Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Comparación de Algoritmos de Aprendizaje Automático para la Clasificación de Golpes y Niveles de Pádel: Ensamble de Clasificadores

Autor: David Gómez Vázquez
Tutor: Daniel Gutiérrez Reina

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Comparación de Algoritmos de Aprendizaje Automático para la Clasificación de Golpes y Niveles de Pádel: Ensamble de Clasificadores

Autor:
David Gómez Vázquez

Tutor:
Daniel Gutiérrez Reina
Profesor titular

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Proyecto Fin de Carrera: Comparación de Algoritmos de Aprendizaje Automático para la Clasificación de Golpes y Niveles de Pádel: Ensamble de Clasificadores

Autor: David Gómez Vázquez

Tutor: Daniel Gutiérrez Reina

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mi familia

A mis amigos

A mis maestros

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia por brindarme siempre apoyo en mis decisiones y ofrecerme sus sabios consejos. En especial, a mis padres por escucharme siempre que lo necesito, aunque simplemente sea para quejarme. También, agradecer a mis amigos y compañeros de clase por alegrarme estos años que hemos compartido juntos.

Por otro lado, querría a todos los deportistas que dedicaron su tiempo para que este proyecto haya sido capaz de realizarse, así como a los anteriores estudiantes que participaron en este proyecto. También, a Daniel por ofrecerme esta oportunidad y orientarme en el proceso.

David Gómez Vázquez

Sevilla, 2023

Resumen

Actualmente, la inclusión de la tecnología para la mejora en el ámbito deportivo es un hecho. Además, el uso de la inteligencia artificial para la resolución de los problemas está en pleno auge. En este trabajo combinaremos estos dos aspectos, utilizando la inteligencia artificial para impulsar una mejora en el rendimiento de los jugadores de pádel.

Así, en este trabajo haremos uso de algoritmos de aprendizaje automático, en particular, algoritmos de ensamble de clasificadores, para clasificar los distintos golpes de pádel y el niveles del jugador. Utilizaremos distintos algoritmos: perceptrón multicapa, árbol de decisión, K vecinos más próximos, máquina de vector soporte, clasificador por votación, Bagging, Random Forest, AdaBoost y Gradient Boosting.

Abstract

Currently, the incorporation of technology for enhancement in the sports domain is a reality. Furthermore, the utilization of artificial intelligence for problem-solving is on the rise. In this study, we will merge these two aspects, leveraging artificial intelligence to enhance the performance of paddle tennis players.

Thus, in this research, we will employ machine learning algorithms, specifically ensemble classifier algorithms, to categorize various paddle tennis shots and player skill levels. We will utilize diverse algorithms including multilayer perceptron, decision tree, K-nearest neighbors, support vector machine, voting classifier, Bagging, Random Forest, AdaBoost, and Gradient Boosting.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación

IA	Inteligencia artificial
KNN	K vecinos más próximos
SVM	Máquina de vector soporte
IMU	Unidad de medición inercial
CHAID	Detección automática de interacciones mediante chi-cuadrado
WPT	World Padel Tour
MLP	Perceptrón multicapa

1 INTRODUCCIÓN

El éxito en crear inteligencia artificial sería el mayor evento en la historia de la humanidad. Desafortunadamente, puede ser también el último, a menos que aprendamos a evitar los riesgos.

- Stephen Hawking -

El pádel es un deporte de pala que consiste en hacer botar la bola en el campo contrario. Este deporte es similar al tenis, con la particularidad de que la pista está rodeada por pared y rejas, pudiendo interactuar con ellas para conseguir el punto. Se juega por parejas, ganando la pareja que consiga lograr dos sets.

El pádel nace en México en 1969. La pista utilizada era de 20 metros de longitud por 10 metros de anchura, al igual que en la actualidad. Esta pista estaba rodeada por un muro de pared de 3 metros en los fondos y 2 metros en los laterales. Posteriormente, se incorporó rejas en las paredes laterales debido al calor que sufrían los jugadores. En 1974, el pádel se extiende a España, y un año más tarde, en 1975, el pádel llega a Argentina, donde tiene un gran éxito, expandiéndose a los países fronterizos. En 1991, se forma la Federación Internacional de Pádel en Madrid, entidad encargada de organizar torneos internacionales y de crear un reglamento oficial del deporte [1]. En la actualidad existen dos torneos de gran importancia, World Padel Tour y Premier Padel, donde compite la élite de este deporte.

Este deporte ha experimentado un importante auge en los últimos años. Según un estudio llevado a cabo conjuntamente por las universidades de Murcia y Granada, el número de jugadores de pádel en España ha pasado de 1.2 millones en 2010 a 4.2 millones en 2015 [2]. Este incremento de 3 millones de jugadores de pádel en solo 5 años ha situado al pádel entre los diez deportes más practicados en el país. Además, destaca la alta participación de personas mayores de 35 años que practican este deporte, lo que tiene efectos positivos en la salud y ayuda a reducir el riesgo de desarrollar enfermedades cardiovasculares [2].

Por otro lado, la inteligencia artificial se ha consolidado como una disciplina fundamental en la resolución de problemas en diversas áreas. La inteligencia artificial consiste en proporcionar a la máquina la capacidad intelectual de los humanos. Estas máquinas son capaces de captar la información del entorno con sensores, procesarlas y tomar decisiones. Este nombre fue concebido por McCarthy en una conferencia en el Darmouth College, donde se reunieron científicos de alta reputación para discutir la posibilidad de la creación de una máquina inteligente [3].

La inteligencia artificial (IA) ha experimentado un crecimiento exponencial en su aplicación en diversos campos, como la medicina, el marketing y la automatización. Su capacidad para automatizar tareas de manera ágil y eficiente ha revolucionado múltiples industrias, permitiendo realizar actividades las 24 horas del día que previamente requerían la intervención del ser humano. Uno de los sectores que ha sido especialmente impactado por estos avances es la medicina, donde la integración de la IA ha llevado a notables mejoras en la manipulación de muestras, la asistencia a pacientes, la creación de prótesis y precisión de los diagnósticos [4] [5].

En la actualidad, la inteligencia artificial no ha alcanzado el nivel de la inteligencia humana. Sin embargo, existen robots capaces de imitar ciertas capacidades humanas, como el habla, el movimiento, la comprensión del lenguaje y la percepción visual. Un ejemplo destacado es Sophia, un robot humanoide con la habilidad de mantener conversaciones con personas. Sophia puede expresar emociones a través de sus gestos faciales y ha

ganado popularidad por sus interacciones en entrevistas. Su reconocimiento llegó al punto de ser otorgada la ciudadanía en Arabia Saudita [6].

En particular, Machine Learning o "aprendizaje automático", una rama de la inteligencia artificial, proporciona a los ordenadores la capacidad de aprender sin la necesidad de la intervención humana. Esta técnica se basa en el desarrollo de algoritmos que se entrenan con datos para resolver diferentes problemas, analizando la información y las relaciones existentes entre los datos para crear modelos que permitan nuevos elementos [7]. Entre las aplicaciones más destacadas del aprendizaje automático se encuentra el reconocimiento de voz, el reconocimiento de imagen, la conducción autónoma, así como en el análisis de datos en distintos campos, como el financiero, el médico o el industrial [8].

¿Y si aplicamos la inteligencia artificial al pádel? Existen numerosos estudios científicos que investigan cómo mejorar el físico y la fuerza de las personas, realizando ejercicios para fortalecer ciertas partes del cuerpo o contabilizar la energía empleada en distintas actividades físicas. Además, la ciencia ha permitido mejorar las técnicas utilizadas en distintos deportes con el objetivo de aumentar el rendimiento de los jugadores. En el caso del pádel, se pueden analizar distintos aspectos del juego utilizando la inteligencia artificial. Por ejemplo, se puede investigar cuáles son los golpes que otorgan más puntos, la técnica adecuada para cada movimiento y que acciones pueden aportar una mayor ventaja competitiva. Con el uso de algoritmos de aprendizaje automático, se pueden analizar grandes cantidades de datos de partidos o entrenamientos y utilizarlo para mejorar el rendimiento y la habilidad de los jugadores.

En nuestro caso, utilizaremos diferentes algoritmos de aprendizaje para la clasificación de golpes de pádel. En concreto, desarrollaremos métodos de ensamble de algoritmos, donde se combinan las predicciones de varios clasificadores para entrenar un algoritmo y mejorar la predicción final. Aunque los algoritmos individuales pueden ser efectivos, los ensambles de clasificadores tienen la capacidad de mejorar la robustez y precisión de las predicciones, obteniendo modelos con mayor rendimiento. Al utilizar ensambles de clasificadores se puede aumentar la generalización del modelo, adaptándose a nuevos datos, y disminuye la posibilidad de overfitting. En este trabajo estudiaremos la comparación del rendimiento de los ensambles con los algoritmos individuales en el contexto del pádel.

Un ejemplo sobre la aplicación de algoritmos de ensamble se muestra en el estudio [9], donde se lleva a cabo una investigación sobre la clasificación de la calificación crediticia, que es una actividad crucial en la gestión financiera. Se entrenan cinco algoritmos de ensamble: Random Forest, AdaBoost, Extreme Gradient Boosting, Light Gradient Boosting Machine y Stacking. Además, se utilizan clasificadores bases que incluyen redes neuronales, regresión logística, árbol de decisión, SVM y el clasificador bayesiano. Los resultados muestran una mejora en el rendimiento al utilizar los métodos de ensamble, con la excepción del método AdaBoost. Destaca que el algoritmo Random Forest proporciona el mejor rendimiento.

En nuestro caso, emplearemos distintos tipos de clasificadores bases, como son el árbol de decisión, k vecinos más próximos (KNN), Máquinas de Vector Soporte (SVM) y el perceptrón multicapa (MLP). Para la combinación de los clasificadores bases, usaremos distintos algoritmos de ensamble de clasificadores, como son por el clasificador por votación, Bagging, Pasting, Random Forest, AdaBoost y Gradient Boosting.

Para ello, necesitaremos de un dispositivo capaz de recopilar datos sobre los golpes realizados para generar una base de datos adecuada para el entramiento de algoritmos. Este dispositivo ha sido diseñado específicamente para detectar las velocidades angulares y las aceleraciones lineales de la mano del jugador con la que realiza el golpe.

La creación del dispositivo capaz de captar los movimientos de los jugadores y la recopilación de las muestras para formar la base datos fueron llevadas a cabo previamente en los trabajos [10] [11]. En estos estudios, además de construir la base de datos, se evaluaron diversos algoritmos de aprendizaje para la clasificación de los tipos de golpes realizados, tanto en el dominio del tiempo como el dominio frecuencial. En nuestro caso, como ya hemos mencionado, utilizaremos algoritmos de ensamble de clasificadores en el dominio temporal, que no se han empleado previamente. Además, entrenaremos estos algoritmos para clasificar el nivel del jugador, un aspecto que no ha sido abordado previamente en la investigación.

1.1 Descripción del problema

El pádel, al ser de un deporte con poca historia y aún en desarrollo, carece de una cantidad significativa de investigaciones. Esta escasez de estudios limita tanto el avance en la práctica profesional del pádel como la mejora de las metodologías de enseñanza por los entrenadores.

Para analizar el movimiento de los jugadores, es necesario una base de datos con golpes de jugadores y de un algoritmo capaz de clasificar el movimiento realizado. De este modo, podemos estudiar las tácticas óptimas y movimientos idóneos en el pádel. Esta investigación proporcionará una base objetiva para evaluar y comparar el desempeño de distintos jugadores, contribuyendo así a la mejora en este deporte.

1.2 Objetivos

El objetivo principal de este trabajo será la comparación de distintos algoritmos de aprendizaje automático para la clasificación de golpes u niveles de pádel. Se entrenarán los algoritmos para la distinción del tipo de golpe utilizado y el nivel del jugador que lo ha realizado. De esta forma, se crearán modelos matemáticos capaces de clasificar estas categorías facilitando la obtención de datos para futuras investigaciones.

Se mostrarán los resultados obtenidos para cada clasificador, comparando los distintos métodos para encontrar el más efectivo. En concreto, se buscará una mejora del rendimiento al utilizar los métodos de ensamble de clasificadores. Se empleará distintos métodos de ensamble de clasificadores y se estudiará su comportamiento para la clasificación de los golpes de pádel realizados. Así, los objetivos a lograr son:

- Comparación de los algoritmos de ensamble de clasificadores con los clasificadores bases utilizados en el ensamble, tanto para la clasificación del tipo de golpe como para la clasificación del nivel del jugador.
- Comparación de los distintos algoritmos de ensamble de clasificadores para obtener el mejor resultado, tanto para la clasificación del tipo de golpe como para la clasificación del nivel del jugador.
- Comparación del tiempo utilizado para el entrenamiento de los distintos algoritmos de aprendizaje automático, tanto para la clasificación del tipo de golpe como para la clasificación del nivel del jugador.

2 ESTADO DEL ARTE

En los últimos años, ha surgido un creciente interés en la comunidad científica y deportiva por investigar el rendimiento de los jugadores de pádel y la influencia de las distintas acciones en el juego. El estudio de los movimientos realizados por los jugadores supone una posibilidad para mejorar la técnica de los deportistas y ofrecer una mejor compresión de las estrategias que aportan una mayor ventaja en el juego. Estas investigaciones han empleado conjuntos de datos recopilados a través de diversas fuentes, como sensores inerciales, sistemas de seguimiento de movimiento y cámaras de alta velocidad.

En el ámbito de la inteligencia artificial, se puede notar un incremento de uso de algoritmos de aprendizaje automático para la clasificación de los distintos golpes y su impacto en la obtención de puntos. La correcta identificación de estos golpes puede proporcionar información valiosa para los entrenadores, permitiendo un análisis más detallado de las fortalezas y debilidades de los jugadores.

En este apartado del estado de arte, se examinarán detalladamente las contribuciones más relevantes en este campo.

En primer lugar, se analizará estudios previos realizados en el tenis. Aunque existe algunas diferencias entre ambos deportes, comparten ciertos aspectos en cuanto al movimiento y golpes ejecutados por los jugadores. Además, al ser un deporte con mayor popularidad e historia que el pádel, se pueden encontrar un mayor número de estudios desarrollados.

En [12], Thomas Perri, Marchar Reid, Alistair Murphy, Kieran Howle y Rob Duffield presentan un estudio sobre la detección de golpes y movimientos de tenis. En este estudio, se utilizó un acelerómetro, giroscopio y magnetómetro situado en la columna cervical para tomar las medidas del golpe y movimiento de 8 jugadores jóvenes, competidores de torneos internacionales. Estos sensores permiten captar las velocidades y aceleraciones del jugador. Se utilizaron algoritmos de aprendizaje automático supervisados como Random Forest para clasificar el tipo de golpe realizado por el jugador, obteniendo una precisión de 98% para clasificar saques y de 94% para golpes de derecha y revés. También, dentro de los golpes de derecha y revés se realizó una clasificación del movimiento realizado, siendo alguno de estos estos golpes drive, remate y voleas.

