$

Se utiliza la función SoftMax para exagerar las diferencias entre los valores de la observación dando un valor cercano a 0 si es un valor mucho más bajo que el máximo y 1 si se aplica al mayor:

$$softmax(k, x_1, x_2, \ldots, x_n) = \frac{e^{x_k}}{\sum_{k=1}^{n} e^{x_n}}$$

Y las ecuaciones de probabilidad se pueden escribir como:

$$\Pr(Y_i = c) = softmax(c, \beta_1 \cdot X_i, \ldots, \beta_k \cdot X_i)$$

5. Desarrollo práctico

La primera parte de este capítulo está dedicada al lenguaje utilizado para desarrollar el trabajo, R, los modelos de aprendizaje autónomo evaluados y los paquetes cuya instalación es necesaria para la ejecución de dichos modelos. Seguidamente, se continúa con la descripción de los datos que se emplean y se detalla el código utilizado para realizar el análisis del rendimiento de los distintos algoritmos de aprendizaje autónomo sobre los datos.

5.1 R

R es un lenguaje de programación y un entorno software dedicado al análisis estadístico apoyado por la R Foundation for Statistical Computing [25] y distribuido bajo la licencia GNU General Public License. Es, junto con Python, uno de los lenguajes de programación más utilizados en el análisis de datos y Machine Learning por parte de investigadores y analistas. Cabe destacar que, en el momento de la redacción de este trabajo, Julio 2020, este lenguaje se encuentra en el octavo puesto del índice TIOBE [26] de lenguajes de programación, mientras que el top tres son: C, Java y Python. Se ha optado por este lenguaje porque, comparado con Python, éste es el más utilizado en el ámbito académico y la investigación científica.

Este lenguaje permite la manipulación y almacenamiento de datos, obtener gráficas de alta calidad, utilizar un formato de documentación propio basado en LaTeX, utilizar diversas técnicas de análisis estadístico y su funcionalidad es incluso ampliable mediante el uso de distintos paquetes según la finalidad deseada. Estos paquetes y el software R se pueden descargar de cualquiera de los mirrors CRAN (Comprehensive R Archive Network) que hay distribuidos por todo el mundo. En el caso de este trabajo, se ha optado por utilizar el mirror de RedIRIS, la red académica y de investigación que proporciona servicios de comunicaciones a la comunidad científica y universitaria española [27].

5.2 CARET

El paquete caret [28], Classification And Regression Training, contiene herramientas para procesar los datos y múltiples funciones para facilitar la ejecución de los modelos de aprendizaje autónomo. Este paquete, que fue creado en 2016 por Max Kuhn y en continua actualización, puede encontrarse en CRAN.

Se ha optado por este paquete ya que permite el uso de funciones concretas para tareas específicas de la realización de este trabajo, por ejemplo, tiene funciones para cada uno de los modelos que fueron descritos en el capítulo 4, lo que facilita el desarrollo práctico de este proyecto. Además, pese a que R tiene muchas funciones de modelado, éstas pueden tener una sintaxis distinta. Este paquete hace que esas funciones tengan una forma estándar entre todas.

De los múltiples comandos de caret, el más importante es train, cuya función es aplicar el método de clasificación deseado con sus correspondientes parámetros.

Para utilizarlo, se necesita determinar qué modelo quiere evaluarse, fijar los parámetros de ajuste, si el modelo los requiere, y definir el método de remuestreo.

En el comando train hay siempre tres argumentos principales:

- La variable que se quiere predecir.
- El data frame en el que están los datos.
- El método de clasificación.

Además, en el desarrollo de este trabajo se utiliza validación cruzada de 10 iteraciones y se mide el modelo según su *Accuracy*.

El primer valor, validación cruzada, es un método utilizado en la evaluación de los resultados en estadística que garantiza que dichos resultados son independientes. Es una técnica aplicada cuando el objetivo es predecir la precisión de un modelo y sirve para validar los modelos.

Las 10 iteraciones es el número estándar de subconjuntos en los que se dividen los datos de entrenamiento para utilizar una de estas particiones como datos de prueba y los otros 9 restantes como los de entrenamiento. Para obtener un único resultado se realiza la media aritmética de los resultados obtenidos en cada iteración.

En el contexto de la estadística y del aprendizaje autónomo, *Accuracy,* es la exactitud de un modelo, o el porcentaje de predicciones que el modelo realiza correctamente, que puede obtenerse mediante la división del número de predicciones correctas entre el número total de predicciones.

A continuación, se indican brevemente las opciones utilizadas en este trabajo.

5.2.1 LDA Linear Discriminant Analysis

El método ha sido implementado utilizando el método "lda" para clasificación que usa el paquete MASS [29] sin parámetros de ajuste.

El paquete MASS consta de funciones y datasets para soportar "Modern Applied Statistics with S" de Venables y Ripley (4ª edición, 2002).

5.2.2 Recursive Partitioning

Se utiliza el método "rpart" para clasificación y regresión utilizando el paquete rpart [30] con el parámetro de ajuste:

- **Parámetro de complejidad** (cp, numeric): mínima mejora que el modelo necesita en cada nodo. El valor por defecto es 0.01. Es un parámetro de parada del algoritmo que sirve para determinar si se continúa añadiendo una rama al árbol de decisión. Si el beneficio de esa rama es menor de 0.01, no se añade. En el caso de dar un valor al parámetro de 0.0, se tiene el árbol más complejo posible.

El paquete rpart implementa la mayoría de las ideas del libro de 1984 sobre CART de Breiman, Friedman, Olshen y Stone. Debido a que CART es una marca de una implementación de software, se utilizó la denominación de Recursive Partitioning o rpart para hacer este software libre.

5.2.3 Bagged CART

El segundo método de árboles de decisión implementado es el "treebag", para clasificación y regresión utilizando los paquetes ipred, plyr y e1071 sin parámetros de ajuste.

- El paquete ipred [31] consta de modelos de predicción mejorados mediante clasificación indirecta y agregación para los problemas de clasificación y regresión.
- El paquete plyr [32] contiene una serie de herramientas que permiten dividir un problema en varias piezas, operar sobre cada pieza y luego volver a unirlas todas.
- El paquete e1071 [33] contiene múltiples funciones del Departamento de Estadística, Grupo de Teoría de la Probabilidad de TU Wien.

5.2.4 Random Forest

El tercer método basado en CART es el "rf", para clasificación y regresión utilizando el paquete randomForest [34] con el parámetro de ajuste:

- **Número de predictores aleatoriamente seleccionados** (mtry, numeric). El valor por defecto de este parámetro es para clasificación la raíz cuadrada del número de variables de entrada y para regresión el número de variables de entrada dividido entre 3.

El paquete randomForest implementa la clasificación y regresión basada en un conjunto de árboles de decisión de Breiman (2001).

5.2.5 KNN k-Nearest Neighbours

La implementación de esta forma se ha realizado mediante el método "knn", para clasificación y regresión con el parámetro de ajuste:

- **Número de vecinos** (k, numeric). Este valor en caret por defecto empieza en k=5 y se incrementa de dos en dos.

5.2.6 SVM – Support Vector Machine

En este trabajo se ha utilizado el modelo "Support Vector Machines with Radial Basis Function Kernel", "svmRadial", que sirve para clasificación y regresión utilizando el paquete kernlab [35] con los parámetros de ajuste:

- **Sigma** (sigma, numeric). Si no se especifica lo contrario, este parámetro toma por valor 1/(dimensión de los datos).

- **Coste** (C, numeric). Por defecto, este método utiliza C=1.

El paquete kernlab trata diversos métodos de aprendizaje automático basados en Kernel: clasificación, regresión, clustering, etc.

5.2.7 Multinomial Logistic Regression

En este trabajo se ha utilizado el modelo “Penalized Multinomial Regression”, “multinom” que sirve para clasificación usando el paquete nnet [36] con el parámetro de ajuste:

- **Weight decay** (decay, numeric). El valor por defecto es 0.

Este parámetro se debe a que el paquete nnet corresponde a redes neuronales. En los casos en los que hay una gran cantidad de datos, hay también una gran cantidad de interacciones en la red, y a veces no es deseable que haya demasiadas. Utilizar “weight decay” es una técnica de regularización para evitar el sobreajuste del modelo.

El paquete nnet contiene software para redes neuronales feed-forward, aquellas en las que la información solo se mueve desde los nodos de entrada a los de salida, sin bucles en la red.

5.3 Datos de estudio

Los datos utilizados como secuencias de entrenamiento de los algoritmos son las medidas de potencia detectadas en la planta -2 del edificio de I+D+i de la Universidad de Cantabria y obtenidas en el Trabajo de Fin de Grado de Andrei Pietro Sciddurlo Juaristi [2]. Estos datos fueron facilitados por el tutor del trabajo, Alberto Eloy García Gutiérrez.

Se mantiene la misma disposición de la planta -2 planteada en dicho Trabajo de Fin de Grado. Como se puede apreciar en la siguiente imagen, la mayoría de los despachos de la planta están divididos en 2 mitades excepto el despacho 225, que debido a su tamaño está dividido en 4. En la imagen también se muestra dónde se encuentran las nueve balizas Bluetooth en cada despacho.

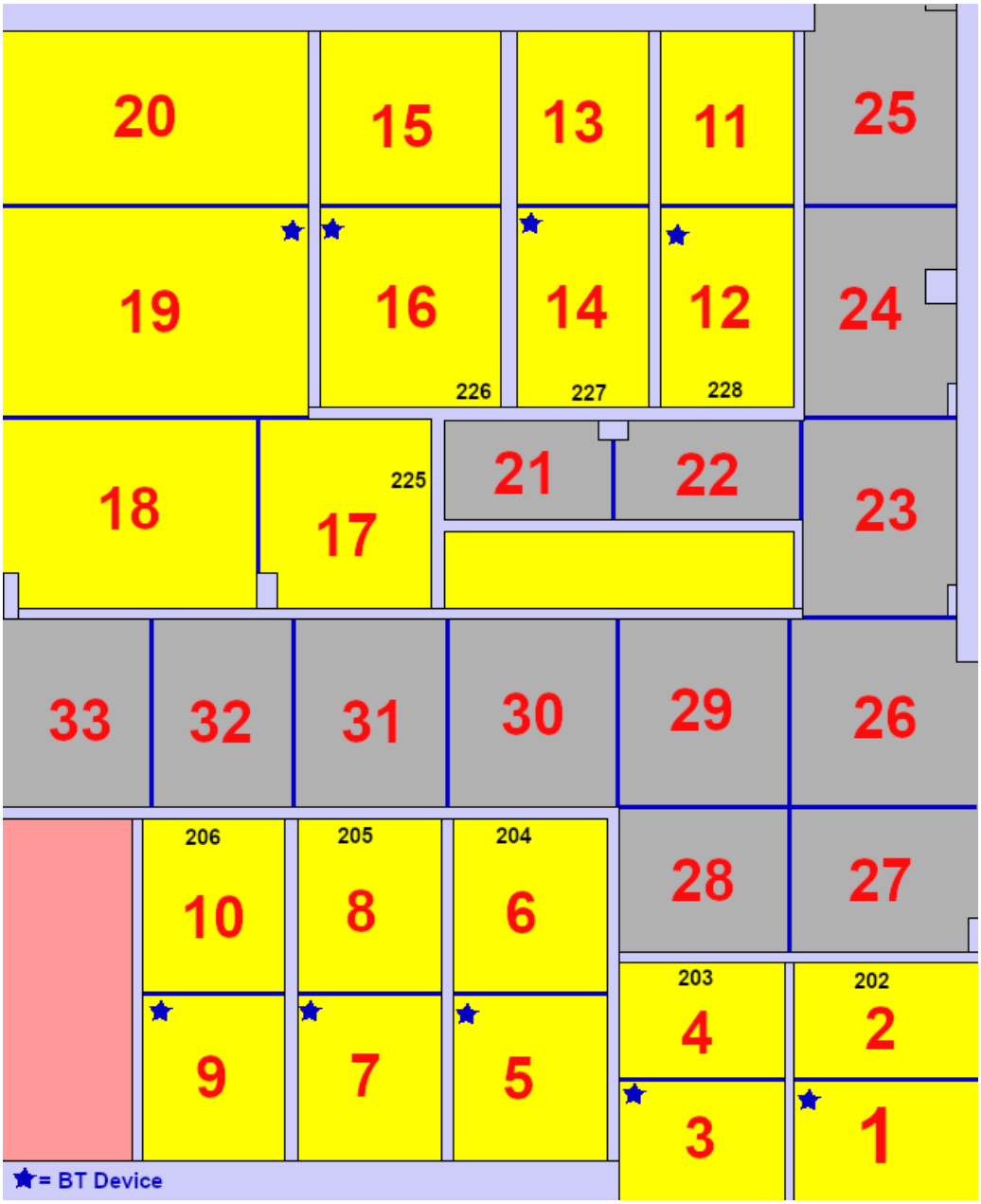

Figura 3. División de zonas.