El estudio [13] presenta una clasificación de tres golpes de tenis, derecha, revés y saque, utilizando las medidas captadas por una IMU colocada en la muñeca del jugador. Se entrenó tres clasificadores Naive Bayes para cada golpe, empleando de forma separada los datos recogidos del acelerómetro, giroscopio y magnetómetro para cada clasificador. Además, se realizó una comparación entre utilizar los golpes de 4 a 7 jugadores. Cabe destacar que la mayor precisión obtenida para los distintos sensores fue el acelerómetro, alcanzando una precisión 80%, seguida del magnetómetro y posteriormente el giroscopio. En cuanto a la diferencia de golpes utilizados en el entrenamiento de los clasificadores, los resultados muestran que no existe una gran discrepancia en la precisión obtenida. Finalmente, la combinación de los tres sensores logró alcanzar un 90% de aciertos, lo cual supone una mejora del 10% con respecto a la precisión obtenida con el acelerómetro.

Un estudio sobre el diseño de un dispositivo capaz de captar información de los movimientos de jugadores de tenis y su implementación para clasificar distintas actividades realizadas durante el juego se llevó a cabo en [14]. En este estudio, se prepara una base de datos propia, utilizando dos sensores para la recogida de datos. El dispositivo utilizado es el SensorTag CC2650STK de Texas Instruments, situados en la muñeca y cintura del jugador. Se captaron señales a una frecuencia de muestreo de 20 Hz. Este dispositivo contiene un giroscopio y un acelerómetro para captar las aceleraciones y velocidades en tres ejes, al igual que en los casos anteriores. Para la detección y clasificación de golpes, se utilizó el espectrograma de la señal, tomando 40 muestras para cada golpe, lo que equivale a 2 segundos por movimiento. Por último, se entrenó un algoritmo de aprendizaje semi-supervisado para la clasificación, empleando Convolutional Autoencoder en el aprendizaje no supervisado y perceptrón multicapa en el aprendizaje supervisado. Se desarrollaron tres modelos de clasificación: uno para categorizar si se trata de un golpe de tenis (derecha, revés, volea o globo) o una

actividad normal (correr, saltar, caminar…), en caso de actividad normal, para detectar que actividad normal se ha realizado, y en caso de golpe de tenis, clasificar que golpe se ha realizado. Los resultados obtenidos, utilizando como datos de testeo un nuevo jugador, fueron un 99.52% de precisión para la clasificación de golpe de tenis o actividad normal, un 83.4% para la clasificación de las actividades normales y un 96.51% para la clasificación de golpes de tenis.

En cuanto al pádel, se pueden destacar los siguientes estudios:

Bernardino Javier Sánchez-Alcaraz, Javier Courel-Ibáñez, Diego Muñoz, Pablo Infantes-Córdoba, Franco Sáenz y Alejandro Sánchez-Pay [15] realizaron un estudio sobre la distribución de las acciones ofensivas y su influencia en el resultado del partido. Se analizó 2054 movimientos ofensivos correspondientes a cuatro finales masculinas del circuito profesional World Padel Tour a través de videos de los partidos. Las acciones ofensivas estudiadas fueron la volea de derecha, volea de revés, remate y bandeja, concluyendo que la volea es la acción más usada, seguido de la bandeja y finalmente el remate. Los ganadores realizaron un mayor número de golpes ofensivos, siendo la mayoría de los puntos (más del 80%) resueltos utilizando menos de 3 acciones de ataques.

En [16], se realizó un análisis CHAID para clasificar el tipo de golpe de acuerdo con la posición del jugador, la zona de juego, lateralidad del golpeo, altura de golpeo y eficacia. Se analizaron un total de 1963 golpes de 3 finales del campeonato World Padel Tour de 2014. Este estudio demuestra una gran influencia de la ubicación espacial del jugador en la clasificación de golpes. La zona de la red se caracteriza por su eficacia, siendo el remate y la bandeja los golpeos que más puntos ganadores lograron.

El estudio [17] muestra la importancia de mantener la red durante un partido profesional, tanto para el pádel masculino como femenino. Los datos fueron tomados de 12 competiciones profesionales de este deporte, que se obtuvieron a partir de grabaciones de los partidos disputados. Para la investigación, la pista se divide en dos zonas: la zona de la red y la zona de fondo. Los resultados nos muestran un mayor porcentaje de cambio de posiciones en el juego femenino, donde el 50.7% de los puntos la red permanece ocupada sin cambios por la pareja al saque. Por otro lado, la pareja al saque se adueña de la red sin cambios en el 65.9% de los puntos disputados en los partidos masculinos. También, se concluye que la posición de la red ofrece más oportunidades de ganar el punto, mientras que en la zona de fondo se producen más errores.

En [10], Guillermo Cartes llevó a cabo un análisis del uso de distintos algoritmos de aprendizaje automático para clasificar golpes de pádel, además de diseñar el dispositivo utilizado para la toma de muestras. El dispositivo utilizado fue la IMU LSM9DS1 incorporada al brazo del jugador, para captar las aceleraciones y velocidades del jugador al golpear la bola. En cuanto a los algoritmos de aprendizaje utilizados, se comparó los resultados del entrenamiento de redes densamente conectadas, redes convolucionales 1D, árboles de decisión, k vecinos más cercanos y máquinas de vector soporte. Se obtuvo un mejor porcentaje de aciertos con el uso de redes convolucionales 1D, alcanzando el máximo de 93.35% de aciertos, seguido de la red neuronal densamente conectada y la máquina de vector soporte, con 92.06% y 91.85% respectivamente.

El estudio anterior fue continuado por Claudia Martínez [11], donde se añade golpes de jugadores zurdos al conjunto de datos y se transforman al dominio de la frecuencia. Para la clasificación de los golpes, se utilizaron los mismos algoritmos de aprendizaje. Podemos apreciar que al añadir los movimientos de jugadores zurdos se disminuye levemente el porcentaje de aciertos en el dominio temporal, obteniendo como mejor resultado un 91.278% de aciertos. También, se observa una disminución de precisión en la clasificación de los golpes al transformar los datos al dominio de la frecuencia, llegando a disminuir un 20% de aciertos en algunos clasificadores. En este dominio destaca el entrenamiento de máquinas de vector soporte, con un resultado de 80.76% de precisión incluyendo jugadores zurdos y un 82,1% excluyéndolos.

3 SISTEMA DE RECOPILACIÓN DE DATOS

La recopilación de datos, al igual que el dispositivo diseñado y desarrollado para obtener las velocidades y aceleraciones de los movimientos fue realizado en proyectos previos por Guillermo Cartes Domínguez y Claudia Martínez Valerio [10] [11]. En este apartado, debido su importancia, se expondrá la información destacada sobre el dispositivo utilizado y la forma en que está caracterizada los datos.

3.1 Componentes hardware y software

Existe una gran variedad de golpes en el pádel, como la bandeja, el remate, el saque, cada uno caracterizado por un movimiento específico del cuerpo y brazo. En concreto, nos centraremos en el movimiento de la mano a la hora de realizar los golpes, por lo que será necesario un instrumento que capte las velocidades angulares y aceleraciones lineales de la mano. Se utilizan dos sensores diferentes para tomar las medidas, un giroscopio para obtener las velocidades angulares y un acelerómetro para las aceleraciones lineales. A continuación, se explicará de forma breve los componentes utilizados.

- **Raspberry pi** es una placa de microordenador de bajo coste y pequeñas dimensiones, donde recogeremos los datos de los sensores [18].
- **IMU** es un dispositivo que utiliza giroscopios y acelerómetros para medir las velocidades, orientación y aceleraciones que éste experimenta [19].
- **Alimentación** para la Raspberry pi, conectada por USB a 5V y 3A.
- **Soporte** para mantener los componentes en la muñeca para que se ajuste cómodamente al jugador para no causar molestias o restricciones en el movimiento del jugador.

Figura 1. Dispositivo de toma de medidas. Disponible en [10].

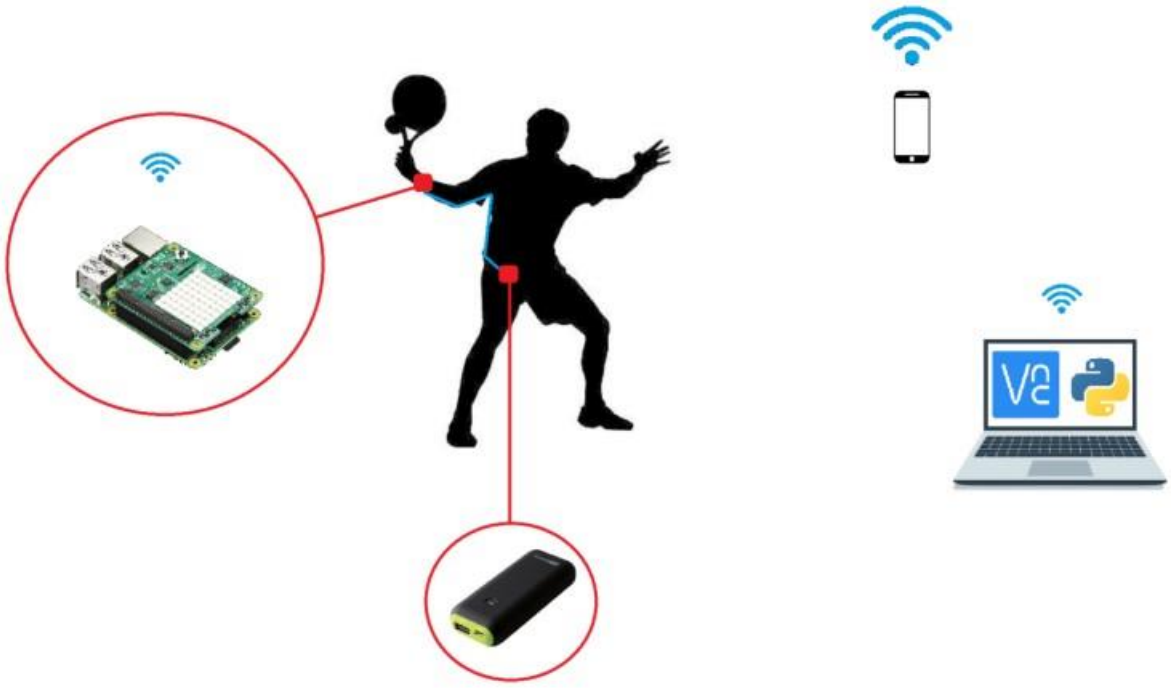

Figura 2. Sistema de recopilación de datos. Disponible en [10].

3.2 Disposición del conjunto de datos

La base de datos, además de las medidas tomadas de los sensores a través de la Raspberry pi, deberá contener las características del movimiento y del jugador. Esta información es importante para tener una base completa para futuros estudios. Así, el conjunto de datos presenta la siguiente estructura:

Tabla 1. Disposición del conjunto de datos.

Ax0… Ax39	**Ay0… Ay39**	**Az0… Az39**	**Vx0… Vx39**	**Vy0… Vy39**	**Vz0… Vz39**	**Tipo de golpe**	**Número de golpe**	**Tiempo de golpe**	**Sexo**	**Nivel**	**Mano**	**Revés**	**Altura**	**Edad**	**ID**
Golpe 1															
⋮															
Golpe N															

A continuación, se explicará los elementos de la tabla que necesitan una aclaración de la forma en que se clasifican:

Las 240 primeras columnas se corresponden a las 40 muestras de cada grado de libertad, teniendo en total 6 grados de libertad, que se corresponden a velocidades angulares y aceleraciones lineales de la mano. Se utilizan 40 muestras debido a que los sensores toman medidas de cada golpe a 20 Hz durante 2 segundos.

Los tipos de golpes que se consideran en la clasificación son 13, coincidiendo la numeración del listado con el número que lo representa en la base de datos:

0. Fondo de derecha: consiste en golpear la bola una vez haya botado desde el lado del brazo hábil del jugador a la altura de la cintura. Normalmente el jugador estará situado cerca de la línea de saque.
1. Fondo de revés: es similar al fondo de derecha, con la diferencia de golpear la bola desde el lado contrario del brazo hábil.
2. Derecha con pared: el golpe se realiza después de que la bola haya botado y posteriormente toque la pared, golpeando la bola desde el lado del brazo hábil del jugador.
3. Revés con pared: es similar al golpe anterior, golpeando la bola desde el lado contrario.
4. Globo de derecha: es un golpe fundamental en el pádel, ya que un globo bien realizado permite al jugador situarse cerca de la red, donde se realizan la mayoría de los golpes decisivos. Este golpe se utiliza cuando el rival está situado cerca de la red, haciendo pasar la bola por encima del jugador rival sin que este pueda alcanzarla, golpeando desde el lado del brazo hábil del jugador.
5. Globo de revés: parecido al globo de derecha, golpeando la bola desde el lado contrario del brazo hábil del jugador.
6. Globo de derecha con pared: consiste en realizar el globo de derecha después de que la bola haga contacto con la pared.
7. Globo de revés con pared: se realiza el mismo movimiento que el globo de revés, salvo que la bola deberá tocar primero la pared antes de realizar el movimiento.
8. Volea de derecha: este golpe se realiza cuando el jugador está situado cerca de la red, golpeando la bola desde el lado del brazo hábil del jugador antes de que la bola toque el suelo.
9. Volea de revés: este golpe se realiza cuando el jugador está situado cerca de la red, golpeando la bola desde el lado contrario del brazo hábil del jugador antes de que la bola toque el suelo.
10. Bandeja: es un golpe defensivo que se realiza cuando el jugador está situado cerca de la red y la bola viene alta, golpeando la bola a la altura de los hombros antes de que bote. Este movimiento permite al jugador recuperar la red de forma rápida y se realiza normalmente desde el lado hábil del jugador.
11. Remate: es un golpe que busca obtener el punto, golpeando la bola por encima de la cabeza, imprimiendo a la bola la mayor fuerza posible, de forma que el jugador contrario no sea capaz de alcanzarla.
12. Saque: Es el movimiento que da inicio al juego. El jugador que saca debe situarse detrás de la línea de saque y golpear la bola después de hacerla botar, sin que sobrepase su cintura, para meterla en el recuadro correspondiente del campo rival.

Es importante aclarar que estos no son todos los golpes que existen en el pádel, pero si se pueden considerar los golpes básicos. Existen otros golpes como la víbora, la contrapared, la chiquita o la dejada.

El nivel del jugador se clasificará en 5 categorías dependiendo de la trayectoria del jugador:

1. Iniciación: el deportista no está federado y lleva menos de 1 año jugando regularmente.
2. Amateur: el deportista no está federado y lleva más de 1 año jugando regularmente.
3. Intermedio: el deportista juega competiciones federadas en su comunidad y es categorizado como jugador de nivel intermedio por dos entrenadores nacionales.
4. Avanzado: el deportista juega competiciones federadas en su comunidad y es categorizado como

jugador de nivel alto por dos entrenadores nacionales.