Debido al tiempo de operación de las baterías de las balizas Bluetooth, las medidas fueron tomadas en 5 escaneos distintos durante 3 días seguidos, realizando medidas por la mañana, 9:30, y por la tarde, 17:00. También se tomaron las medidas de los 88 puntos de acceso Wi-Fi activos en todos los escaneos.

Los datos están organizados de forma que la primera fila de la tabla corresponde a los identificadores de cada baliza Bluetooth o punto de acceso Wi-Fi, la primera columna es la zona donde se tomó la medida y los datos que llenan la tabla son las potencias medidas para cada cabecera MAC.

Combinando los cinco escaneos, se tienen 3795 filas de medidas. En Bluetooth se tienen 10 columnas, 1 de zona y 9 de cada baliza, en Wi-Fi se tienen 89 columnas, 1 de zona y 88 de cada punto de acceso y en Bluetooth+Wi-Fi hay 98 columnas, ya que aparecen las balizas Bluetooth y los puntos de acceso Wi-Fi. En las siguientes tablas se puede ver una pequeña muestra de la organización de los datos.

ZONA	C8:FD:19:11:96:A6	C8:FD:19:11:98:A6	C8:FD:19:11:9C:93	C8:FD:19:11:9F:53
1	-120	-99	-120	-120
1	-94	-103	-120	-120
1	-92	-120	-120	-120
1	-94	-120	-120	-120

Tabla 1. Formato de los datos Bluetooth.

ZONA	00:12:17:A7:C6:77	00:12:17:A7:C6:78	00:1B:1B:84:7B:C0	00:1B:1B:85:C4:00
1	-78	-59	-120	-84
1	-78	-59	-120	-84
1	-78	-74	-120	-120
1	-78	-74	-120	-120

Tabla 2. Formato de los datos Wi-Fi.

ZONA	C8:FD:19:11:96:A6	C8:FD:19:11:98:A6	00:12:17:A7:C6:77	00:12:17:A7:C6:78
1	-120	-99	-78	-59
1	-94	-103	-78	-59
1	-92	-120	-78	-74
1	-94	-120	-78	-74

Tabla 3. Formato de los datos Bluetooth+Wi-Fi

Además de las medidas utilizadas para entrenar los modelos, se tienen también las medidas de dos trayectos en movimiento por Bluetooth, un trayecto por Wi-Fi y un trayecto con medidas en ambos sistemas.

- El trayecto Bluetooth 1 corresponde a la salida de la planta a través de las escaleras hacia la planta superior desde el despacho 226. Con la prueba de un trayecto así se pretende analizar el planteamiento de un movimiento común de los usuarios dentro del edificio moviéndose entre la planta -1 y -2.

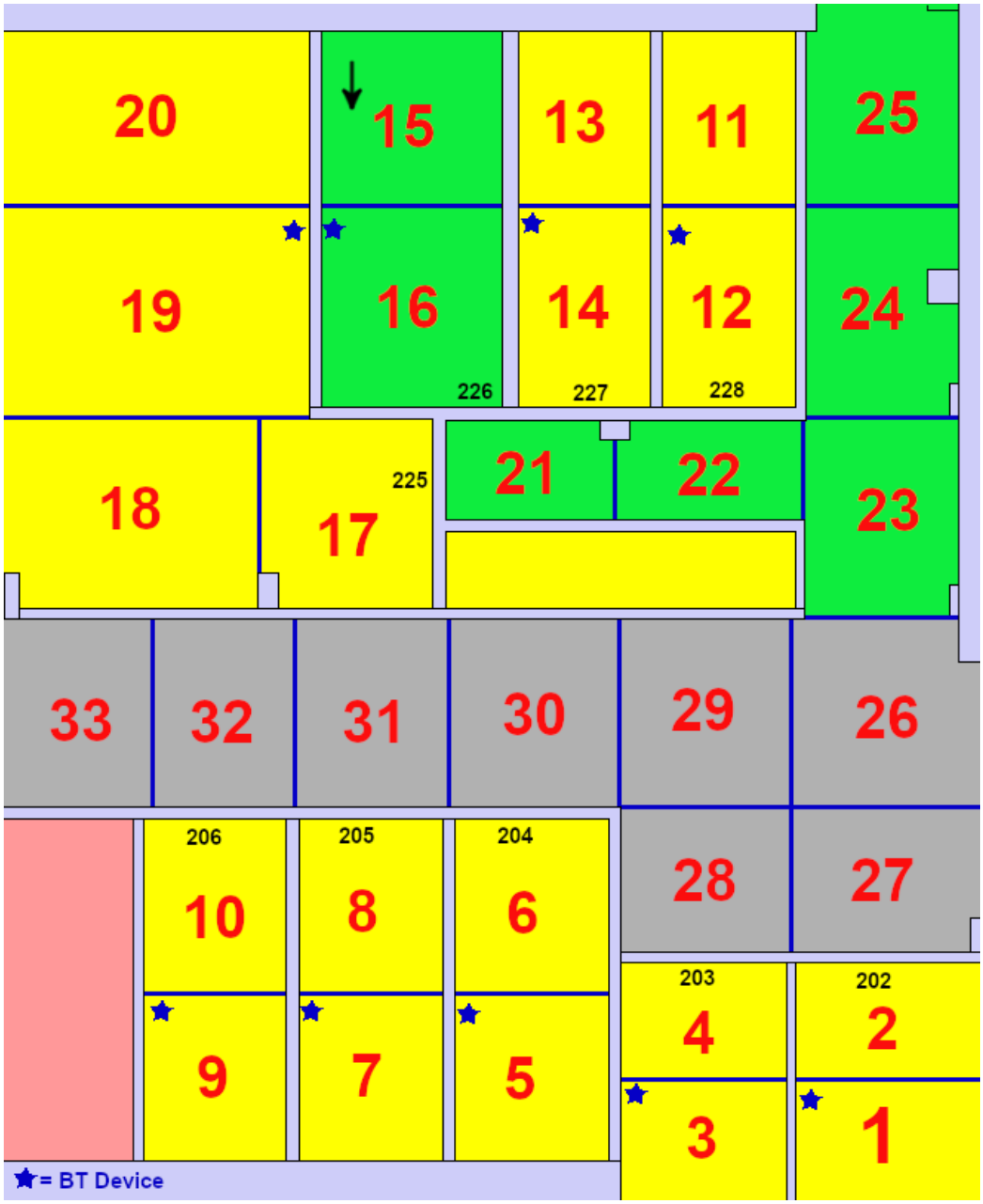

Figura 4. Trayecto Bluetooth 1.

- El trayecto Bluetooth 2 corresponde a la salida del edificio por la puerta principal desde el despacho 206. Este planteamiento sirve para analizar el problema del usuario en movimiento saliendo de uno de los despachos hacia el exterior caminando a lo largo del pasillo.

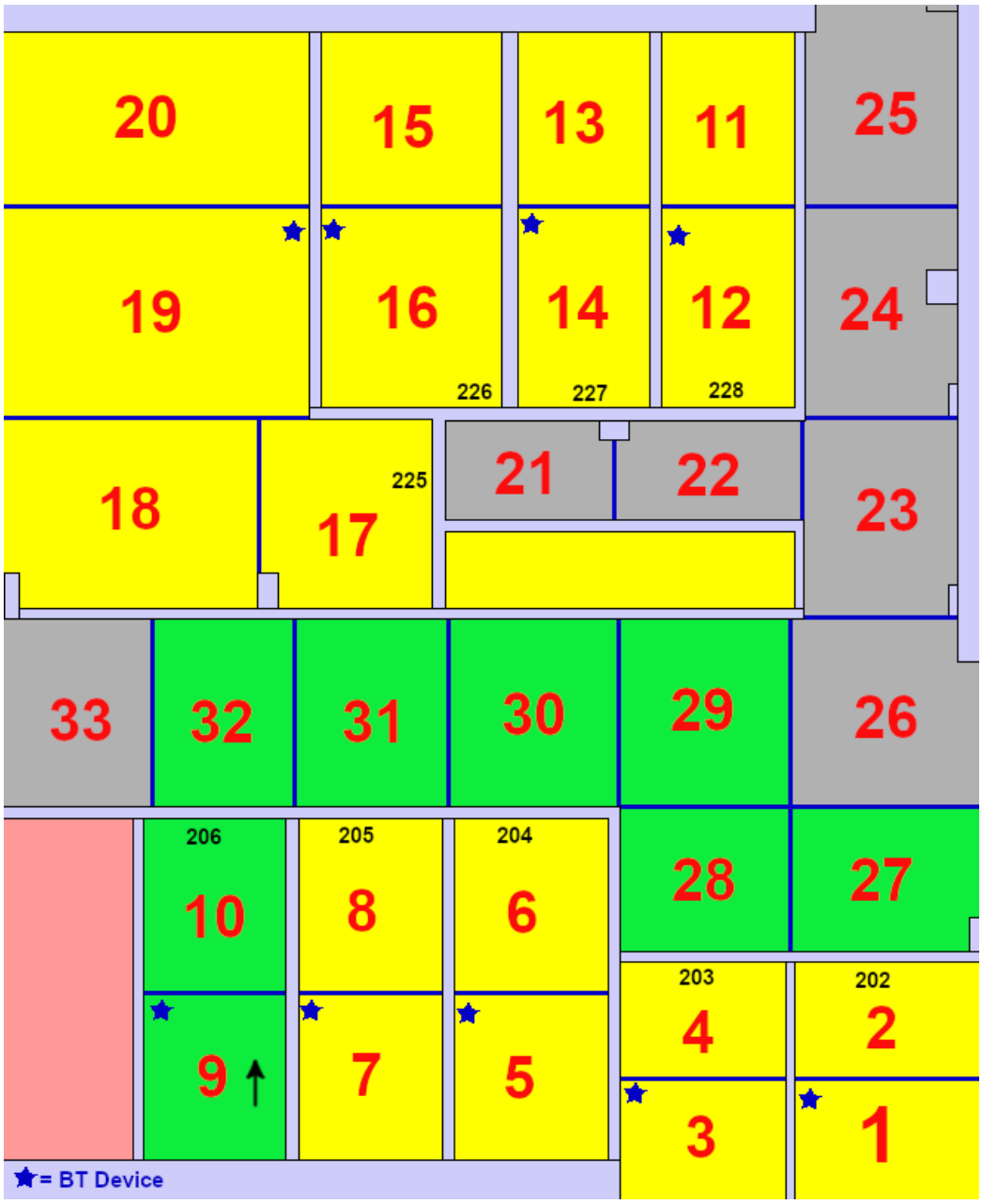

Figura 5. Trayecto Bluetooth 2.

- El trayecto Wi-Fi 1 corresponde a un recorrido desde el extremo del pasillo hasta el despacho 225. Es un problema mayor a los trayectos de Bluetooth ya que en este caso la ruta tiene lugar en zonas adyacentes separadas por paredes. Además, en el caso Bluetooth ambas rutas pasan en algún punto del trayecto por un despacho con baliza Bluetooth, mientras que los puntos de acceso Wi-Fi están situados por todo el mapa.

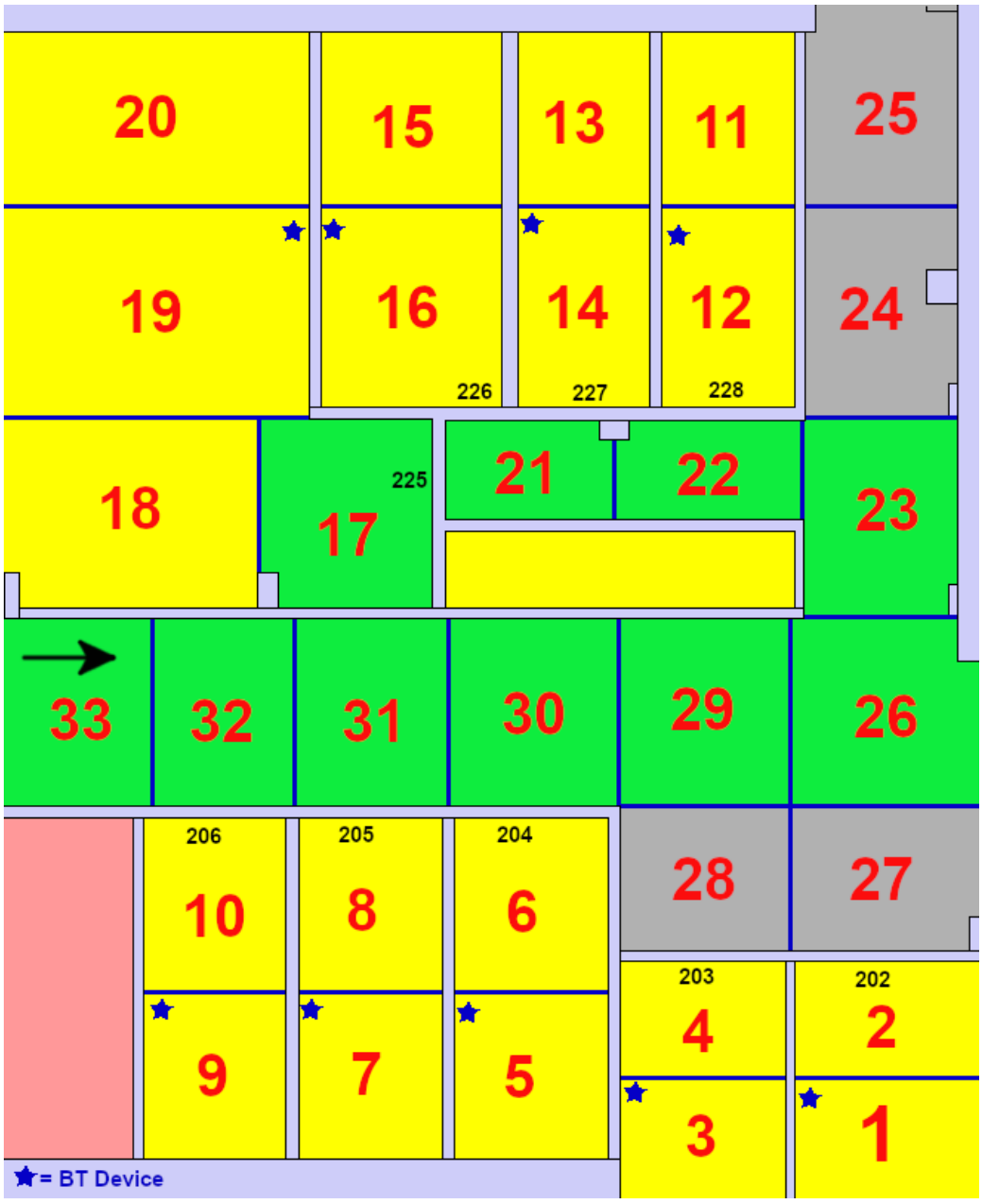

Figura 6. Trayecto Wi-Fi

- El trayecto combinado corresponde al recorrido desde el comienzo del pasillo del despacho 225 hasta la puerta del despacho 226. Es una ruta similar a la del caso de solo Wi-Fi, pero con la diferencia de que ésta no entra en los despachos, solo se estudia la localización en los pasillos.

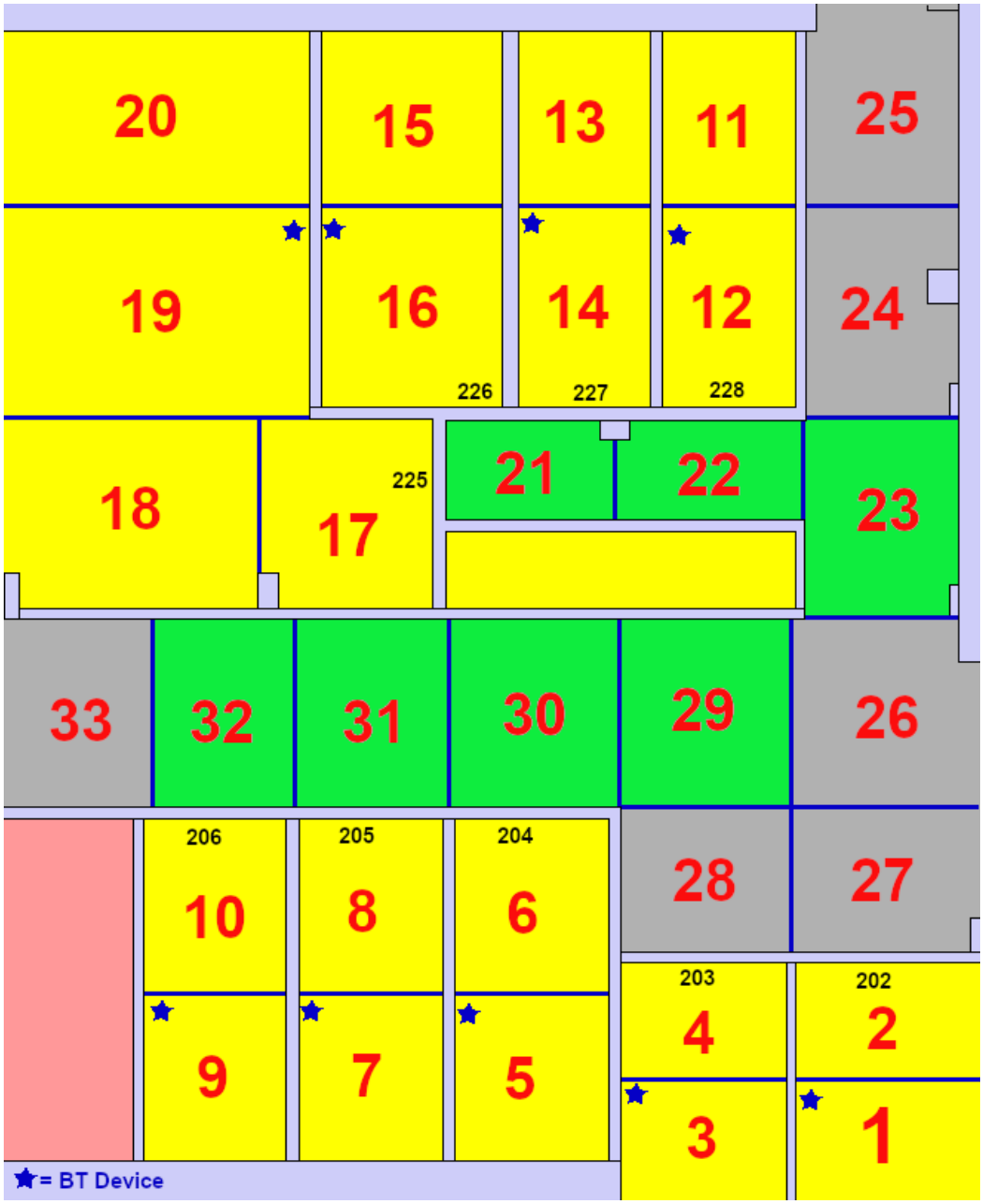

Figura 7. Trayecto Bluetooth+Wi-Fi

5.4 Desarrollo

Durante el desarrollo de este trabajo se ha instalado la herramienta R sobre la que se trabaja con los distintos conjuntos de datos: las tablas de datos de entrenamiento Bluetooth o Wi-Fi se cargan en el programa, se analizan las características de dichos datos, se evalúan los distintos algoritmos comprobando el tiempo que tardan, se comparan los modelos, se importan los datos de los trayectos Bluetooth o Wi-Fi y se obtiene la matriz de confusión entre las predicciones obtenidas por los modelos de Machine Learning y las medidas de los trayectos reales.

5.4.1 Instalación de R

Para trabajar con R se debe instalar este software. Debido a que este proyecto ha sido realizado sobre Windows se optó por descargar R desde la página web de R Project utilizando el mirror de RedIRIS.

Aunque se pueden utilizar IDEs como RStudio para trabajar con este sistema, se optó por utilizar directamente la consola R, que no necesita ninguna instalación más.

Para aplicar los modelos de ML sobre R se debe instalar el paquete caret que fue comentado en el capítulo 5.2. junto con todos los paquetes necesarios de los que depende para su correcto funcionamiento.

```
library(caret, quietly = TRUE)
```

5.4.2 Programa

Los estudios sobre Bluetooth, Wi-Fi y con las medidas combinadas, se han realizado tanto con cada uno de los 5 escaneos como con el conjunto de dichos escaneos. En total se ha probado 6 veces con cada tipo de medida. Los programas utilizados siguen los mismos pasos, con la única diferencia de los archivos que se cargan y las modificaciones necesarias que deben hacerse en los datos de validación de cada ruta evaluada que se comenta más adelante.

Para el caso Wi-Fi y datos combinados, se ha optado por omitir el algoritmo *multinom* ya que causa error en la ejecución del programa, pero por lo demás sigue los mismos pasos.