5. Profesional: el deportista es internacional, juega en el World Padel Tour.

El elemento de la tabla “mano” sirve para identificar si el jugador que realiza el golpe es diestro o zurdo. El elemento “revés” hace referencia a si el deportista golpea la bola del revés con una o dos manos, tomando valores 1 o dos respectivamente. Finalmente, la columna ID se utiliza para identificar al jugador, asignando a cada jugador un número.

La base de datos cuenta con un número total de 4002 golpes. En estos datos se incluye personas de edades comprendidas entre los 18 y 58 años. Se han incluido muestras tanto de hombres como de mujeres, que comprenden desde los niveles iniciación hasta avanzado, incorporando además una jugadora profesional competidora del World Padel Tour. También, se abarca tanto jugadores diestros como zurdos, aportando una mayor variedad en el conjunto de datos.

En la Figura 3 podemos observar la diferencia de las muestras tomadas por el giroscopio y acelerómetro entre un golpe de derecha y un golpe de revés. En este caso concreto, se puede apreciar una clara distinción en los datos captados por el giroscopio en los tres ejes. Esta diferencia entre los distintos golpes permitirá a los algoritmos de aprendizaje automático clasificar las etiquetas de cada golpe.

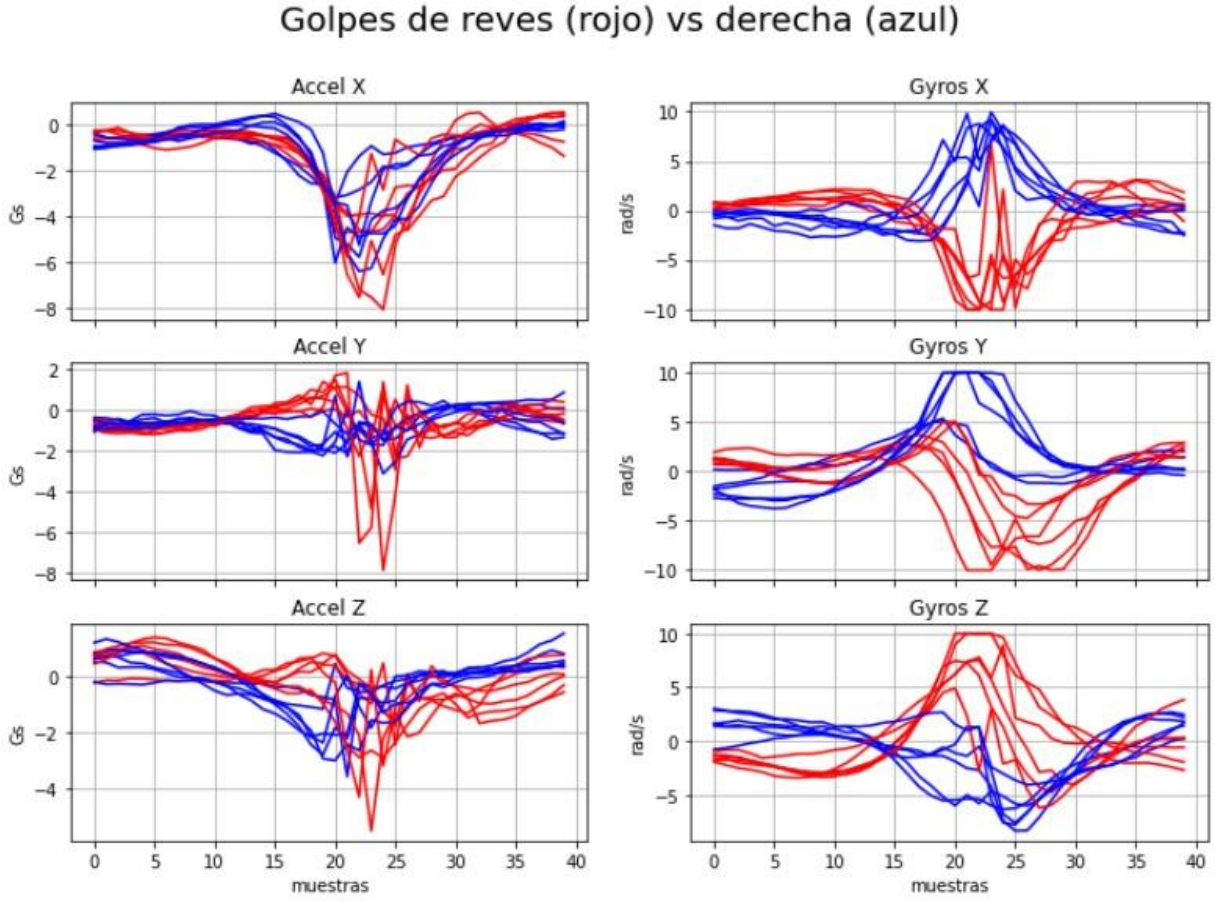

Figura 3. Comparación golpes de revés y derecha.

En la Figura 4 y Figura 5, se disponen los golpes de saques realizados por una jugadora profesional y una jugadora amateur. En la jugadora profesional se puede ver que los movimientos realizados son muy similares, superponiéndose unas muestras sobre otras. En cambio, los valores de la gráfica de la jugadora amateur son más dispersos, no se aprecia una tendencia clara. Estas diferencias nos permitirán clasificar los jugadores por su nivel.

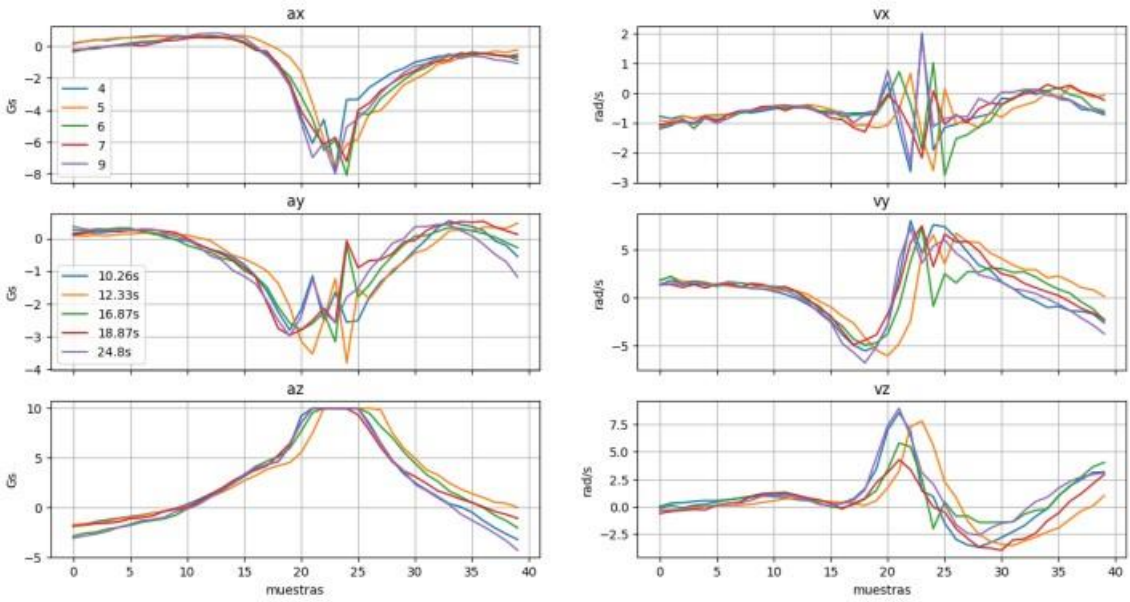

Figura 4. Muestras de saque de jugadora profesional.

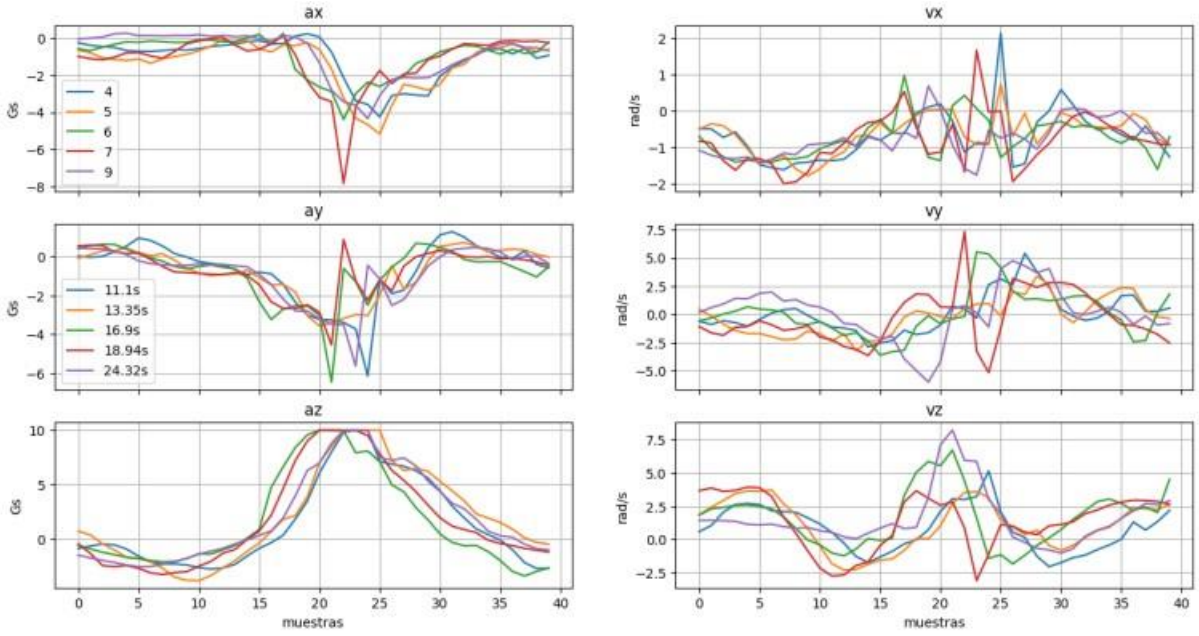

Figura 5. Muestras de saque de jugadora amateur.

4 METODOLOGÍA

En este apartado, introduciremos los conceptos necesarios para entender que es el aprendizaje automático, así como el funcionamiento de los algoritmos de aprendizaje utilizados en el proyecto y los factores que hay que tener en cuenta a la hora de crear los distintos algoritmos, ya que cada algoritmo utilizará distintos parámetros que variará su resultado.

4.1 Inteligencia artificial, Machine Learning y Deep Learning

La inteligencia artificial es la capacidad de las máquinas de realizar tareas que normalmente requerirían de la inteligencia humana, como el razonamiento, el aprendizaje, la creatividad, la percepción y muchas otras habilidades intelectuales. De esta manera, la inteligencia artificial permite a las máquinas captar información del entorno, procesarla y actuar en función de estos datos, con el objetivo de tomar decisiones o realizar tareas específicas. Los avances en la informática y la disponibilidad de enormes cantidades de datos han despertado un gran interés en la inteligencia artificial, lo que ha llevado a grandes avances en este campo en los últimos años [20].

El Machine Learning es una subrama de la inteligencia artificial que se centra en desarrollar técnicas que permiten identificar patrones a partir de datos, siendo el puente entre el Big Data y la inteligencia artificial. Los modelos de Machine Learning entrenan algoritmos utilizando datos reales para aprender a realizar tareas específicas, como la clasificación o predicción [21].

Finalmente, el Deep Learning o aprendizaje profundo es una rama del Machine Learning que se basa en redes neuronales artificiales, que imitan el funcionamiento del cerebro humano. Las redes neuronales profundas están compuestas por múltiples capas de neuronas, lo que les permite aprender patrones complejos en grandes conjuntos de datos. En particular, el Deep Learning ha mostrado un gran éxito en el reconocimiento de imágenes y voz [21].

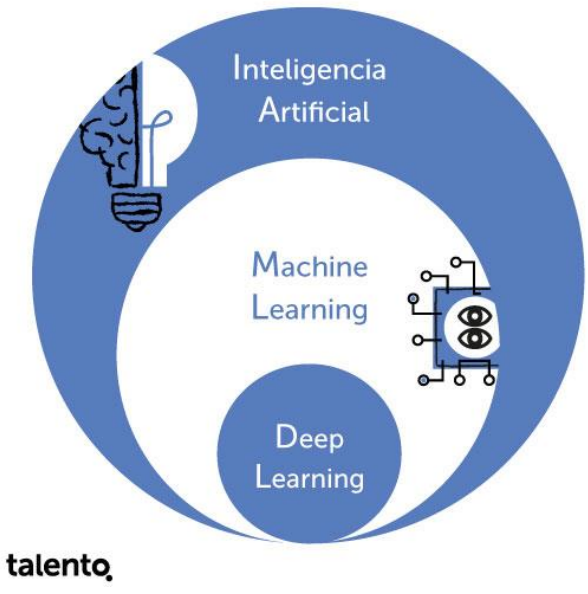

Figura 6. Inteligencia artificial, Machine Learning y Deep Learning. Disponible en [22].

4.2 Plataforma de desarrollo

TensorFlow es la plataforma de código abierto utilizada para la creación de modelos de aprendizaje automático. Esta plataforma permite realizar distintas tareas, como la preparación de los datos, la creación y la implementación de modelos [23]. En nuestro caso se ha instalado la versión 2.10. Keras es la API de alto nivel de TensorFlow para crear y entrenar de forma rápida algoritmos de Deep Learning [24] [25]. La versión utilizada es la 2.10.

Por otro lado, Scikit-learn es una librería de código abierto para Machine Learning en Python. Esta librería nos permitirá entrenar y evaluar modelos de forma rápida y sencilla. El lenguaje de programación utilizado para el desarrollo de este trabajo ha sido Python. En nuestro caso, hemos utilizado la versión de Python 3.9 y la versión de Scikit-learn 1.2 [26]. Finalmente, la plataforma empleada para el desarrollo del código ha sido Spyder 5.4.

4.3 Algoritmos de aprendizaje automáticos

Existen muchos tipos de algoritmos de aprendizaje automático, pudiéndose clasificar de la siguiente forma:

- Aprendizaje supervisado: en el aprendizaje supervisado, los datos con los que se entrena los algoritmos contienen la solución, denominada etiqueta. Esta etiqueta hace referencia a la categoría que pertenece cada dato. El algoritmo desarrolla un modelo para aproximar la relación entre las características de entrada y las etiquetas, comparando la solución de los datos con el resultado obtenido por el algoritmo [27].
- Aprendizaje no supervisado: por otro lado, los algoritmos de aprendizaje no supervisado utilizan datos que no contienen la solución o etiquetas. El algoritmo de aprendizaje trata de buscar patrones y estructuras en los datos para agruparlos o encontrar relaciones entre ellos [27].
- Aprendizaje por refuerzo: con el aprendizaje por refuerzo, el sistema aprende mediante prueba y error, siendo recompensado al tomar las decisiones correctas. De esta forma, el algoritmo busca maximizar las recompensas, explorando distintas acciones y observando como estas afectan a su recompensa [21].