Debido a que el proceso a seguir en todos los casos ha sido similar [37], se opta por detallar los pasos del programa utilizado sobre los datos del primer escaneo de Bluetooth, que aparece en el anexo junto con los otros dos programas, Wi-Fi y datos combinados.

El primer paso es cargar los datos .csv en el programa, junto con las cabeceras, y determinar que la primera columna, en la que aparecen las zonas, sea de tipo *factor*, ya que es la que contiene el valor de salida, la zona del mapa.

```
datos <- "Dataset_Alberto_Scan1_BT.csv"

dataset <- read.csv(datos, header = TRUE, sep=";")

dataset$ZONA <- as.factor(dataset$ZONA)
```

El siguiente paso es determinar *Accuracy* como forma de evaluar los modelos y especificar el uso de validación cruzada de 10 iteraciones, que consiste en repetir y calcular en distintas particiones.

```
control <- trainControl(method="cv", number=10)

metric <- "Accuracy"
```

A continuación, se construyen los 7 modelos de aprendizaje mediante la instrucción t*rain*, que fue descrita en el capítulo 5.2. Además, se han añadido comandos al principio y final de cada *train* para calcular el tiempo que tarda cada modelo y se reinicia el número aleatorio antes de cada algoritmo para asegurar que los resultados puedan ser comparables.

```
# a) linear algorithms

set.seed(7)

start_time_lda <- Sys.time()

fit.lda <- train(ZONA~., data=dataset, method="lda", metric=metric, trControl=control)

end_time_lda <- Sys.time()

end_time_lda - start_time_lda

Time difference of 0.7186182 secs

# b) nonlinear algorithms

# CART

set.seed(7)

start_time_cart <- Sys.time()

fit.cart <- train(ZONA~., data=dataset, method="rpart", metric=metric, trControl=control)

end_time_cart <- Sys.time()

end_time_cart - start_time_cart

Time difference of 0.8748009 secs

# kNN

set.seed(7)

start_time_knn <- Sys.time()

fit.knn <- train(ZONA~., data=dataset, method="knn", metric=metric, trControl=control)

end_time_knn <- Sys.time()

end_time_knn - start_time_knn

Time difference of 2.456062 secs
```

```
# c) advanced algorithms

# SVM

set.seed(7)

start_time_svm <- Sys.time()

fit.svm <- train(ZONA~., data=dataset, method="svmRadial",
metric=metric, trControl=control)

end_time_svm <- Sys.time()

end_time_svm - start_time_svm

Time difference of 42.68206 secs

# Random Forest

set.seed(7)

start_time_rf <- Sys.time()

fit.rf <- train(ZONA~., data=dataset, method="rf", metric=metric,
trControl=control)

end_time_rf <- Sys.time()

end_time_rf - start_time_rf

Time difference of 24.06981 secs

#Bagged CART

set.seed(7)

start_time_bcart <- Sys.time()

fit.bcart <- train(ZONA~., data=dataset, method="treebag",
metric=metric, trControl=control)

end_time_bcart <- Sys.time()

end_time_bcart - start_time_bcart

Time difference of 4.367335 secs

#Penalized Multinomial Regression

set.seed(7)

start_time_multinom <- Sys.time()

fit.multinom <- train(ZONA~., data=dataset, method="multinom",
metric=metric, trControl=control)

end_time_multinom <- Sys.time()

end_time_multinom - start_time_multinom

Time difference of 41.09269 secs
```

Una vez se tienen los 7 algoritmos se aplican los siguientes comandos para poder ver cuál de ellos es el que mejor resultado obtiene.

```
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn,
svm=fit.svm, rf=fit.rf, bcart=fit.bcart, multinom=fit.multinom))
summary(results)
Call:
summary.resamples(object = results)

Models: lda, cart, knn, svm, rf, bcart, multinom
Number of resamples: 10

Accuracy
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lda      0.6666667 0.7036983 0.7087379 0.7106349 0.7355769 0.7425743    0
cart     0.3203883 0.3599642 0.3762376 0.3704708 0.3840498 0.4077670    0
knn      0.6960784 0.7506188 0.7586391 0.7517001 0.7714764 0.7864078    0
svm      0.6764706 0.7554969 0.7623762 0.7585552 0.7777621 0.8118812    0
rf       0.7941176 0.8463664 0.8639706 0.8640650 0.8886529 0.9215686    0
bcart    0.7647059 0.7977266 0.8374102 0.8258924 0.8461538 0.8823529    0
multinom 0.6176471 0.6529126 0.6730293 0.6744223 0.7012473 0.7156863    0

Kappa
             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
lda      0.6562934 0.6943365 0.6995186 0.7015344 0.7272745 0.7344524    0
cart     0.2993878 0.3406696 0.3569804 0.3511148 0.3651691 0.3895258    0
knn      0.6865894 0.7427786 0.7510269 0.7438987 0.7643343 0.7797220    0
svm      0.6664354 0.7477835 0.7549163 0.7509804 0.7707879 0.8059067    0
rf       0.7877316 0.8415326 0.8596656 0.8597959 0.8851573 0.9191038    0
bcart    0.7573835 0.7913482 0.8322741 0.8204199 0.8412971 0.8786557    0
multinom 0.6058264 0.6419423 0.6626772 0.6641844 0.6918232 0.7067222    0
dotplot(results)
```

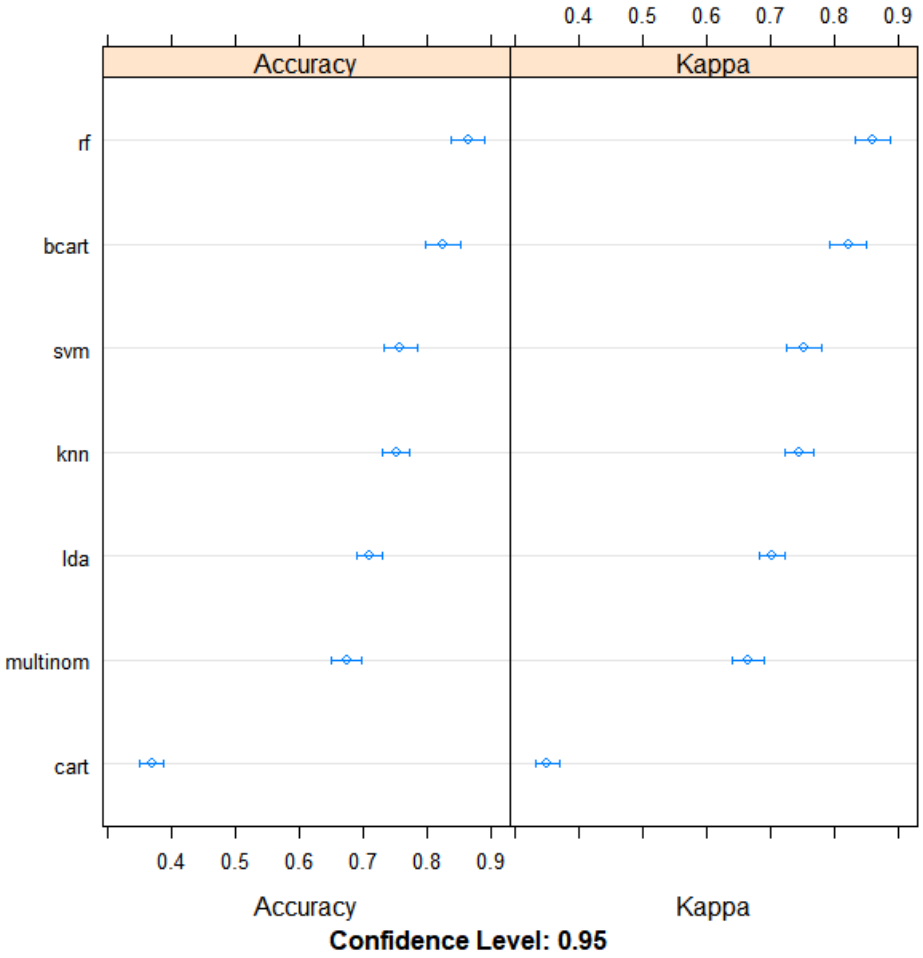

Figura 8. Gráfica de resultados de los entrenamientos de los algoritmos con Bluetooth y Scan1.

Se comprueba que el mejor método es el Random Forest y se opta por ver el modelo.

```
print(fit.rf)
Random Forest


1023 samples

   9 predictor

  33 classes: '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',
'13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '24', '25',
'26', '27', '28', '29', '30', '31', '32', '33'


No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 919, 920, 921, 919, 922, 920, ...

Resampling results across tuning parameters:
```

```
  mtry  Accuracy   Kappa
  2     0.8640650  0.8597959
  5     0.8620951  0.8577624
  9     0.8454928  0.8406409

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 2.
```

Una vez se tiene ese modelo, se debe comprobar su acierto mediante el uso de las medidas de las trayectorias Bluetooth, o datos de validación, para lo que se deben importar sus datos de forma similar a los de entrenamiento.

```
Ruta <- "ruta1_bt.csv"

validacion <- read.csv(Ruta, header = TRUE, sep=";")

validacion$ZONA <- as.factor(validacion$ZONA)
```

Debido a que la primera columna de los datos de validación no tiene todas las zonas que sí tienen los datos de entrenamiento, ya que solo se realiza la medida en las zonas del trayecto, se debe incluir el siguiente comando que añade esos factores que faltan para cada ruta.

```
#ruta1 BT

validacion$ZONA <- factor(validacion$ZONA, levels = c(levels(validacion$ZONA),"1","2","3","4","5","6","7","8","9","10","11","12","13","14","17","18","19","20","26","27","28","29","30","31","32","33"))

#ruta2 BT

#validacion$ZONA <- factor(validacion$ZONA, levels = c(levels(validacion$ZONA),"1","2","3","4","5","6","7","8","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","33"))
```

Por último, se estima la precisión del modelo y se obtiene la matriz de confusión.

```
predictions <- predict(fit.rf, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))
```

En la imagen siguiente se puede ver esta matriz de confusión. Se aprecia que se diferencian las zonas del trayecto, 15, 16, 21, 22, 23, 24 y 25, de las zonas por las que no ha pasado el usuario. Se puede observar también que, en el caso de errar en la clasificación, este fallo corresponde a identificar otra zona cercana.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 9. Matriz de confusión Trayecto 1 Bluetooth con Scan 1 y Random Forest.