En este trabajo se dispone de un conjunto de datos con sus correspondientes etiquetas, como se mostró en el apartado 3, por lo que únicamente se trabajará con algoritmos de aprendizaje supervisado.

4.3.1 Conceptos previos

4.3.1.1 Modelo Matemático

El modelo matemático relaciona las características de los datos con sus soluciones, lo que nos permite predecir el resultado de un algoritmo. Este modelo está compuesto por dos fases: una fase de entrenamiento y otra fase de predicción. En la fase de entrenamiento, el algoritmo desarrolla el modelo al exponerlo a los datos de entrada. De esta forma, el modelo aprende la relación entre las características de los datos y sus categorías, modificando unas variables, denominadas parámetros, que ajustan el comportamiento del modelo. En la fase de predicción, el algoritmo estima una categoría para datos no etiquetados basándose en el modelo entrenado [7].

4.3.1.2 Hiperparámetros

Los hiperparámetros son variables externas al modelo elegidas por el programador que influirán en el comportamiento del algoritmo. Cada algoritmo posee distintos hiperparámetros, siendo imposible de conocer a priori los mejores valores para estas variables. Los valores de los hiperparámetros se ajustan mediante prueba y error o utilizando valores genéricos [28].

4.3.1.3 Conjunto de datos

El conjunto de datos utilizado para el entrenamiento de algoritmos juega un papel muy importante. Esta base de datos debe de contener muestras suficientes para que el algoritmo funcione adecuadamente. Para problemas simples, es normal utilizar miles de muestras. Es importante evitar errores y ruido en la toma de muestras. Además, los datos deben contener información representativa, de forma que se pueda generalizar el resultado [29].

Los datos se pueden dividir en tres conjuntos: datos de entrenamiento, datos de validación y datos de testeo. Los datos de entrenamiento se utilizan para obtener el modelo matemático del algoritmo, Por otro lado, los datos de validación permiten ajustar los hiperparámetros del algoritmo que mejor resultados obtenga. Finalmente, los datos de testeo son utilizados para evaluar el error del algoritmo, por lo que no son utilizados durante el entrenamiento del modelo. El error obtenido se denomina out-of-sample error. Es común utilizar el 60% de los datos para el entrenamiento, 20% para la validación y 20% para el testeo [29].

4.3.1.4 Overfitting y underfitting

El fenómeno de overfitting se produce cuando el algoritmo no es capaz de generalizar de forma adecuada, ya que se ajusta demasiado a los datos introducidos. Así, El algoritmo será incapaz de reconocer nuevos datos de entrada. Por otro lado, el underfitting se produce cuando no se tiene suficientes muestras como para generalizar el conocimiento [30].

4.3.2 Perceptrón multicapa

El perceptrón multicapa (MLP) es un algoritmo de aprendizaje profundo compuesto por al menos una capa de entrada, una o varias capas ocultas y una capa salida. Cada una de estas capas están compuestas, a su vez, por múltiples neuronas. Para comprender su funcionamiento, explicaremos una serie de conceptos previos necesarios.

4.3.2.1 Perceptrón

El perceptrón es la arquitectura más simple en una red neuronal. Esta arquitectura consta de una capa compuesta únicamente de una neurona artificial. La función de un perceptrón es recibir datos de entrada, entrenar el algoritmo con estos datos para crear un modelo matemático y generar un resultado [7].

El modelo matemático desarrollado por el perceptrón posee la siguiente estructura:

$$y = W * X + b \ (1)$$

Donde:

- Y = Recta que separa las clases
- X = Es la característica de entrada
- W = Pendiente de la recta, denomina vector de pesos.
- B = Es el punto de intersección de la recta en el eje, denominada sesgo.

Así, los elementos que debe aprender el modelo son el vector de pesos W y el sesgo b. Una vez aprendidos, se puede realizar una clasificación binaria de las muestras utilizando una función de activación no lineal. Dicha función de activación puede expresarse como:

$$z = \begin{cases} 1 \ si \ y \geq 0 \\ 0 \ si \ y < 0 \end{cases} (2)$$

Esta clasificación se representa en la Figura 7, donde podemos observar una división de los datos en un espacio de dos dimensiones. Al tratarse de un espacio de dos dimensiones, la función que separa las dos clases

se trata de una recta. Las muestras que pertenecen a la clase 0 tienen forma de cuadrado, mientras que las muestras pertenecientes a la clase 1 han sido representadas como círculos.

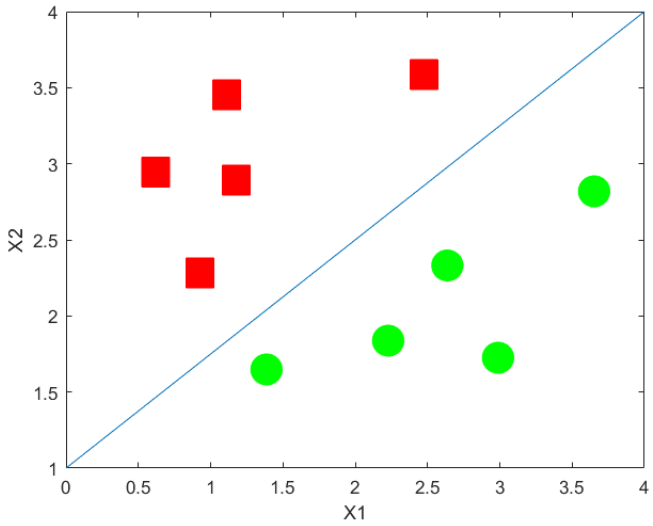

Figura 7. Representación clasificación perceptrón.

4.3.2.2 Función de activación

La función de activación define la salida de una neurona y se utilizan para que los modelos sean no lineales. En el ejemplo anterior hemos implementado una función de activación escalón. Sin embargo, existen numerosas funciones de activación, siendo las más comunes [31]:

- Lineal: la salida es igual a la entrada.

$$f(x) = x \quad (3)$$

- Sigmoide: los valores de salida están comprendidos entre {0,1}, por lo que la salida se interpreta como una probabilidad.

$$f(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

- Tanh: esta función se conoce como tangente hiperbólica. La salida varía entre el intervalo {-1,1} y está centrada en cero.

$$f(x) = \tanh(x) \quad (5)$$

- ReLU: Esta función de activación es muy utilizada para el entrenamiento de redes neuronales. Para valores negativos el resultado es cero, mientras que los valores positivos no se modifican.

$$f(x) = \max(0, x) \quad (6)$$

- Softmax: la función softmax permite la clasificación multiclase. Por lo tanto, esta función se utiliza en la última capa en caso de querer clasificar múltiples clases. Además, permite estimar las probabilidades de las distintas clases, siendo la suma de las probabilidades de todas las clases igual a uno. El resultado será la etiqueta con mayor probabilidad.

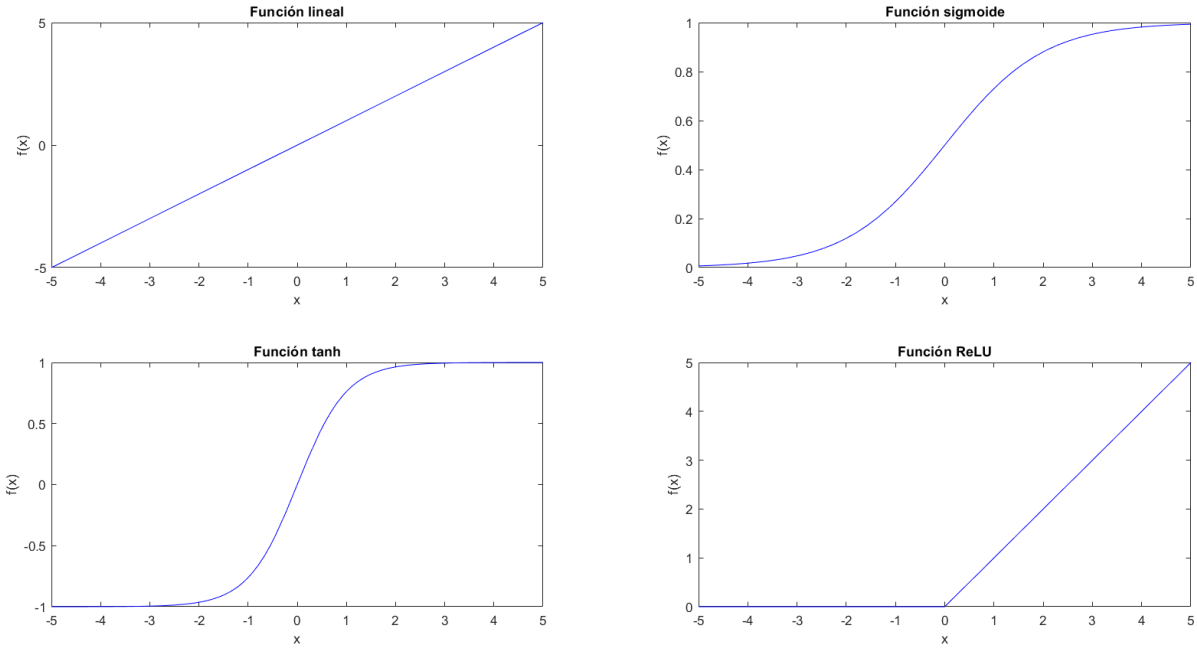

Figura 8. Gráficas de las funciones de activación.

4.3.2.3 Proceso de aprendizaje

El proceso de aprendizaje de una red neuronal consta de una serie iteraciones, mediante los cuales la red neuronal es capaz de aprender los valores de sus parámetros. La primera etapa se denomina forward propagation. En esta fase, los datos se exponen a las capas de la red neuronal, generando predicciones en sus neuronas. La primera capa de la red neuronal utiliza los datos de entrada, mientras que las capas ocultas utilizan la información de la capa anterior. Así, la información se propaga a través de las capas sucesivas hasta llegar a la última, cuyo resultado será la predicción final [7].

Posteriormente, se calcula el error cometido en la predicción final mediante la función de loss, que se emplea para medir la calidad del resultado. El resultado ideal será un error de cero, mientras que un valor mayor indicará un peor rendimiento de la red neuronal. Así, se ajustarán los valores de los parámetros del modelo de la red neuronal para reducir el error.

Una vez calculado el error, esta información se propaga hacia atrás en las capas. Esta fase se conoce como back propagation. Cada neurona de la red recibe una porción de esta información, proporcional a su contribución a la predicción final. Una vez que todas las neuronas han recibido su información correspondiente, los pesos de los parámetros se pueden ajustar nuevamente para mejorar el rendimiento del modelo.

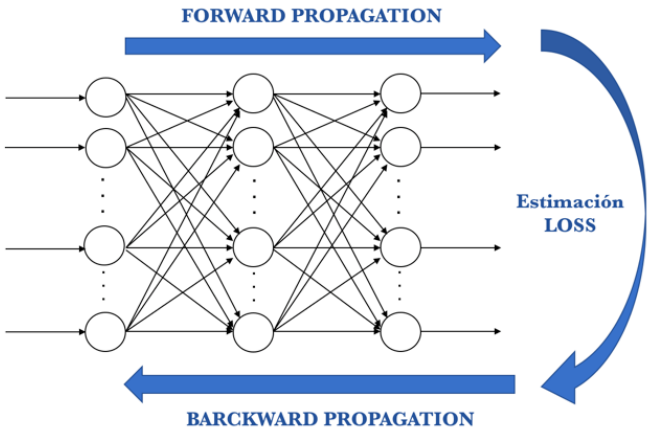

Figura 9. Representación proceso de aprendizaje. Disponible en [7].

Para ajustar los valores de los parámetros, se utilizan varias técnicas llamadas optimizadores. La técnica gradient descent es una de las más populares. Esta técnica modifica los valores de los parámetros en pequeños intervalos, calculando la derivada de la función de loss. La derivada nos permite apreciar que dirección hay que seguir para alcanzar el mínimo global del error. Este proceso se repite hasta alcanzar el número máximo de iteraciones o hasta alcanzar el mínimo error.

4.3.2.4 Hiperparámetros

Al implementar el clasificador perceptrón multicapa en Scikit-learn, se deben tener en cuenta dos factores. El primer factor a tener en cuenta es que la última capa de la red neuronal utiliza la función de activación softmax. También, la única función de loss permitida es Cross-Entropy. Los hiperparámetros modificados son [32]:

- Hidden_layer_sizes: vector con número de capas ocultas y sus neuronas.
- Activation: función de activación utilizadas en las capas ocultas.
- Solver: optimizador utilizado.
- Batch_size: el conjunto de datos se divide en paquetes para entrenar la red neuronal. Este hiperparámetro permite escoger el tamaño de datos que contiene un paquete en cada iteración.
- Max_iter: número de veces que los datos de entrenamientos pasan por la red neuronal.

4.3.3 Árbol de decisión

El árbol de decisión es un algoritmo de aprendizaje automático muy utilizado que presenta una estructura de árbol. Este algoritmo realiza una división recursiva del conjunto de datos hasta clasificarlos en las etiquetas de clases específicas. Los árboles de decisión tienen varias ventajas: son fáciles de interpretar, ya que se pueden representar gráficamente, pueden utilizarse para tareas de clasificación como regresión y permite el manejo de datos continuos y discretos [33]. Su estructura está compuesta por:

- Nodo raíz: es el nodo donde inicia el algoritmo. Por este nodo deberá pasar el conjunto entero de datos dado que no tiene ningún otro nodo previo.
- Nodo de decisión: cada nodo de decisión realiza una división de los datos y representa una característica del conjunto de datos.
- Nodo hoja: cada nodo hoja representa un resultado, es decir, una etiqueta de clasificación. Este nodo termina el flujo del diagrama.
- Rama: cada rama simboliza una opción posible y conecta dos nodos.

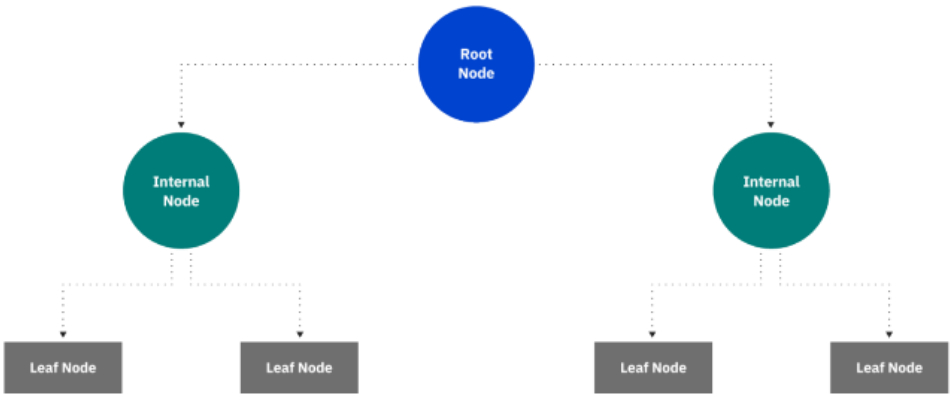

Figura 10. Estructura del árbol de decisión. Disponible en [33].

4.3.3.1 Función de impureza

La función de impureza nos permite seleccionar el mejor criterio de división posible en cada nodo, es decir, la mejor característica para la separación de los datos. Estas funciones evalúan la calidad de cada condición, de manera que los datos se puedan clasificar de la mejor forma posible. Existen varias funciones de impurezas, entre las que destaca la impureza de Gini, que mide la probabilidad de clasificar incorrectamente un punto de datos aleatorio, y la entropía, que mide la impureza de los valores de las muestras [33].

4.3.3.2 Hiperparámetros

Los árboles de decisión tienen muchos hiperpárametros que se pueden ajustar. A continuación, explicaré los hiperparámetros que han sido modificados para nuestro conjunto de datos que nos permiten obtener un mejor resultado [34].

- Max_depth: profundidad del árbol de decisión
- Min_samples_split: mínimo número de muestras para dividir un nodo.
- Min_samples_leaf: mínimo número de muestras en un nodo hoja.
- Criterion: función de impureza utilizada.

4.3.4 K Vecinos más próximos

El algoritmo K-Nearest Neighbors (KNN) es un algoritmo de aprendizaje automático supervisado cuyo principio se basa en la distancia entre el conjunto datos. Los datos con características similares tienden a situarse próximos en el espacio de características. Para la clasificación, se asigna al nuevo dato la etiqueta más común entre los k vecinos más próximos [35].

Cabe destacar que este algoritmo se considera un modelo de "aprendizaje perezoso", ya que esta técnica no entrena un modelo. En lugar de eso, el algoritmo almacena el conjunto de datos para realizar el cálculo de las distancias en la fase de predicción. Como inconveniente, al aumentar el número de datos, se ocupa más memoria y aumentará el tiempo de procesado, por lo que este algoritmo no escala bien.

En la Figura 11 podemos observar un ejemplo de la representación del espacio de características de dos dimensiones del algoritmo k vecinos más cercanos. El dato para clasificar es el objeto central con forma de estrella. En función del número de vecinos utilizados, podemos observar que el resultado de la clasificación variará. Así, si se escoge utilizar k=3, el resultado será la clase B, mientras que si se utiliza k=6, se asigna el nuevo dato a la clase A.

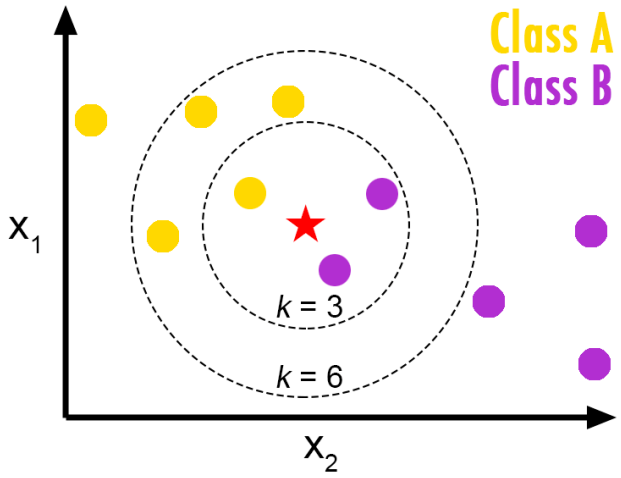

Figura 11. Representación de k vecinos más próximos. Disponible en [36].

4.3.4.1 Métrica de la distancia

Como ya se ha mencionado, este método se basa en la distancia entre el conjunto de datos. Para determinar la proximidad entre el punto a clasificar y sus vecinos, se pueden utilizar varias medidas, entre la que destaca la distancia euclidiana y la distancia Manhattan. La distancia euclidiana mide la línea recta entre dos puntos mientras que la distancia Manhattan mide el valor absoluto.

Fórmula distancia euclidiana: $\boldsymbol{d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)}}$ (7)

Fórmula distancia Manhattan: $\boldsymbol{d(p,q) = \sum_{i=1}^{n}|p_i - q_i|}$ (8)

Donde

- d(p,q) = distancia
- $P = (p_1, p_2, \dots, p_n)$
- $Q = (q_1, q_2, \dots, q_n)$
- N = número de dimensiones

4.3.4.2 Hiperparámetros

Este algoritmo posee pocos hiperparámetros a modificar. Únicamente variaremos el número de vecinos más cercanos (n_neighbors) y la métrica de la distancia [37].

- N_neighbors: número de vecinos más próximos.
- P: permite variar entre la distancia euclidiana y la distancia Manhattan.

4.3.5 Máquina de Vector Soporte

Las Máquinas de Vector Soporte (SVM) constituyen un conjunto de algoritmos de aprendizaje supervisado desarrollados en la década de los 90. Su funcionamiento se basa en la división de los datos en dos o más categorías mediante un hiperplano. En el contexto de las SVM, un hiperplano es la superficie que separa los puntos de datos en diferentes clases. Este algoritmo se esfuerza por encontrar el hiperplano que pueda separar las muestras entre distintas categorías con la mayor distancia mínima posible entre el hiperplano y las poblaciones de datos, es decir, maximizando el margen que separa las clases [38].

En la Figura 12 observamos un ejemplo de la aplicación del algoritmo SVM para la clasificación de dos clases. El hiperplano es la línea negra que separa ambas poblaciones, situado con el máximo margen entre las muestras de las dos clases.

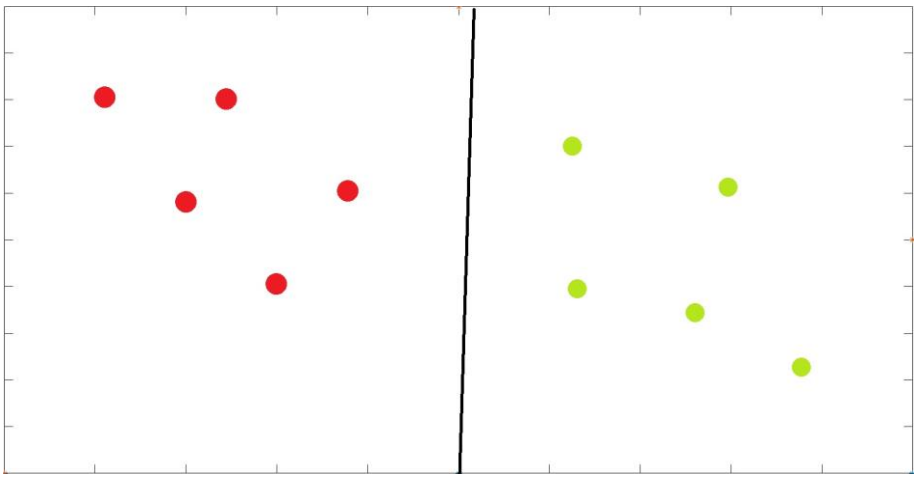

Figura 12. Representación algoritmo SVM.

4.3.5.1 Kernel

En ocasiones, la separación de la clase puede ser no lineal, obteniendo peores resultados. Sin embargo, al expandir las dimensiones de las muestras, se puede llegar a conseguir una separación lineal de las categorías. Para aumentar las dimensiones se emplea Kernel. Las funciones de Kernel más comunes son Kernel lineal, polinómico y gaussiano.

En la Figura 13 podemos observar un ejemplo del aumento de dimensiones aplicando Kernel. En el gráfico de la izquierda las muestras de las dos clases en dos dimensiones se encuentran mezcladas, de forma que es imposible utilizar un hiperplano para la clasificación. Por otro lado, en el gráfico de la derecha se ha aumentado a tres dimensiones, existiendo una división clara entre las dos categorías.

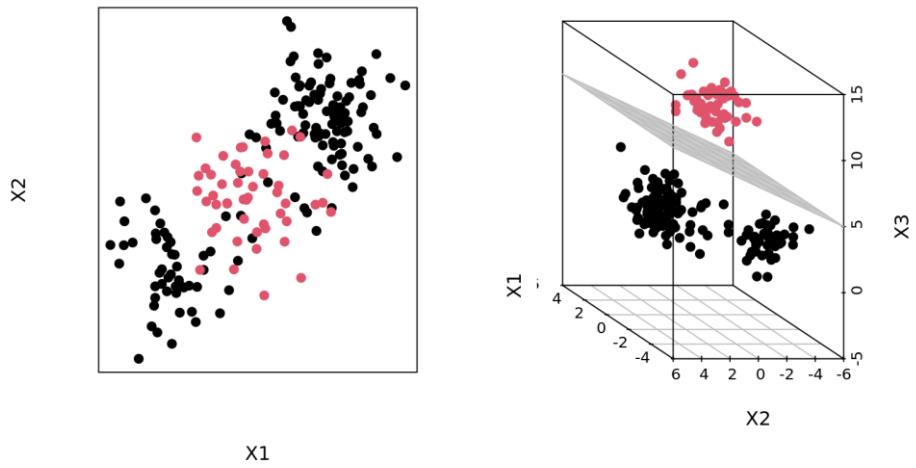

Figura 13. Aumento de dimensiones. Disponible en [38].

4.3.5.2 Separación multiclase

La separación mediante el hiperplano es capaz de dividir dos clases distintas, pero no es capaz por sí solo de clasificar más de dos clases. Para ello, se han inventado varios métodos que permiten a este algoritmo clasificar múltiples clases [38].

- One versus one: para un total de N clases, se entrenan N(N-1)/2 SVMs, comparando todos los posibles pares de clases. El resultado será la clase más asignada en las clasificaciones de los distintos SVMs.
- One versus all: en este método se entrenan N SVMs, comparando una de las clases frente al resto. Se asigna la clase cuya predicción sea positiva.
- DAGSVM: este método utiliza el mismo principio que el primer método, mejorando el rendimiento. Se acorta el tiempo de entrenamiento eliminando la comparación innecesaria entre pares. Una vez que una clase haya sido asignada frente a otra, no se realizarán más comparaciones con la clase no seleccionada.

4.3.5.3 Hiperparámetros

Los hiperparámetros modificados en el algoritmo SVM se pueden encontrar en [39] y son:

- C: variable para ajustar la regularización. La regularización evita que el modelo se ajuste demasiado a los datos de entrenamiento, lo que llevaría a un overfitting. A menor valor de C, se permite un margen del hiperplano más amplio, resultando en un modelo más generalizado. Al aumentar el valor de C, se obtiene un modelo más estricto, pero con menor capacidad de generalización.

- Kernel: define el Kernel utilizado en el algoritmo y permite emplear un Kernel lineal, polinómico, gaussiano o sigmoide.
- Probability: permite estimar la probabilidad. Esta opción será necesaria para algunos algoritmos de ensamble de clasificadores.
- Decision_function_shape: define el método a utilizar para la separación multiclase. Se puede optar por one versus one o one versus all.

4.4 Algoritmos de ensamble de clasificadores

Los algoritmos de ensamble de clasificadores son una técnica de Machine Learning donde se utiliza las predicciones de varios modelos individuales para crear un modelo final más robusto y preciso. Este método mejora la capacidad de generalización a costa de una mayor complejidad computacional [29].

4.4.1 Clasificador por votación

Los clasificadores por votación junta las predicciones de varios clasificadores individuales, obteniendo como resultado final la clase más votada. Estos clasificadores se entrenan con el mismo conjunto de datos. En este método es fundamental utilizar distintos tipos de algoritmos, de forma que sean lo más independientes entre sí. Cada algoritmo cometerá distintos tipos de errores, los cuales se compensarán al integrar las predicciones de los demás [40].

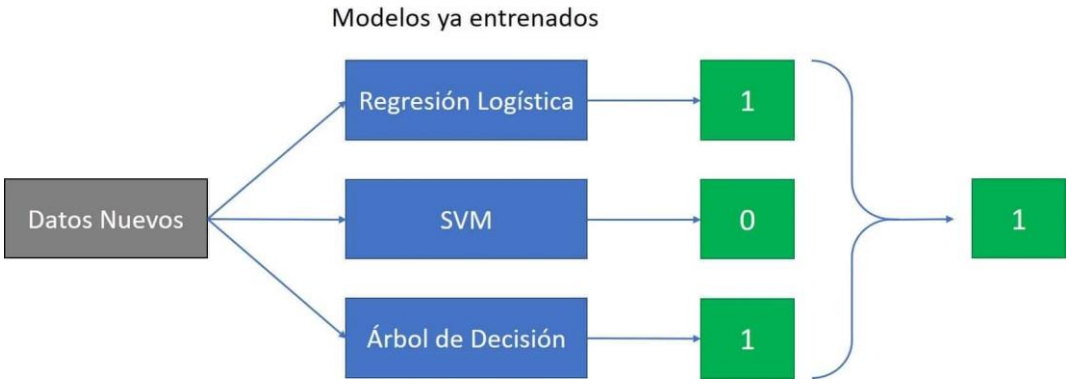

Figura 14. Ejemplo clasificador por votación. Disponible en [40].

4.4.1.1 Hard voting vs soft voting

Existen dos métodos para combinar las votaciones, denominadas ‘hard voting’ y ‘soft voting’. El método ‘hard voting’ emplea las predicciones finales de los clasificadores. Por otro lado, ‘soft voting’ emplea la probabilidad de las clases de los clasificadores para la votación. La clase con mayor probabilidad será escogida. Sin embargo, hay que tener en cuenta que no todos los clasificadores proporcionan la probabilidad de las clases.

4.4.1.2 Hiperparámetros

Este clasificador cuenta con pocos hiperparámetros. En nuestro caso hemos modificado los siguientes [41]:

- Estimator: incluye los clasificadores individuales utilizados para general la predicción final.
- Voting: se debe seleccionar entre ‘hard voting’ y ‘soft voting’.
- Weigths: peso que se le da a cada clasificador.

4.4.2 Bagging y Pasting

Ambos métodos utilizan múltiples modelos del mismo algoritmo, entrenados de forma independiente con distintos conjuntos de datos. Los distintos modelos se pueden entrenar en paralelo, ahorrando tiempo de ejecución. Estos métodos se aplican especialmente a clasificadores individuales con alto sesgo, es decir, aquellos que tienden a cometer errores sistemáticos. Al igual que en el clasificador por votación, se emplea la clase más frecuente como resultado final [42].