Esta instrucción de la matriz de confusión devuelve también estadísticas generales y estadísticas por cada clase.

```
Overall Statistics

               Accuracy : 0.6667
                 95% CI : (0.3838, 0.8818)
    No Information Rate : 0.2
    P-Value [Acc > NIR] : 0.0001132

                  Kappa : 0.6114

 Mcnemar's Test P-Value : NA
Statistics by Class:

                     Class: 15 Class: 16 Class: 21 Class: 22 Class: 23 Class: 24 Class: 25 Class: 1 Class: 2
Sensitivity            0.50000   0.50000    0.6667   0.50000    1.0000   0.50000    1.0000       NA       NA
```

Specificity	0.84615	0.92308	1.0000	0.92308	0.9231	1.00000	1.0000	1	1
Pos Pred Value	0.33333	0.50000	1.0000	0.50000	0.6667	1.00000	1.0000	NA	NA
Neg Pred Value	0.91667	0.92308	0.9231	0.92308	1.0000	0.92857	1.0000	NA	NA
Prevalence	0.13333	0.13333	0.2000	0.13333	0.1333	0.13333	0.1333	0	0
Detection Rate	0.06667	0.06667	0.1333	0.06667	0.1333	0.06667	0.1333	0	0
Detection Prevalence	0.20000	0.13333	0.1333	0.13333	0.2000	0.06667	0.1333	0	0
Balanced Accuracy	0.67308	0.71154	0.8333	0.71154	0.9615	0.75000	1.0000	NA	NA

	Class: 3	Class: 4	Class: 5	Class: 6	Class: 7	Class: 8	Class: 9	Class: 10	Class: 11
Sensitivity	NA	NA	NA	NA	NA	NA	NA	NA	NA
Specificity	1	1	1	1	1	1	1	1	1
Pos Pred Value	NA	NA	NA	NA	NA	NA	NA	NA	NA
Neg Pred Value	NA	NA	NA	NA	NA	NA	NA	NA	NA
Prevalence	0	0	0	0	0	0	0	0	0
Detection Rate	0	0	0	0	0	0	0	0	0
Detection Prevalence	0	0	0	0	0	0	0	0	0
Balanced Accuracy	NA	NA	NA	NA	NA	NA	NA	NA	NA

	Class: 12	Class: 13	Class: 14	Class: 17	Class: 18	Class: 19	Class: 20	Class: 26	Class: 27
Sensitivity	NA	NA	NA	NA	NA	NA	NA	NA	NA
Specificity	1	1	1	1	1	1	1	1	1
Pos Pred Value	NA	NA	NA	NA	NA	NA	NA	NA	NA
Neg Pred Value	NA	NA	NA	NA	NA	NA	NA	NA	NA
Prevalence	0	0	0	0	0	0	0	0	0
Detection Rate	0	0	0	0	0	0	0	0	0
Detection Prevalence	0	0	0	0	0	0	0	0	0
Balanced Accuracy	NA	NA	NA	NA	NA	NA	NA	NA	NA

	Class: 28	Class: 29	Class: 30	Class: 31	Class: 32	Class: 33
Sensitivity	NA	NA	NA	NA	NA	NA
Specificity	1	1	1	1	1	1
Pos Pred Value	NA	NA	NA	NA	NA	NA
Neg Pred Value	NA	NA	NA	NA	NA	NA
Prevalence	0	0	0	0	0	0
Detection Rate	0	0	0	0	0	0
Detection Prevalence	0	0	0	0	0	0
Balanced Accuracy	NA	NA	NA	NA	NA	NA

6. Resultados

En este apartado se detallan los resultados obtenidos para las distintas rutas de Bluetooth, Wi-Fi y datos combinados.

6.1 Bluetooth

Como aparece en la siguiente tabla, el método con el mayor tiempo de entrenamiento es *svm*, seguido de *multinom*, y el menor tiempo es *lda*.

Escaneo	lda	CART(rpart)	kNN	svm	rf	bcart	multinom
Total Scan	1.093501 secs	1.174913 secs	1.751102 secs	3.112897 mins	1.475793 mins	9.923691 secs	2.406225 mins
Scan1	0.7186182 secs	0.8748009 secs	2.456062 secs	42.68206 secs	24.06981 secs	4.367335 secs	41.09269 secs
Scan2	0.8012102 secs	0.93556 secs	1.328014 secs	30.01388 secs	15.85331 secs	3.078752 secs	30.6497 secs
Scan3	0.887284 secs	1.020193 secs	1.616962 secs	29.97048 secs	15.70419 secs	3.266394 secs	30.20586 secs
Scan4	0.7342319 secs	0.8279262 secs	2.780599 secs	25.55535 secs	14.00151 secs	2.572978 secs	26.0922 secs
Scan4	0.8082628 secs	1.064094 secs	1.839517 secs	32.2552 secs	16.0331 secs	3.299018 secs	29.82948 secs

Tabla 4. Tiempos de ejecución de train con cada algoritmo con Bluetooth.

En la gráfica siguiente se pueden ver estos datos de la tabla superior representados de forma más visual, para apreciar mejor la diferencia de tiempos.

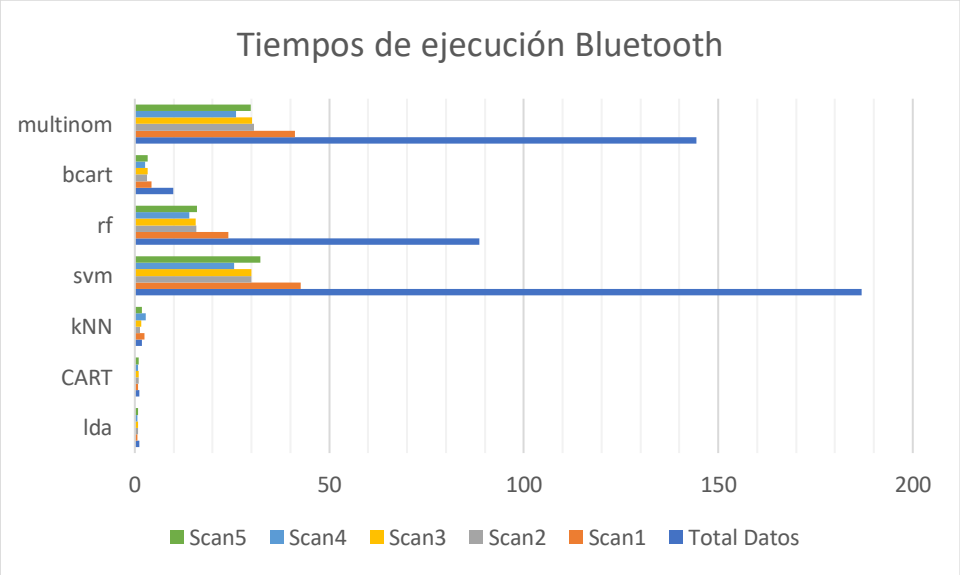

Figura 10. Gráfica tiempos de ejecución (s) Bluetooth.

Sin embargo, en la información obtenida durante la ejecución del programa, que puede verse también en forma de gráfica se observa que, aunque estos algoritmos tarden más, no son los que mejor solución dan, ya que el mejor es Random Forest, cuyo entrenamiento tarda 1.47 minutos.

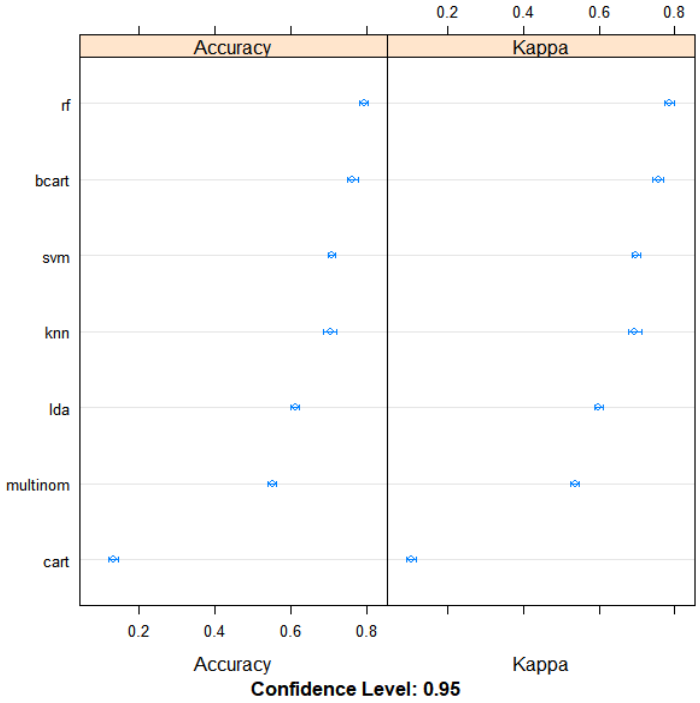

Figura 11. Gráfica de resultados de los entrenamientos de los algoritmos con Bluetooth y Total-Scan.

6.1.1 Trayecto 1 Bluetooth

En Bluetooth se ha optado por comprobar los trayectos utilizando los dos modelos que mejores resultados dan durante el entrenamiento, Random Forest y Bagged CART.

En la siguiente tabla se muestra el valor del término *Accuracy* para dichos modelos utilizando los distintos escaneos de las medidas. Estos valores pueden tomar valores entre 0 y 1, equivalentes a la representación en porcentajes 0% y 100%.

	Random Forest	Bagged CART
Total Scan	0.667	0.5333
Scan1	0.667	0.4
Scan2	0.5333	0.6
Scan3	0.4667	0.5333
Scan4	0.8	0.5333
Scan5	0.4667	0.5333

Tabla 5. Accuracy de modelos con Trayecto 1 Bluetooth.

Es destacable que aplicando Random Forest sobre los datos del cuarto escaneo se obtiene el valor más alto de *Accuracy*, 0.8, cuya matriz de confusión se muestra en la siguiente

imagen, y que, para el total de datos, Random Forest también da el mejor resultado con un 0.667.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 12. Matriz de confusión Trayecto 1 Bluetooth con Scan 4 y Random Forest.

6.1.2 Trayecto 2 Bluetooth

De la misma forma, cambiando los datos de validación, se ha evaluado el segundo trayecto con los mismos métodos.

	Random Forest	Bagged CART
Total Scan	0.5625	0.375
Scan1	0.3125	0.25
Scan2	0.1875	0.0625
Scan3	0.25	0.25
Scan4	0.3125	0.3125
Scan5	0.5625	0.5

Tabla 6. Accuracy de modelos con Trayecto 2 Bluetooth.

Comparando ambas tablas se puede apreciar que los resultados de este segundo trayecto son mucho peores, y que solo en tres casos se llega a una *Accuracy* superior al 0.5. Una

vez más, se cumple que Random Forest devuelve mejores resultados que Bagged CART, por lo que se considera que es el algoritmo más adecuado de entre todos los evaluados.

En la imagen se muestra la matriz de confusión de este trayecto utilizando como datos de entrenamiento todos los escaneos y con Random Forest ya que es una de las formas que da mejor resultado.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
31	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	0
32	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 13. Matriz de confusión Trayecto 2 Bluetooth con Total Scan y Random Forest

6.2 Wi-Fi

Las tablas de medidas Wi-Fi cuentan con más datos debido a que hay un mayor número de puntos de acceso Wi-Fi que balizas Bluetooth, por ello, como refleja la tabla de tiempo de ejecución del entrenamiento de cada modelo, se puede apreciar que esta duración es mayor que en el caso de Bluetooth, y que aquel modelo que tardaba un máximo de alrededor de 3 minutos, en este caso tarda más de 11 minutos.

	lda	CART (rpart)	knn	svm	rf	bcart
Total Scan	3.804791 secs	3.728282 secs	11.48982 secs	11.50601 mins	9.352072 mins	55.85469 secs
Scan1	1.629865 secs	1.723149 secs	2.002991 secs	1.491223 mins	1.451169 mins	11.8107 secs
Scan2	1.416973 secs	1.800398 secs	1.797774 secs	59.87701 secs	1.017416 mins	9.372524 secs
Scan3	1.446049 secs	1.502979 secs	1.82044 secs	1.068232 mins	1.027298 mins	10.0231 secs
Scan4	1.406815 secs	1.627236 secs	1.763921 secs	59.76503 secs	1.088941 mins	12.08188 secs
Scan5	1.778636 secs	1.825733 secs	1.880764 secs	1.153656 mins	1.154923 mins	11.40469 secs

Tabla 7. Tiempos de ejecución de train con cada algoritmo con Wi-Fi.

Los tiempos de la tabla superior se han representado también en una gráfica para distinguir la diferencia en el tiempo de entrenamiento de los modelos.

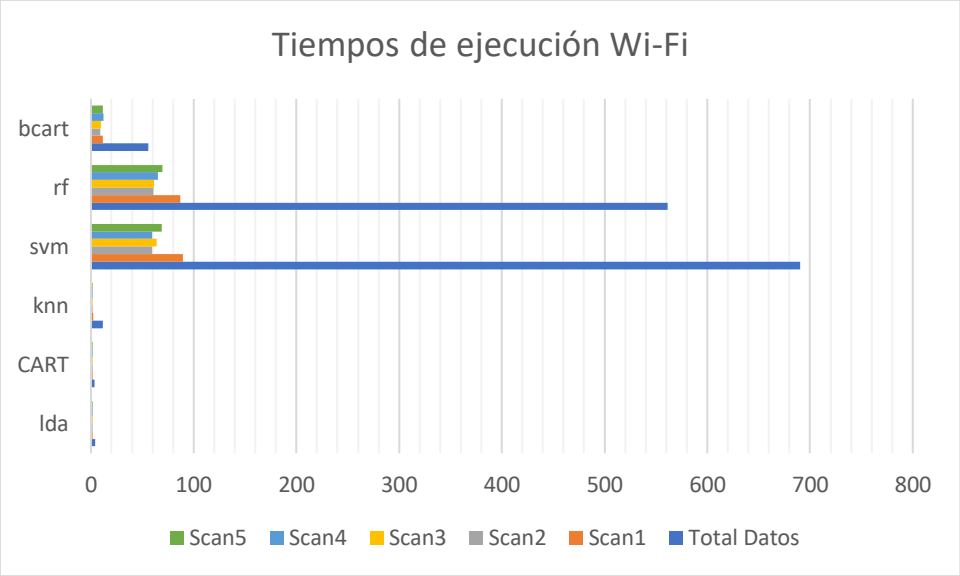

Figura 14. Gráfica tiempos de ejecución (s) Wi-Fi.

En cuanto a la *Accuracy* de los modelos, en la tabla y en la gráfica se puede ver que hay más modelos con valores muy altos, y el algoritmo CART(*rpart*) tiene una *Accuracy*

pésima, por lo que se ha optado por evaluar 5 de los 6 algoritmos en lugar de los 2 que se contemplaron en Bluetooth.

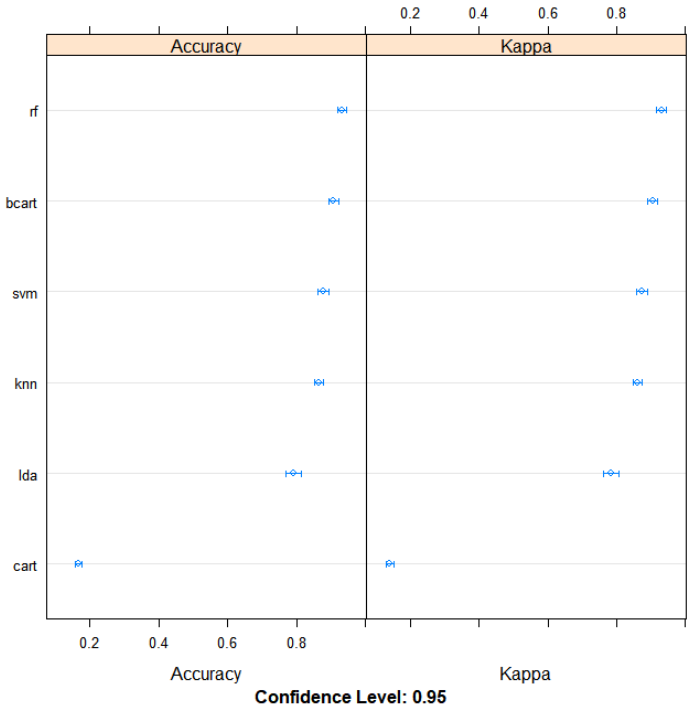

Figura 15. Gráfica de resultados de los entrenamientos de los algoritmos con Wi-Fi y Total-Scan.

6.2.1 Trayecto 1 Wi-Fi

Sin embargo, aunque la *Accuracy* del entrenamiento sea superior, los resultados de las predicciones que se obtienen son mucho peores, ya que ninguno de estos métodos alcanza el 0.5 de *Accuracy*, aunque el método que mejor resultados obtiene es Bagged CART.

	rf	Bagged CART	svm	lda	knn
Total Scan	0.2381	0.2857	0.1429	0.0952	0.1429
Scan1	0.0952	0.0952	0.0952	0.1429	0.0952
Scan2	0.2381	0.3333	0.0476	0.1429	0.0952
Scan3	0.2381	0.3333	0.1905	0.1905	0.1429
Scan4	0.1429	0.3333	0	0.1429	0.0952
Scan5	0.1429	0.0952	0	0.0952	0.0952

Tabla 8. Accuracy de modelos con Trayecto 1 Wi-Fi.

En la siguiente imagen se muestra la matriz de confusión de este trayecto mediante Bagged CART utilizando los datos de entrenamiento del tercer escaneo, ya que son estos datos los que han dado los valores más altos de *Accuracy* en los 5 modelos.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	2	1	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	2
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 16. Matriz de confusión Trayecto 1 Wi-Fi con Scan 3 y Bagged CART.

6.3 Bluetooth + Wi-Fi

En el caso del uso de los datos combinados de Bluetooth y Wi-Fi, se tiene un resultado parecido al de los datos Wi-Fi, con altos tiempos de ejecución, aunque ligeramente menores que en el caso de solo Wi-Fi.

	lda	CART (rpart)	knn	svm	rf	bcart
Total Scan	3.480667 secs	2.797547 secs	6.718923 secs	9.710402 mins	7.07379 mins	41.18315 secs
Scan1	1.474087 secs	1.598725 secs	1.875984 secs	1.475788 mins	1.917891 mins	15.02814 secs
Scan2	1.305486 secs	1.548611 secs	1.657049 secs	1.243496 mins	1.170904 mins	10.78474 secs
Scan3	1.401727 secs	1.659736 secs	1.707518 secs	1.275118 mins	1.178628 mins	10.90526 secs
Scan4	1.474793 secs	1.818564 secs	1.886806 secs	1.071366 mins	1.29547 mins	10.89708 secs
Scan5	1.445864 secs	1.572209 secs	1.590182 secs	1.070914 mins	1.295811 mins	12.13252 secs

Tabla 9. Tiempos de ejecución de train con cada algoritmo con Bluetooth+Wi-Fi.

Se vuelve a elaborar la gráfica con los valores de la tabla superior, con la intención de ver más clara la diferencia de tiempos entre los algoritmos.

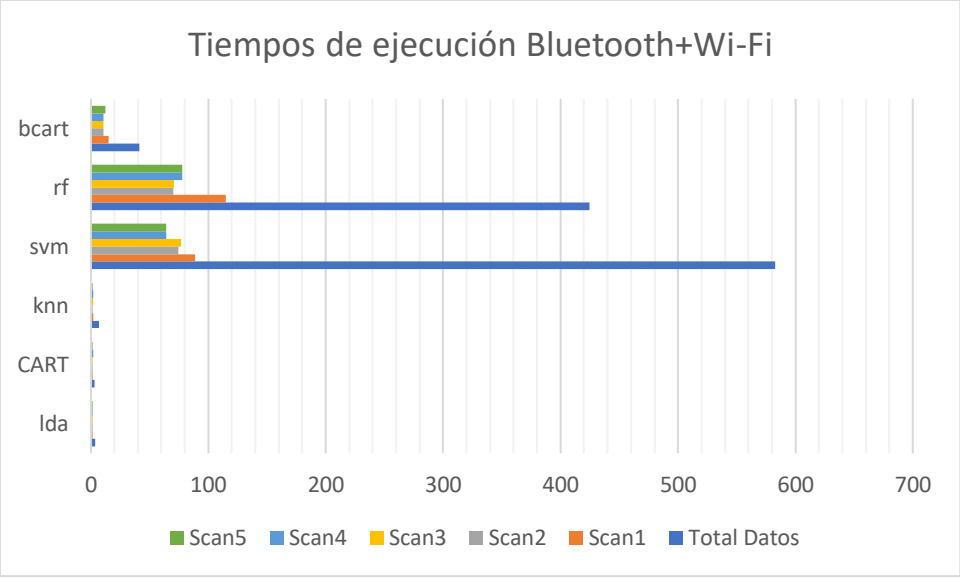

Figura 17. Gráfica tiempos de ejecución (s) Bluetooth+Wi-Fi.

La tabla de *Accuracy* es también parecida a la tabla de solo Wi-Fi, con una gran diferencia entre CART con el peor resultado y el resto de los algoritmos más parecidos, todos por encima del 0.8 de media, por lo que se han vuelto a evaluar 5 de los 6 algoritmos.

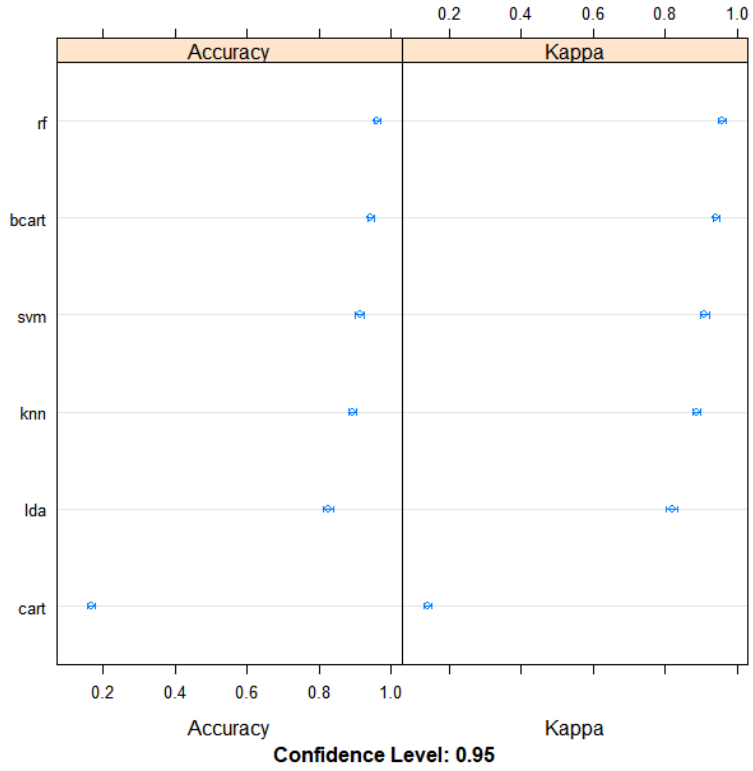

Figura 18. Gráfica de resultados de los entrenamientos de los algoritmos con Bluetooth+Wi-Fi y Total-Scan.