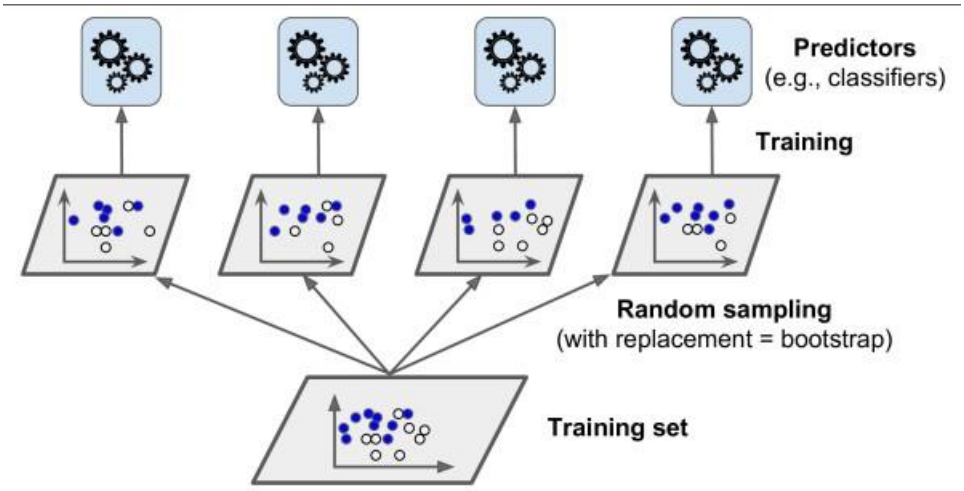

Figura 15. Ejemplo clasificador Bagging. Disponible en [29].

La diferencia entre ambos métodos radica en la forma de utilizar el conjunto de datos. En el método de Pasting, el conjunto de datos se submuestrea en diferentes subconjuntos, asegurando que un mismo dato no aparezca en más de un subconjunto. Por otro lado, el método de Bagging presenta una técnica conocida como Bootstrap, que permite que una misma muestra pueda aparecer en dos o más subconjuntos.

4.4.2.1 Hiperparámetros

En Scikit-learn se emplea el algoritmo Bagging por defecto con la opción de cambiar a Pasting mediante una variable [43].

- Bootstrap: esta variable permite cambiar de Bagging a Pasting.
- Estimator: clasificador individual a utilizar.
- N_estimator: número de clasificadores base empleados.
- Max_samples: número de muestras a extraer para cada clasificador base. Únicamente se usa en el método Bagging.
- Max_features: número de características a extraer para cada clasificador base.
- N_jobs: permite entrenar varios estimadores bases en paralelo utilizando varios núcleos del ordenador.
- Oob-score: al utilizar Bagging, el 37% de las muestras no son utilizadas. A estas muestras se les denomina out-of-bag. Se pueden utilizar como datos de validación, evaluando el modelo.

4.4.3 Random forest

Random forest es un algoritmo de ensamble entrenado con el método Bagging o Pasting, en el cual se utiliza como estimador base el árbol de decisión. Es considerado uno de los algoritmos más populares y potentes en la actualidad. Siguiendo la estrategia Bagging y Pasting, este algoritmo entrena múltiples árboles de decisiones en diferentes subconjuntos de datos y finalmente utiliza la clase más votada como resultado [44].

Al aplicar Random Forest, no todas las características de los datos son aplicadas en cada árbol de decisión. En cambio, cada árbol se entrena con diferentes subconjuntos de características, lo que proporciona mayor diversidad en su crecimiento. Además, los árboles de decisión suelen enfrentar el problema de overfitting, que es mitigado al aplicar la técnica de Random Forest.

4.4.3.1 Extremely Randomized Trees

Como se mencionó previamente, el algoritmo Random Forest divide el conjunto de características en subconjuntos distintos para entrenar los árboles de decisiones. Cada árbol de decisión busca los umbrales de división óptimos en los nodos para cada característica. Esta estrategia es distinta para los Extremely Randomized Trees, donde no se busca el mejor umbral, sino que se utiliza un umbral aleatorio para cada atributo. Esta aleatoriedad permite una mayor diversidad en los árboles, formando un modelo más robusto y generalizado [29].

4.4.3.2 Hiperparámetros

Los hiperparámetros de Random Forest y Extremely Randomized Trees son iguales, y contienen los hiperparámetros del árbol de decisión y el método Bagging [45] [46].

- N_estimators: número de clasificadores base empleados.
- Max_samples: número de muestras a extraer para cada clasificador base.
- Bootstrap: esta variable permite cambiar de Bagging a Pasting
- Max_depth: profundidad del árbol de decisión
- Min_samples_split: mínimo número de muestras para dividir un nodo.
- Min_samples_leaf: mínimo número de muestras en un nodo hoja.
- Criterion: función de impureza utilizada.
- N_jobs: permite entrenar varios estimadores bases en paralelo utilizando varios núcleos del ordenador.
- Oob_score: al utilizar un método Bagging, el 37% de las muestras no son utilizadas. A estas muestras se les denomina out-of-bag. Se pueden utilizar como datos de validación, evaluando el modelo.

4.4.4 Adaboost

Adaboost es un algoritmo de ensamble de clasificadoras que se utiliza para mejorar el rendimiento de múltiples clasificadores débiles, creando así un modelo de clasificación fuerte. Este método utiliza la técnica boosting. Esta técnica entrena múltiples algoritmos de un mismo clasificador secuencialmente, mejorando los errores cometidos por el modelo previo [47].

4.4.4.1 Funcionamiento

En concreto, Adaboost se centra en la importancia que se otorga a los datos. En primer lugar, se le otorga el mismo peso a cada dato, creando un modelo del clasificador débil [29].

$$w_i(x_i, y_i) = \frac{1}{N} \quad (9)$$

Donde:

- $w(x_i, y_i)$ = Peso de los datos
- N = Número de datos

Posteriormente, se calcula la eficiencia del clasificador. A mayor número, más preciso será el modelo. Sin embargo, si el modelo acierta menos que de forma aleatoria, la importancia del clasificador puede ser negativa:

$$\alpha = \eta \log(\frac{1-Error\ total}{Error\ total}) \quad (10)$$

Donde:

- α = Importancia del clasificador
- η = Learning rate
- Error total = suma de los pesos de los datos mal clasificados

Finalmente, los datos clasificados incorrectamente aumentan de peso y se normalizan. Al incrementar el peso de los ejemplos mal clasificados conseguimos que el próximo modelo se centre en ellos.

Ejemplos bien clasificados: $w_i^* = w_i$ (11)

Ejemplos mal clasificados: $w_i^* = w_i * e^{\alpha}$ (12)

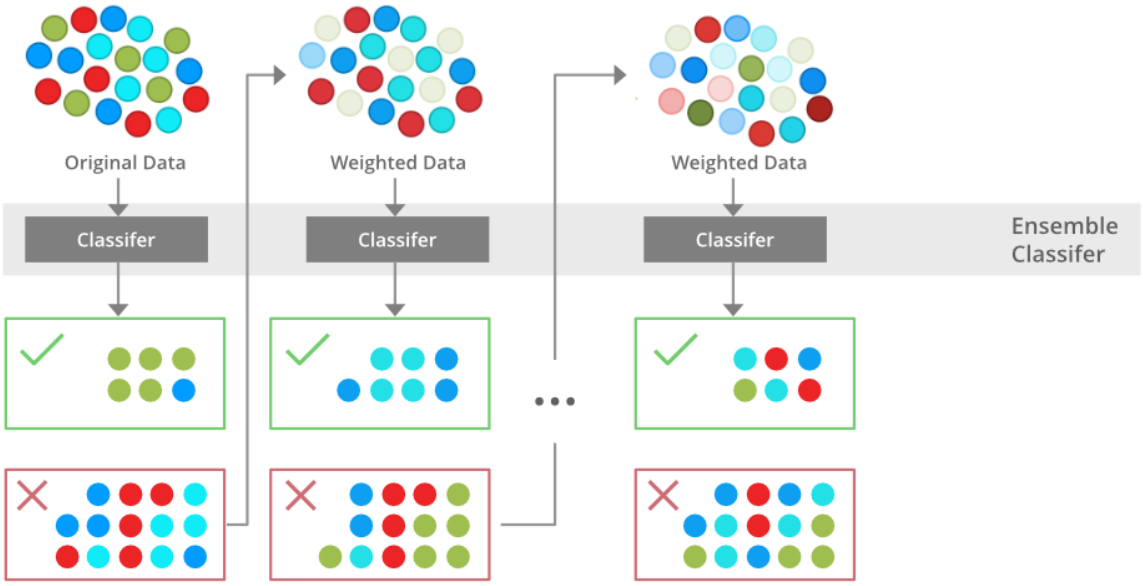

Figura 16. Ejemplo clasificación Adaboost. Disponible en [48].

Una vez se hayan normalizado los datos con sus pesos actualizados, se itera hasta obtener un menor error o hasta que se realiza el número de predicciones deseadas. Para realizar la predicción, se escoge la clase más votada, otorgando a cada clasificador débil un peso en la votación.

4.4.4.2 Hiperparámetros

Los hiperparámetros para el método Adaboost son [49]:

- Estimator: clasificador base.
- N_estimators: número máximo de estimaciones realizadas.
- Learning_rate: peso aplicado a cada clasificador de cada iteración. A mayor valor, mayor influencia tendrán los nuevos clasificadores.

4.4.5 Gradient Boosting

Gradient Boosting es un algoritmo de aprendizaje supervisado que utiliza la técnica Boosting para aumentar el

rendimiento de un clasificador base. Al igual que el método AdaBoost, Gradient Boosting emplea la estrategia Boosting, explicada anteriormente. Sin embargo, ambos modelos se diferencian en cómo se corrigen los errores. A diferencia de AdaBoost, que trata de corregir los errores otorgando mayor importancia a los datos mal clasificados, Gradient Boosting utiliza los errores residuales cometidos por los modelos anteriores [50].

4.4.5.1 Funcionamiento

El funcionamiento del modelo puede explicarse en 4 pasos:

1- Se entrena un primer modelo. Como algoritmo base se suele utilizar los árboles de decisión poco profundos, ya que suelen proporcionar un mejor resultado.

2- Se calcula el error residual del modelo. Estos errores residuales se utilizan como etiquetas para el nuevo modelo.

$$F_n(x) = y - \eta F_1(x) - \eta F_2(x) - \cdots - \eta F_{n-1}(x) \quad (13)$$

Donde:

- $F_n(x)$ = Error residual de los n-1 modelos anteriores.
- y = Etiqueta inicial del conjunto de datos.
- η = Factor de aprendizaje
- n = número total de modelos entrenados.

3- Se entrena el siguiente modelo con las nuevas etiquetas. Se utilizan los errores residuales como etiquetas para que el nuevo modelo sea capaz de corregir estos errores.

4- Se itera hasta alcanzar el número máximo de estimaciones o hasta alcanzar un umbral de error deseado.

La predicción final se calcula como la suma de las predicciones de los modelos por su factor de aprendizaje (Learning rate).

4.4.5.2 Hiperparámetros

En Scikit-learn, el clasificador base utilizado para Gradient Boosting es obligatoriamente el árbol de decisión, por lo que este algoritmo empleará hiperparámetros propios de Boosting y de los árboles de decisión [51].

- N_estimators: número máximo de estimaciones realizadas.
- Learning_rate: peso aplicado a cada clasificador de cada iteración. A mayor valor, mayor influencia tendrán los nuevos clasificadores.
- Max_depth: profundidad del árbol de decisión
- Min_samples_split: mínimo número de muestras para dividir un nodo.
- Min_samples_leaf: mínimo número de muestras en un nodo hoja.
- Criterion: función de impureza utilizada.

5 RESULTADOS

En este apartado se presentarán los resultados obtenidos aplicando los distintos algoritmos de aprendizaje supervisados explicados durante el apartado 4. Realizaremos un estudio de los mejores hiperparámetros para cada método, experimentando con distintas combinaciones hasta obtener la mejor precisión posible. Desarrollaremos un estudio tanto para la clasificación de tipos de golpes de pádel como para la clasificación del nivel del jugador. La base de los códigos ha sido tomada de los trabajos predecesores [10] [11], realizando ciertas modificaciones sobre los códigos para adaptarlos a las necesidades de este proyecto, y ampliando los algoritmos utilizados, intentando mejorar a través de los métodos de ensamble las predicciones de los clasificadores. Además, las redes neuronales desarrolladas en Keras por los trabajos predecesores no son compatibles con los ensambles de clasificadores de Scikit-learn, por lo que se propone el algoritmo perceptrón multicapa para su reemplazo.

Como datos de entrada, utilizaremos las muestras tomadas por los sensores. Los algoritmos se entrenarán con las 240 muestras de cada golpe realizado, como se mencionó en el apartado 3. No se utilizarán ningún otro elemento de la Tabla 1 para la clasificación de los golpes, a parte de la etiqueta necesaria, ya que se han empleado algoritmos de aprendizaje supervisado.

En los trabajos previos [10] [11], aparte de manejar la serie temporal de los datos, se utilizan otros dos métodos: "feature engineering" y el dominio de la frecuencia. Feature engineering o ingeniería de características se basa en encontrar las mejores características de los datos para entrenar los modelos. Se extrajo para cada grado de libertad el valor máximo, el valor mínimo y el valor medio, reduciendo el número de muestras empleadas en el entrenamiento. Por otro lado, la transformación del dominio del tiempo al dominio de la frecuencia nos permite analizar la frecuencia a la que oscilan los datos. En nuestro caso, debido a los peores resultados obtenidos a través de estos dos métodos, únicamente se empleará la serie temporal de los golpes.

La división de los datos será distinta a la mostrada en el apartado 4.3.1.3, donde se utilizaba como ejemplo una división de 60% para los datos de entrenamiento, 20% para los datos de validación y 20% para los datos de testeo. Los algoritmos entrenados en este trabajo no necesitan el conjunto de datos de validación. En consecuencia, se empleará un 70% para los datos de entrenamiento y un 30% para los datos de testeo en todos los algoritmos utilizados. La partición de los datos se llevará a cabo con una aleatoriedad constante, es decir, los datos se dividen de forma aleatoria, pero esta aleatoriedad será igual para todos los algoritmos.

Los valores de los hiperparámetros utilizados se mostrarán en tablas y se especificará que valores producen un mejor resultado del estimador. En algunos casos, se mostrarán las exactitudes obtenidas al variar el valor de estos parámetros. Además, para cada algoritmo de ensamble de clasificador se comparará su resultado con la exactitud del estimador base.

La búsqueda de los mejores hiperparámetros se realizará con la herramienta 'grid search' o búsqueda en rejilla. En esta herramienta, se deberá introducir los distintos hiperparámetros y sus valores que deseamos comparar. Esta técnica entrena el modelo con cada combinación de hiperparámetros y se selecciona aquel modelo que produce el mejor rendimiento, en nuestro caso medido por la accuracy.

Se utilizarán dos medidas para evaluar el rendimiento de los distintos algoritmos. En primer lugar, la accuracy o exactitud, que muestra la proporción de predicciones correctas sobre el total de predicciones. Esta métrica da a entender cuán bien el modelo está clasificando las instancias correctamente y se utiliza cuando las clases están balanceadas.