6.3.1 Trayecto 1 Bluetooth + Wi-Fi

De nuevo, al igual que en el caso anterior, la *Accuracy* de la predicción es mucho menor que en el caso de Bluetooth, y da un resultado parecido al de los datos Wi-Fi.

	rf	Bagged CART	svm	lda	knn
Total Datos	0.2	0.2	0.1333	0.1333	0.1333
Scan1	0.1333	0.1333	0.0667	0.0667	0.1333
Scan2	0.1333	0	0	0.1333	0
Scan3	0.2667	0.1333	0.0667	0.0667	0.1333
Scan4	0.1333	0.1333	0	0.0667	0.0667
Scan5	0.2	0.2667	0	0.1333	0.1333

Tabla 10. Accuracy de modelos con Trayecto 1 Bluetooth+Wi-Fi.

En esta prueba conjunta, los dos modelos de Machine Learning que mejor funcionan son Random Forest, que era el que daba mejor resultado en Bluetooth, y Bagged CART, que lograba el mejor resultado en Wi-Fi. Se representa la matriz de confusión de este trayecto utilizando Random Forest y el tercer escaneo, ya que corresponde a uno de los valores más altos en la tabla superior, el otro corresponde a Bagged CART con el quinto escaneo.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 19. Matriz de confusión Trayecto 1 Bluetooth + Wi-Fi con Scan 3 y Random Forest.

7. Conclusiones y líneas futuras

A continuación, se comentan las conclusiones a las que se ha llegado al realizar este trabajo y las posibles líneas de investigación que pueden continuar con el objetivo de este estudio.

En general, junto con los trabajos de Javier Gómez Ortiz y Andrei Pietro Sciddurlo Juaristi, este análisis de diversos modelos de Machine Learning demuestra que la utilización de técnicas de aprendizaje autónomo aplicadas al posicionamiento en interiores es una buena opción, mejor en unos casos que otros, y que si se continúa el estudio de estas técnicas y sus parámetros de ajuste es posible mejorar los resultados que se han obtenido hasta este momento y llegar a resolver el problema de la localización en interiores.

Debido a que este trabajo tiene como objetivo la mejora del sistema de posicionamiento en interiores mediante balizas Bluetooth Low Energy, se deben comparar los resultados de este trabajo con el trabajo anterior [2].

En dicho trabajo, se prueban los trayectos de Bluetooth y Wi-Fi y en lugar de probar una nueva ruta con las medidas Bluetooth+Wi-Fi, se prueban los trayectos anteriores sobre un total de 7 conjuntos de datos: 5 escaneos parciales, 1 escaneo total que suma todos los parciales y el conjunto de datos entre Bluetooth y Wi-Fi.

En este estudio, en cambio, se ha añadido un trayecto con las medidas de Bluetooth+Wi-Fi y no se han estudiado los otros trayectos utilizando el conjunto de datos de ambas medidas.

Por tanto, la comparación solo puede realizarse con aquellas pruebas que hayan utilizado los mismos tipos de datos.

En el análisis realizado de los distintos trayectos y datos de entrenamiento utilizado, el mejor resultado lo ha dado el Trayecto 1 con medidas Bluetooth, con el que se ha llegado en el cuarto escaneo al 80% de acierto, el máximo que se ha visto en este trabajo. Sin embargo, el extremo opuesto son los resultados obtenidos a través de las medidas Wi-Fi y Bluetooth+Wi-Fi que dan unos valores mucho más bajos.

Aun así, se puede apreciar una mejora de los resultados, ya que, sin tener en cuenta el caso del uso de datos de entrenamiento Bluetooth+Wi-Fi en otros trayectos, el mejor resultado del trabajo anterior es un 40%, mientras que aplicando Random Forest en este estudio se ha conseguido llegar a un 80%, como aparece en el apartado 6.1.1.

La aplicación de modelos de Machine Learning sobre estos datos, ha conseguido también que aquellos resultados peores en el trabajo anterior, que igual que en este estudio han sido los de medidas Wi-Fi, tengan valores ligeramente más altos, llegando al 33.33% frente al máximo anterior de 28.57%.

A la vista de estos resultados se puede deducir que se ha cumplido el objetivo de mejorar los resultados del sistema BLE mediante la aplicación de los modelos de aprendizaje autónomo.

En cuanto a las líneas futuras de estudio se puede expandir y modificar el sistema de múltiples formas:

- Buscar formas de mejorar los resultados de los modelos utilizados mediante la mejora de sus parámetros de ajuste, ya que en este trabajo se han utilizado los modelos con los parámetros por defecto y es probable que realizando ajustes en dichos valores se puedan obtener mejores resultados.
- Obtener nuevas medidas aumentando el número de balizas Bluetooth, para acercar el número de balizas al número de puntos de acceso Wi-Fi y probar cómo evoluciona el sistema con distintos datos.
- Al igual que el trabajo de Andrei Pietro Sciddurlo Juaristi se inspira en el de Javier Gómez Ortiz y cambia las zonas de estudio, una posibilidad de investigación es probar otra localización para analizar el posicionamiento.
- Otra mejora del sistema es mejorar el muestreo que se realiza de los trayectos en movimiento, ya que las medidas utilizadas para los datos de entrenamiento tienen un tiempo de muestreo más alto y puede que esta diferencia afecte a las predicciones que se realizan.

Pese a las múltiples mejoras que puedan realizarse para la investigación de este tipo de técnicas de posicionamiento, el principal camino a seguir es la aplicación práctica de este sistema. Por ejemplo, una aplicación móvil en el teléfono de una persona en el edificio, que, en colaboración con un sistema de registro de los datos de las balizas y de predicción de la situación del usuario, permita a éste acceder a información útil según su ubicación, en el caso de este edificio, los despachos que se encuentran en la zona.

8. Bibliografía

[1] Javier Gómez Ortiz, «Estudio de la aplicación de Machine Learning a técnicas de posicionamiento en interiores», *Study of the application of Machine Learning to indoor positioning techniques*, oct. 2018, Accedido: ago. 05, 2020. [En línea]. Disponible en: https://repositorio.unican.es/xmlui/handle/10902/14917.

[2] Andrei Pietro Sciddurlo Juaristi, «Sistema BLE para la zonificación y localización en interiores», *BLE system for indoors zonification and positioning*, oct. 2019, Accedido: ago. 05, 2020. [En línea]. Disponible en: https://repositorio.unican.es/xmlui/handle/10902/17079.

[3] «Using Global Localization to Improve Navigation», *Google AI Blog*. http://ai.googleblog.com/2019/02/using-global-localization-to-improve.html (accedido ago. 05, 2020).

[4] «iBeacon - Apple Developer». https://developer.apple.com/ibeacon/ (accedido ago. 05, 2020).

[5] A. Gozalo Madrazo, «Estudio y desarrollo de una solución efectiva para el problema del posicionamiento en interiores», Study and development of an effective solution for the problem of the positioning in interiors, oct. 2017, Accedido: ago. 05, 2020. [En línea]. Disponible en: https://repositorio.unican.es/xmlui/handle/10902/12065.

[6] U.S. Government, *Global Positioning System Standard Positioning Service Signal Specification.* 1995.

[7] «GIS Commons: A Free eText about Geographic Information Systems». https://giscommons.org/ (accedido ago. 05, 2020).

[8] «VIVE™ | Discover Virtual Reality Beyond Imagination». https://www.vive.com/eu/ (accedido ago. 05, 2020).

[9] «Wi-Fi Alliance». https://www.wi-fi.org/ (accedido ago. 05, 2020).

[10] «IEEE 802.11, The Working Group Setting the Standards for Wireless LANs». http://www.ieee802.org/11/ (accedido ago. 05, 2020).

[11] I. Wladawsky-Berger, «'Soft' Artificial Intelligence Is Suddenly Everywhere», *WSJ*, ene. 16, 2015. https://blogs.wsj.com/cio/2015/01/16/soft-artificial-intelligence-is-suddenly-everywhere/ (accedido ago. 05, 2020).

[12] Swamidason, Iwin Thanakumar Joseph & S Velliangiri, Dr & devadass, sorna., «Investigation of Deep Learning Methodologies in Intelligent Green Transportation System», *Journal of Green Engineering.*, vol. 10, pp. 931-950., mar. 2020.

[13] «Machine Learning by Stanford University», *Coursera*. https://www.coursera.org/learn/machine-learning (accedido ago. 05, 2020).

[14] «Las 9 tareas del Machine Learning según sus algoritmos», *Blogthinkbig.com*, nov. 13, 2017. https://blogthinkbig.com/tareas-machine-learning (accedido ago. 05, 2020).

[15] S. Balakrishnama, A. Ganapathiraju, «LINEAR DISCRIMINANT ANALYSIS - A BRIEF TUTORIAL». Institute for Signal and Information Processing Department of Electrical and Computer Engineering Mississippi State University, [En línea]. Disponible en: http://www.music.mcgill.ca/~ich/classes/mumt611_07/classifiers/lda_theory.pdf.

[16] R. A. Fisher, «THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS», *Annals of Eugenics*, vol. 7, n.º 2, pp. 179-188, sep. 1936, doi: 10.1111/j.1469-1809.1936.tb02137.x.

[17] C. R. Rao, «The Utilization of Multiple Measurements in Problems of Biological

Classification», *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 10, n.º 2, pp. 159-203, 1948.
[18] Breiman, L., Friedman, J.H., Olshen, R., and Stone, C.J, «CART Algorithm». ee Wadsworth & Brooks/Cole Advanced Books & Softw, 1984, [En línea]. Disponible en: ftp://ftp.boulder.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Algorithms/14.0/TREE-CART.pdf.
[19] Terry M. Therneau Elizabeth J. Atkinson Mayo Foundation, «An Introduction to Recursive Partitioning Using the RPART Routines», abr. 11, 2019. https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf.
[20] J. Brownlee, «Bagging and Random Forest Ensemble Algorithms for Machine Learning», *Machine Learning Mastery*, abr. 21, 2016. https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/ (accedido ago. 05, 2020).
[21] O. Harrison, «Machine Learning Basics with the K-Nearest Neighbors Algorithm», *Medium*, jul. 14, 2019. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (accedido ago. 05, 2020).
[22] «A Top Machine Learning Algorithm Explained: Support Vector Machines (SVMs)», *Velocity Business Solutions Limited*, feb. 27, 2020. https://www.vebuso.com/2020/02/a-top-machine-learning-algorithm-explained-support-vector-machines-svms/ (accedido ago. 05, 2020).
[23] Richard Williams, «Multinomial Logit Models - Overview». University of Notre Dame, feb. 07, 2019, [En línea]. Disponible en: https://www3.nd.edu/~rwilliam/stats3/Mlogit1.pdf.
[24] F. E. Harrell, *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. New York, NY: Springer New York, 2001.
[25] «The Comprehensive R Archive Network». https://cran.rediris.es/ (accedido ago. 05, 2020).
[26] «index | TIOBE - The Software Quality Company». https://www.tiobe.com/tiobe-index/ (accedido ago. 05, 2020).
[27] I. [at] rediris [dot] es, «RedIRIS - Sobre RedIRIS», jul. 27, 2020. https://www.rediris.es/rediris/ (accedido ago. 05, 2020).
[28] M. Kuhn et al., caret: *Classification and Regression Training*. 2020.
[29] B. Ripley, B. Venables, D. M. Bates, K. H. (partial port ca 1998), A. G. (partial port ca 1998), y D. Firth, *MASS: Support Functions and Datasets for Venables and Ripley's MASS*. 2020.
[30] T. Therneau, B. Atkinson, B. R. (producer of the initial R. port, y maintainer 1999-2017), *rpart: Recursive Partitioning and Regression Trees*. 2019.
[31] A. Peters, T. Hothorn, B. D. Ripley, T. Therneau, y B. Atkinson, *ipred: Improved Predictors*. 2019.
[32] H. Wickham, *plyr: Tools for Splitting, Applying and Combining Data*. 2020.
[33] D. Meyer *et al.*, *e1071: Misc Functions of the Department of Statistics, ProbabilityTheory Group (Formerly: E1071), TU Wien*. 2019.
[34] F. original by L. B. and A. Cutler y R. port by A. L. and M. Wiener, *randomForest: Breiman and Cutler's Random Forests for Classification andRegression*. 2018.
[35] A. Karatzoglou, A. Smola, K. Hornik, N. I. Australia (NICTA), M. A. Maniscalco, y C. H. Teo, *kernlab: Kernel-Based Machine Learning Lab*. 2019.