Además, para analizar en profundidad el rendimiento de cada clasificador, se utilizará una herramienta conocida como matriz de confusión. Esta matriz proporciona una representación detallada de las predicciones realizadas por el clasificador en comparación con las etiquetas reales de las muestras en el conjunto de datos de prueba. En ella, las filas representan las clases reales de las muestras, mientras que las columnas representan las clases predichas por el clasificador. Así, se puede identificar los tipos de errores que un algoritmo puede

cometer al realizar las predicciones. La nomenclatura utilizada para la matriz de confusión es la mostrada en la Tabla 2.

Tabla 2. Nomenclatura tipo de golpes.

D: fondo de derecha	R: revés de fondo
DP: derecha con pared	RP: revés con pared
GD: globo de derecha	GR: globo de revés
GDP: globo de derecha con pared	GRP: globo de revés con pared
VD: volea de derecha	VR: volea de revés
B: bandeja	RM: remate
S: saque	

Como aclaración, no se modificarán los valores de los hiperparámetros de los clasificadores bases en los métodos de ensamble. Al variar sus parámetros, sería posible encontrar un mejor resultado en algunos casos, pero perdería el sentido de comparación entre los algoritmos bases y su ensamble.

5.1 Clasificación tipo de golpe

5.1.1 Perceptrón multicapa

Nuestro algoritmo perceptrón multicapa contará con una capa de entrada de 240 neuronas, con las características de los datos introducidos, y una capa de salida de 13 neuronas, con las etiquetas empleadas.

Para el algoritmo perceptrón multicapa, dividiremos los resultados dependiendo del número de capas ocultas utilizadas. Se estudiará las mejores variables tanto para una capa oculta como para dos capas ocultas. Los hiperparámetros que modificaremos en cada capa se explicaron en el apartado 4.3.2. Estos son el batch size, la función de activación y el número de iteraciones. El optimizador utilizado se mantendrá igual para los distintos modelos y será el optimizador Adam.

5.1.1.1 Perceptrón multicapa con una capa oculta

Los valores tomados para cada hiperparámetro, disponiendo de una capa oculta, se muestran en la Tabla 3.

Tabla 3. Hiperparámetros perceptrón multicapa con una capa oculta.

N.º neuronas	100, 200, 500, 1000
Batch size	10, 30, 50, 70
Función de activación	ReLU, Tanh, sigmoide
N.º iteraciones	70, 100, 150

El mejor resultado se obtiene configurando el número de neuronas de la capa oculta a 1000, el tamaño del lote a 30, la función de activación ReLU y el número de iteraciones a 70. Con estos valores, la exactitud obtenida es del **88.503% (±0.362)**. Cabe destacar que la red neuronal llega a dicha exactitud en la iteración 46.

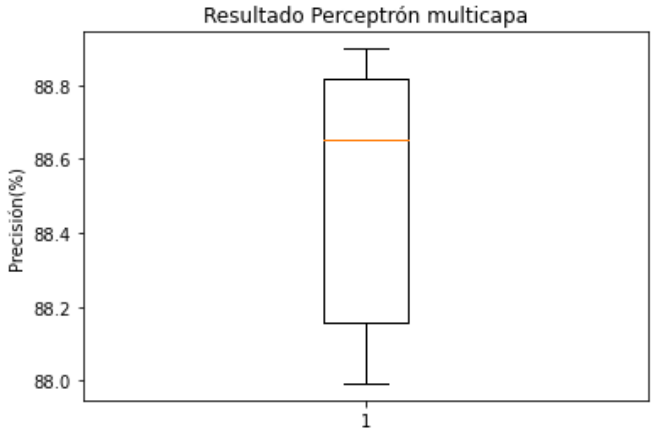

Figura 17. Resultado MLP con una capa oculta. Clasificación tipo de golpe.

En la Figura 18 se puede ver la matriz de confusión para la mejor configuración, es decir, 1000 neuronas en la capa oculta, tamaño de lote 30, función de activación ReLU y número de iteraciones 70. Se producen mayores errores en los golpes similares, como, por ejemplo, se cometen 8 errores al clasificar la volea de revés con el revés, 7 errores al clasificar el globo de derecha con el globo de derecha con pared y 7 errores al clasificar el remate con la bandeja. Por tanto, se puede confirmar que al clasificador le resulta más complejo predecir golpes que tienen un movimiento de brazo parecido, lo cual resulta lógico. También, se puede observar una tendencia del estimador a errar golpes realizados desde el mismo lado, es decir, un golpe de derecha se podrá confundir con otro golpe de derecha diferente, al igual que el revés. Sin embargo, existen pocos errores al clasificar un golpe de derecha como golpe de revés, o viceversa.

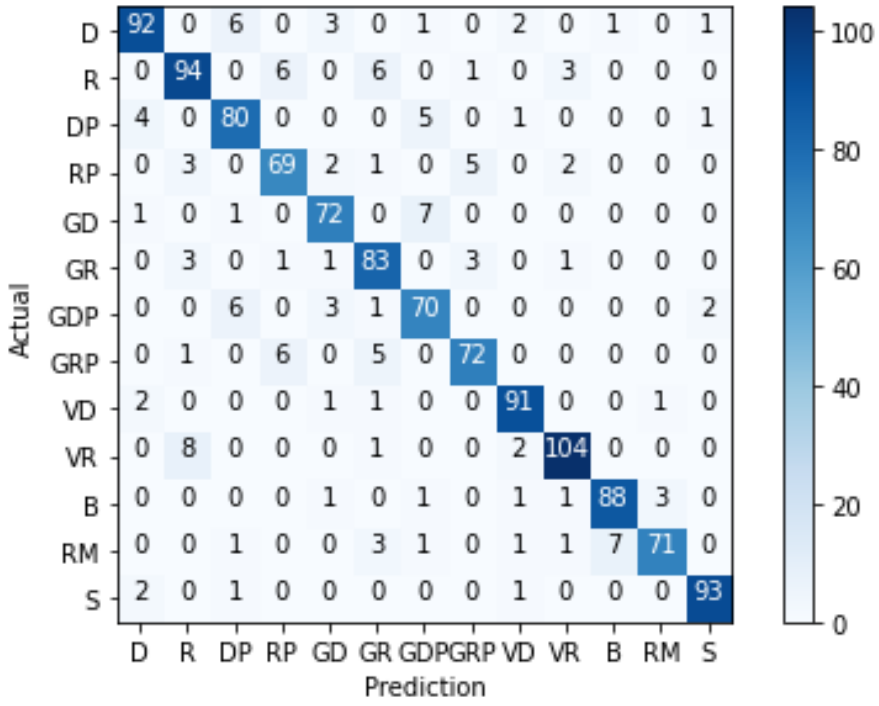

Figura 18. Algoritmo MLP con una capa oculta para clasificación tipo de golpe. Matriz de confusión.

5.1.1.2 Perceptrón multicapa con dos capas ocultas

Al aplicar una segunda capa oculta, dividiremos a la mitad el número de neuronas empleadas en la segunda capa. Los parámetros variados para este caso se muestran en la Tabla 4.

Tabla 4. Hiperparámetros perceptrón multicapa con dos capas ocultas.

N.º neuronas	(200,100), (500,250), (1000,500)
Batch size	10, 30, 50, 70
N.º iteraciones	70, 100, 150

La combinación que mejor predicción ofrece es utilizar 1000 neuronas en la primera capa oculta y 500 en la segunda capa, tamaño de lote 30 y número de iteraciones 70. La función de activación para las capas ocultas será la función ReLU. La exactitud obtenida con estos parámetros es del **89.589% (±1.191)**, alcanzando dicho resultado en 30 iteraciones.

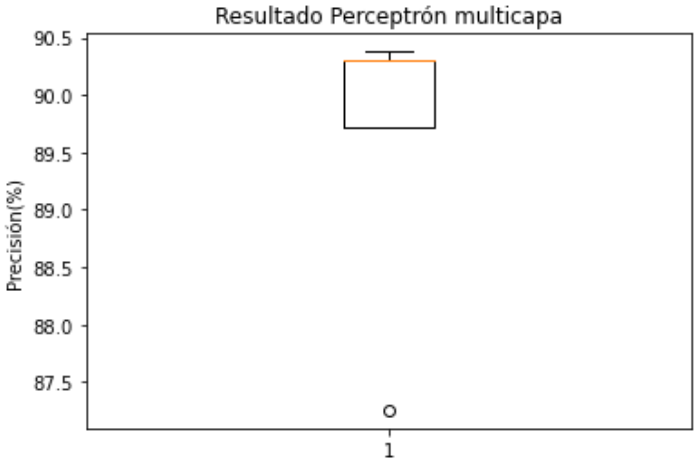

Figura 19. Resultado MLP con dos capas ocultas. Clasificación tipo de golpe.

El resultado aplicando dicha combinación se muestra en la Figura 20. La matriz de confusión nos enseña los errores cometidos por el clasificador. Los mayores errores producidos por este estimador se dan al confundir el fondo de revés con el globo de revés, cometiendo 9 fallos, la derecha después de pared con el globo de derecha después de pared, con 8 fallos, y el globo de revés después de pared con el revés después de pared, cometiendo 7 fallos. Al igual que el clasificador anterior, el algoritmo se confunde al clasificar golpes con movimiento de brazo parecidos.

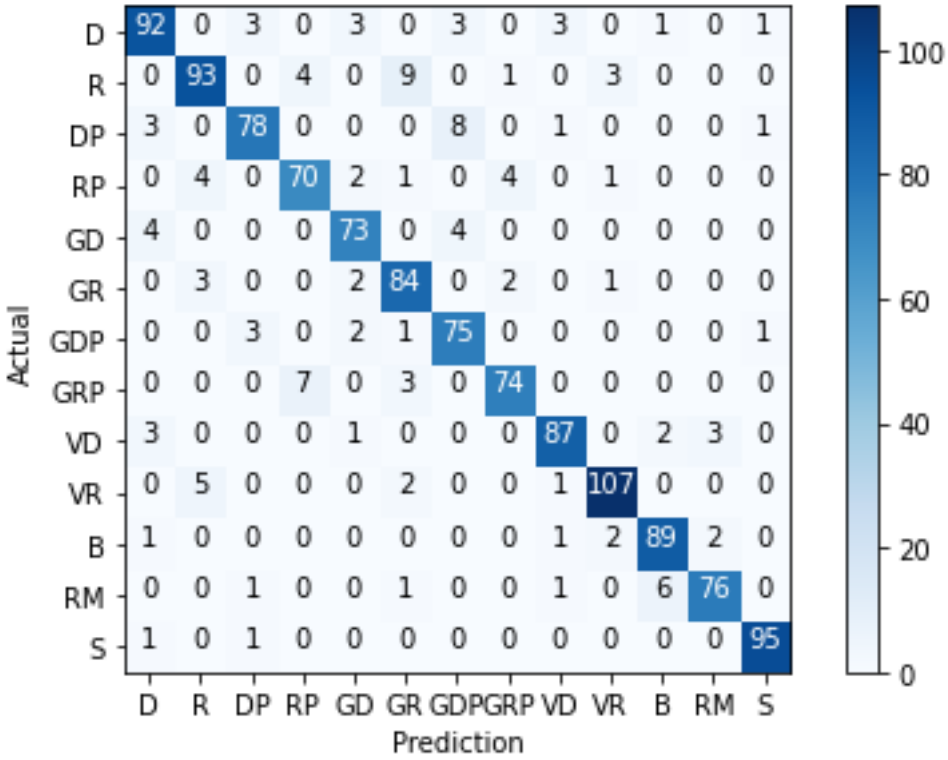

Figura 20. Algoritmo MLP con dos capas ocultas para clasificación tipo de golpe. Matriz de confusión.

5.1.2 Árbol de decisión

El árbol de decisión, detallado en el apartado 4.3.3, cuenta con un gran número de parámetros a modificar. Los valores modificados para cada parámetro se muestran en la Tabla 5.

Tabla 5. Hiperparámetros árbol de decisión.

Max_depth	1, 10, 20, 30, 40
Min_samples_split	2, 4, 8, 10, 20, 100
Min_samples_leaf	1, 2, 3, 4, 5, 10
Criterion	Entropy, Gini

La búsqueda en rejilla nos proporciona la mejor combinación de los hiperparámetros, que se obtiene al emplear los siguientes valores: max_depth=20, min_samples_split=4, min_samples_leaf=1, criterion=Entropy. La exactitud obtenida al utilizar dichos valores es del **54.087% (±0.909)**.

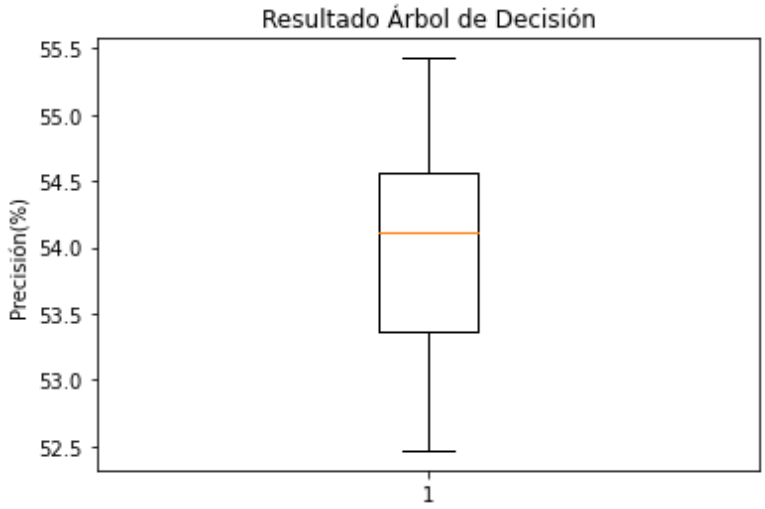

Figura 21. Resultado árbol de decisión con mejor configuración. Clasificación tipo de golpe.

La matriz de confusión mostrada en la Figura 22 nos enseña los errores y aciertos cometidos por el clasificador aplicando los mejores hiperparámetros. Al tener una peor exactitud que los clasificadores comentados anteriormente, se puede apreciar un mayor número de fallos en la clasificación. Entre los errores, destacan los 19 fallos cometidos al catalogar el revés como revés con pared, y 16 fallos al catalogar RP como R, VR como R y GR como GRP. En este caso, los errores remarcados son golpes de revés.