[36] B. Ripley y W. Venables, *nnet: Feed-Forward Neural Networks and Multinomial Log-Linear Models*. 2020.
[37] J. Brownlee, «Your First Machine Learning Project in R Step-By-Step», *Machine Learning Mastery*, feb. 02, 2016. https://machinelearningmastery.com/machine-learning-in-r-step-by-step/ (accedido ago. 05, 2020).

9. Anexos

9.1 Código Bluetooth

```
library(caret, quietly = TRUE)


#Cargar los datos

datos <- "Dataset_Alberto_Scan1_BT.csv"

dataset <- read.csv(datos, header = TRUE, sep=";")

dataset$ZONA <- as.factor(dataset$ZONA)


#Evaluar algoritmos

#10fold crossvalidation

control <- trainControl(method="cv", number=10)

metric <- "Accuracy"


#algoritmos distintos y ver el tiempo que tardan

# a) linear algorithms

set.seed(7)

start_time_lda <- Sys.time()

fit.lda <- train(ZONA~., data=dataset, method="lda", metric=metric,
trControl=control)

end_time_lda <- Sys.time()

end_time_lda - start_time_lda

# b) nonlinear algorithms

# CART

set.seed(7)

start_time_cart <- Sys.time()

fit.cart <- train(ZONA~., data=dataset, method="rpart", metric=metric,
trControl=control)

end_time_cart <- Sys.time()

end_time_cart - start_time_cart

# kNN

set.seed(7)
```

```
start_time_knn <- Sys.time()

fit.knn <- train(ZONA~., data=dataset, method="knn", metric=metric,
trControl=control)

end_time_knn <- Sys.time()

end_time_knn - start_time_knn

# c) advanced algorithms

# SVM

set.seed(7)

start_time_svm <- Sys.time()

fit.svm <- train(ZONA~., data=dataset, method="svmRadial",
metric=metric, trControl=control)

end_time_svm <- Sys.time()

end_time_svm - start_time_svm


# Random Forest

set.seed(7)

start_time_rf <- Sys.time()

fit.rf <- train(ZONA~., data=dataset, method="rf", metric=metric,
trControl=control)

end_time_rf <- Sys.time()

end_time_rf - start_time_rf

#Bagged CART (method = 'treebag') For classification and regression
using packages ipred, plyr and e1071 with no tuning parameters.

set.seed(7)

start_time_bcart <- Sys.time()

fit.bcart <- train(ZONA~., data=dataset, method="treebag",
metric=metric, trControl=control)

end_time_bcart <- Sys.time()

end_time_bcart - start_time_bcart

#Penalized Multinomial Regression (method = 'multinom')For
classification using package nnet with tuning parameters:

set.seed(7)

start_time_multinom <- Sys.time()
```

```
fit.multinom <- train(ZONA~., data=dataset, method="multinom",
metric=metric, trControl=control)

end_time_multinom <- Sys.time()

end_time_multinom - start_time_multinom


#ver los resultados de los distintos algoritmos

results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn,
svm=fit.svm, rf=fit.rf, bcart=fit.bcart, multinom=fit.multinom))

summary(results)


#comparar los modelos

dotplot(results)


#ver el modelo

print(fit.rf)


#importar los datos de validacion

Ruta <- "ruta1_bt.csv"

validacion <- read.csv(Ruta, header = TRUE, sep=";")

validacion$ZONA <- as.factor(validacion$ZONA)

#ruta1 BT

validacion$ZONA <- factor(validacion$ZONA, levels =
c(levels(validacion$ZONA),"1","2","3","4","5","6","7","8","9","10","11
","12","13","14","17","18","19","20","26","27","28","29","30","31","32
","33"))

#ruta2 BT

#validacion$ZONA <- factor(validacion$ZONA, levels =
c(levels(validacion$ZONA),"1","2","3","4","5","6","7","8","11","12","1
3","14","15","16","17","18","19","20","21","22","23","24","25","26","3
3"))

#estimar precision

predictions <- predict(fit.rf, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.bcart, validacion)
```

```
confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))
```

9.2 Código Wi-Fi

```
library(caret, quietly = TRUE)


#Cargar los datos

datos <- "Dataset_Alberto_Scan4_WIFI.csv"

dataset <- read.csv(datos, header = TRUE, sep=";")


dataset$ZONA <- as.factor(dataset$ZONA)

#Evaluar algoritmos

#10fold crossvalidation

control <- trainControl(method="cv", number=10)

metric <- "Accuracy"


#algoritmos distintos y ver el tiempo que tardan


# a) linear algorithms

set.seed(7)

start_time_lda <- Sys.time()

fit.lda <- train(ZONA~., data=dataset, method="lda", metric=metric,
trControl=control)

end_time_lda <- Sys.time()

end_time_lda - start_time_lda

# b) nonlinear algorithms

# CART

set.seed(7)

start_time_cart <- Sys.time()

fit.cart <- train(ZONA~., data=dataset, method="rpart", metric=metric,
trControl=control)

end_time_cart <- Sys.time()

end_time_cart - start_time_cart
```

```
# kNN

set.seed(7)

start_time_knn <- Sys.time()

fit.knn <- train(ZONA~., data=dataset, method="knn", metric=metric,
trControl=control)

end_time_knn <- Sys.time()

end_time_knn - start_time_knn

# c) advanced algorithms

# SVM

set.seed(7)

start_time_svm <- Sys.time()

fit.svm <- train(ZONA~., data=dataset, method="svmRadial",
metric=metric, trControl=control)

end_time_svm <- Sys.time()

end_time_svm - start_time_svm


# Random Forest

set.seed(7)

start_time_rf <- Sys.time()

fit.rf <- train(ZONA~., data=dataset, method="rf", metric=metric,
trControl=control)

end_time_rf <- Sys.time()

end_time_rf - start_time_rf

#Bagged CART (method = 'treebag') For classification and regression
using packages ipred, plyr and e1071 with no tuning parameters.

set.seed(7)

start_time_bcart <- Sys.time()

fit.bcart <- train(ZONA~., data=dataset, method="treebag",
metric=metric, trControl=control)

end_time_bcart <- Sys.time()

end_time_bcart - start_time_bcart


#ver los resultados de los distintos algoritmos
```

```
results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn,
svm=fit.svm, rf=fit.rf, bcart=fit.bcart))

summary(results)


#comparar los modelos

dotplot(results)


#ver el modelo

print(fit.rf)


#importar los datos de validacion

Ruta <- "ruta1_wifi.csv"

validacion <- read.csv(Ruta, header = TRUE, sep=";")

validacion$ZONA <- as.factor(validacion$ZONA)

#ruta1 WIFI

validacion$ZONA <- factor(validacion$ZONA, levels =
c(levels(validacion$ZONA),"1","2","3","4","5","6","7","8","9","10","11
","12","13","14","15","16","18","19","20","24","25","27","28"))

#estimar precision

predictions <- predict(fit.rf, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.bcart, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.svm, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.lda, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.knn, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))
```

9.3 Código Bluetooth + Wi-Fi

```
library(caret, quietly = TRUE)


#Cargar los datos

datos <- "Dataset_Alberto_Scan5_BTWIFI.csv"

dataset <- read.csv(datos, header = TRUE, sep=";")


dataset$ZONA <- as.factor(dataset$ZONA)

#Evaluar algoritmos

#10fold crossvalidation

control <- trainControl(method="cv", number=10)

metric <- "Accuracy"


#algoritmos distintos y ver el tiempo que tardan


# a) linear algorithms

set.seed(7)

start_time_lda <- Sys.time()

fit.lda <- train(ZONA~., data=dataset, method="lda", metric=metric,
trControl=control)

end_time_lda <- Sys.time()

end_time_lda - start_time_lda

# b) nonlinear algorithms

# CART

set.seed(7)

start_time_cart <- Sys.time()

fit.cart <- train(ZONA~., data=dataset, method="rpart", metric=metric,
trControl=control)

end_time_cart <- Sys.time()

end_time_cart - start_time_cart

# kNN

set.seed(7)

start_time_knn <- Sys.time()
```

```
fit.knn <- train(ZONA~., data=dataset, method="knn", metric=metric,
trControl=control)

end_time_knn <- Sys.time()

end_time_knn - start_time_knn

# c) advanced algorithms

# SVM

set.seed(7)

start_time_svm <- Sys.time()

fit.svm <- train(ZONA~., data=dataset, method="svmRadial",
metric=metric, trControl=control)

end_time_svm <- Sys.time()

end_time_svm - start_time_svm


# Random Forest

set.seed(7)

start_time_rf <- Sys.time()

fit.rf <- train(ZONA~., data=dataset, method="rf", metric=metric,
trControl=control)

end_time_rf <- Sys.time()

end_time_rf - start_time_rf

#Bagged CART (method = 'treebag') For classification and regression
using packages ipred, plyr and e1071 with no tuning parameters.

set.seed(7)

start_time_bcart <- Sys.time()

fit.bcart <- train(ZONA~., data=dataset, method="treebag",
metric=metric, trControl=control)

end_time_bcart <- Sys.time()

end_time_bcart - start_time_bcart


#ver los resultados de los distintos algoritmos

results <- resamples(list(lda=fit.lda, cart=fit.cart, knn=fit.knn,
svm=fit.svm, rf=fit.rf, bcart=fit.bcart))

summary(results)
```

```
#comparar los modelos

dotplot(results)


#ver el modelo

print(fit.rf)


#importar los datos de validacion

Ruta <- "rutacombinada_btwifi.csv"

validacion <- read.csv(Ruta, header = TRUE, sep=";")

validacion$ZONA <- as.factor(validacion$ZONA)

#ruta combinada BTWiFi

#validacion$ZONA <- factor(validacion$ZONA, levels =
c(levels(validacion$ZONA),"1","2","3","4","5","6","7","8","9","10","11
","12","13","14","15","16","18","19","20","24","25","27","28"))

#estimar precision

predictions <- predict(fit.rf, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.bcart, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.svm, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.lda, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))

predictions <- predict(fit.knn, validacion)

confusionMatrix(factor(predictions, levels = 1:33),
factor(validacion$ZONA, levels = 1:33))
```