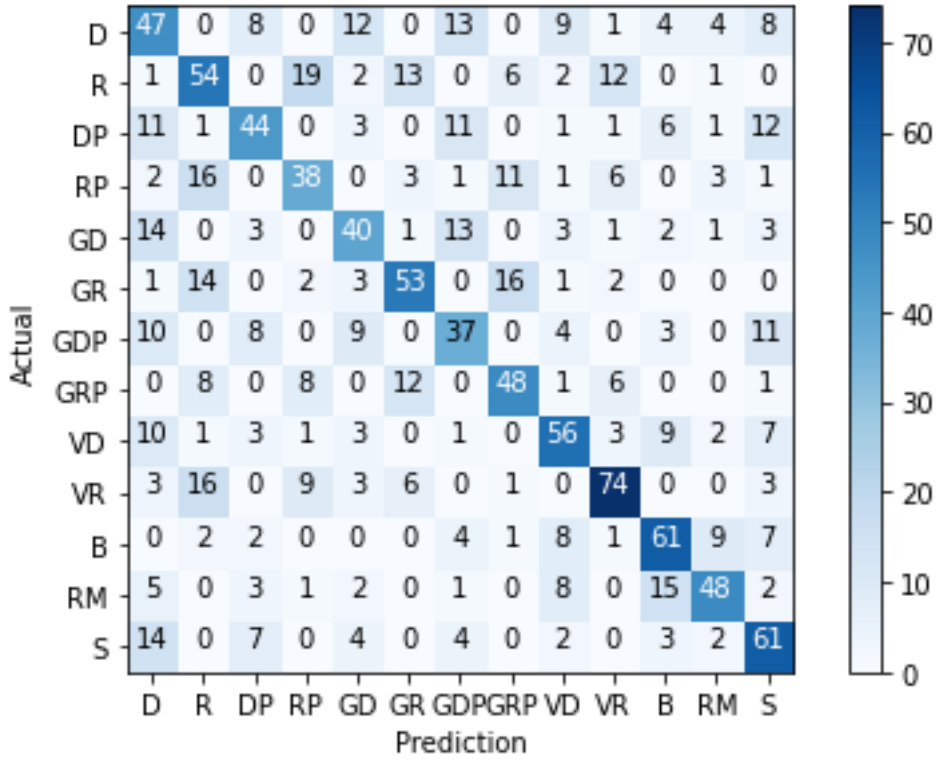

Figura 22. Algoritmo árbol de decisión para clasificación tipo de golpe. Matriz de confusión.

5.1.3 K vecinos más próximos

Como se explicó en el apartado 4.3.4, este clasificador tiene dos variables que podemos variar en busca de un mejor resultado. Estas variables son el número de vecinos más próximos y la métrica empleada para medir la

distancia entre las distintas muestras. En la Tabla 6, se estudia la exactitud obtenida en función de los hiperparámetros utilizados.

Tabla 6. Exactitud (%) algoritmo KNN en función de los Hiperparámetros. Clasificación tipo de golpe.

		K									
		1	2	3	4	5	6	7	8	9	10
Distancia	Manhattan	88.40	84.70	86.27	85.36	84.38	83.39	83.88	81.74	80.67	80.01
	Euclidiana	81.83	76.64	78.78	77.80	77.06	77.47	76.48	75.41	73.77	72.62

Podemos apreciar que la mejor precisión se obtiene para k=1 y utilizando como medida de la distancia la función Manhattan, con una exactitud del **88.40%**. La diferencia entre emplear la función euclidiana y Manhattan supone una variación de alrededor de un 7% de aciertos. Además, se observa que a medida que aumentamos el número de vecinos más cercanos, disminuye la exactitud en ambas funciones. Este valor es constante para una misma división de datos, por lo que se probará esta combinación para distintas separaciones de datos. La exactitud media obtenida es del **87.418% (± 0.997).**

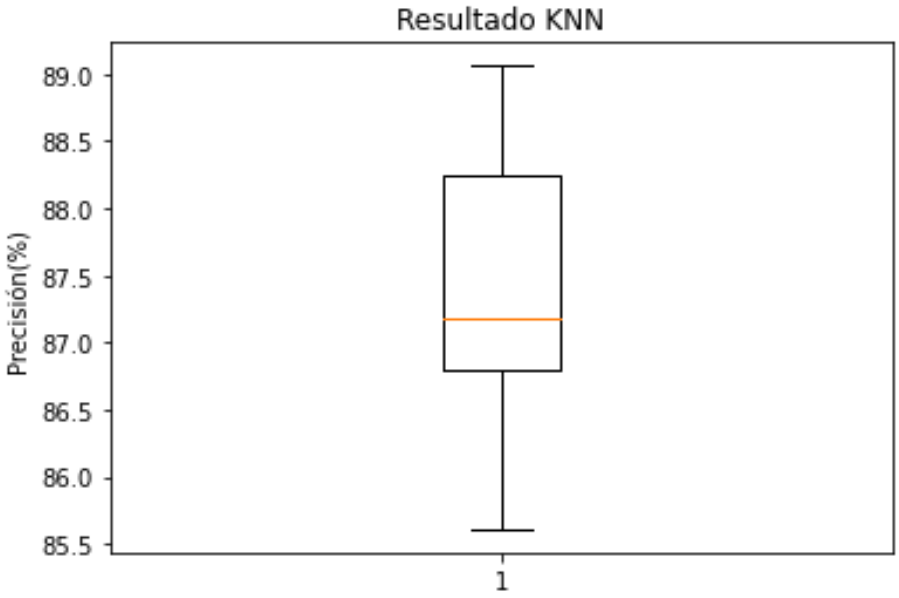

Figura 23. Resultado KNN con mejor configuración. Clasificación tipo de golpe.

La matriz de confusión para k=1 y función Manhattan se muestra en la Figura 24. Los mayores errores cometidos por el clasificador han sido confundir el revés con el globo de revés y el remate con la bandeja, ambos con 9 fallos.

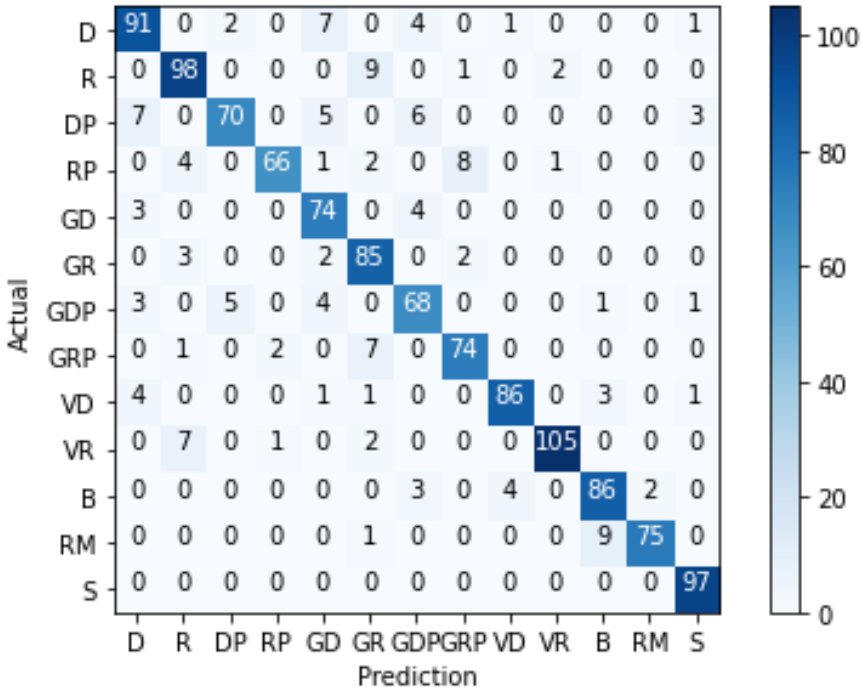

Figura 24. Algoritmo KNN para clasificación tipo de golpe. Matriz de confusión.

5.1.4 Máquina de Vector Soporte

Los resultados variando los parámetros de la Máquina de Vector Soporte, desarrollada en el apartado 4.3.5, se muestran en la Tabla 7. Hay que tener en cuenta que el hiperparámetro "probabilidad" estudiado previamente no afecta a la exactitud obtenida en los resultados, pero será necesaria para emplear el método soft voting del clasificador por votación. También, el hiperparámetro para la separación multiclase (decisión_fuction_shape) no produce ninguna variación en la exactitud alcanzada, por lo que no se muestra en la tabla.

Tabla 7. Exactitud (%) algoritmo SVM en función de los hiperparámetros. Clasificación tipo de golpe.

		C									
		0.1	1	2	5	8	10	12	15	20	50
Kernel	Lineal	57.57	55.02	53.21	52.63	52.63	52.63	52.63	52.63	52.63	52.63
	Polinómico	47.45	77.55	81.91	84.54	85.52	85.28	85.28	84.95	85.12	85.36
	Radial	47.78	84.29	87.34	90.54	90.38	90.54	90.38	89.97	89.97	89.64
	Sigmoide	24.10	22.04	17.85	15.87	14.80	14.56	14.22	13.57	13.98	14.14

La mejor configuración posible consiste en aplicar un valor de regulación de 5 o 10 junto a la función radial de Kernel, alcanzando un **90.54%** de aciertos. Por otro lado, los peores resultados se muestran al emplear la función sigmoide, cuya mejor exactitud es del 24.10% utilizando un valor de regulación de 0.1. Este valor se encuentra muy por debajo del mejor resultado, con una diferencia de 66.44%. Por último, cabe resaltar que la función lineal no muestra un cambio significativo al variar la regulación aplicada, manteniéndose alrededor del 54% todos sus resultados. Al igual que el algoritmo KNN, este resultado es constante para una misma división de datos. Al variar la partición de datos, la exactitud media obtenida es del **88.398% (±1.388)**.

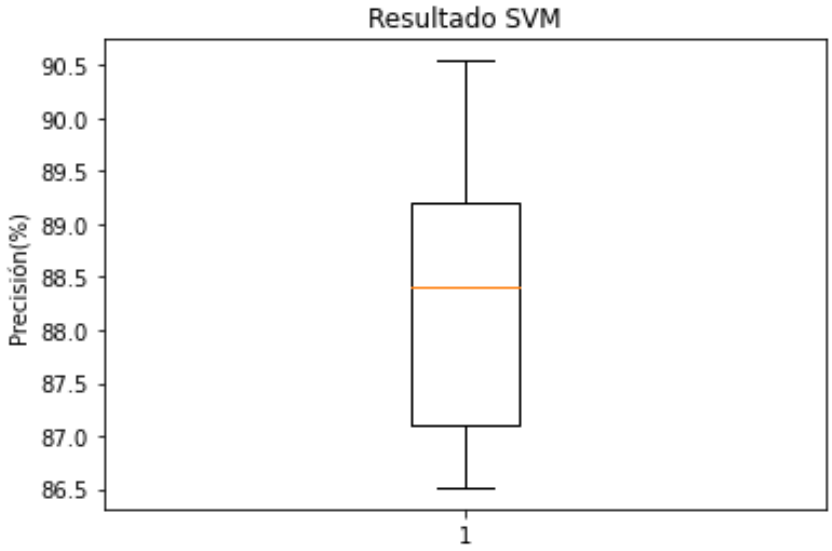

Figura 25. Resultado SVM con mejor configuración. Clasificación tipo de golpe.

La matriz de confusión para c=5 y la función radial se puede ver en la Figura 26. En este caso, el golpe con mayor precisión es la volea de revés. Los mayores errores se producen al etiquetar el revés como globo de revés, el revés de pared como revés y la derecha como globo de derecha. Estos errores se producen ya que son golpes parecidos.

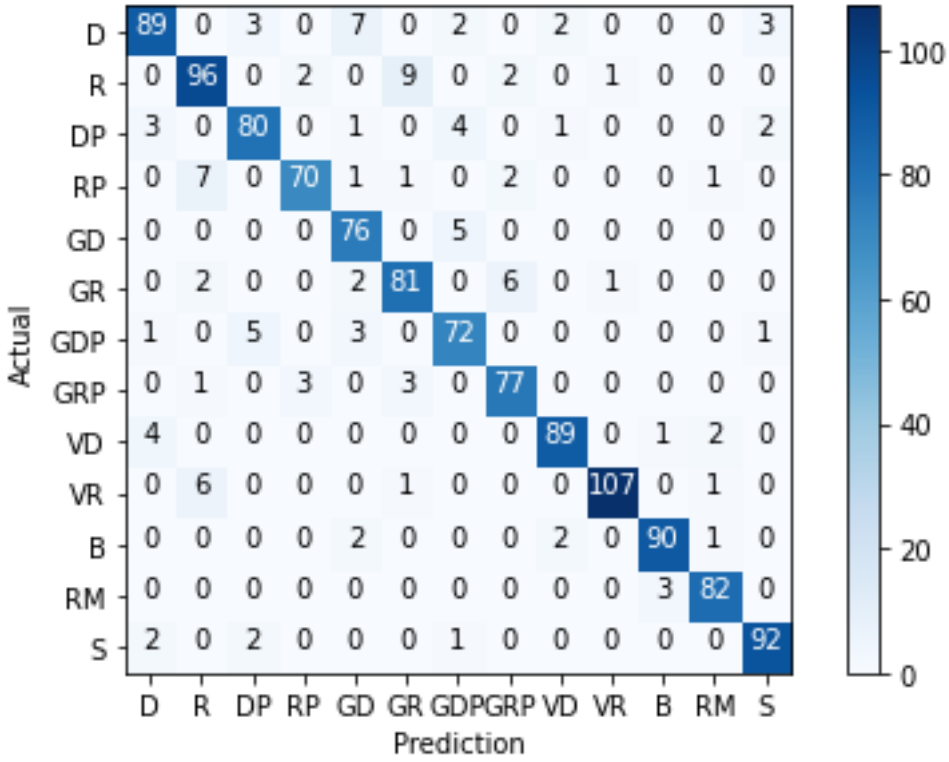

Figura 26. Algoritmo SVM para clasificación tipo de golpe. Matriz de confusión.

5.1.5 Bagging

Para el método de Bagging, entrenaremos tres algoritmos débiles como estimadores base y compararemos la

diferencia entre aplicar este ensamble de clasificadores con entrenar únicamente el algoritmo base. Así, dividiremos los resultados en tres secciones.

Los hiperparámetros modificados para el clasificador Bagging se muestran en la Tabla 8. Para cada técnica Bagging utilizada se estudiará los mismos valores de hiperparámetros Se empleará como parámetros el número de estimadores utilizados para el ensamble, el número máximo de muestras con las que se entrena cada algoritmo base y el número máximo de características tomadas por cada algoritmo base.

Tabla 8. Hiperparámetros Bagging.

N_estimators	10, 50, 100, 250, 500
Max_samples	100, 250, 500, 750, 1000
Max_features	25. 50, 100, 200

5.1.5.1 Estimador base: árbol de decisión

La mejor combinación de hiperparámetros, logrando una exactitud del **83.840% (±0.340)**, se alcanza utilizando 500 estimadores bases, 1000 muestras por estimador y 25 características por estimador.

Figura 27. Resultado Bagging con árbol de decisión utilizando la mejor configuración. Clasificación tipo de golpe.

Los errores cometidos por este clasificador se pueden observar en la Figura 28. La matriz de confusión, para este caso, presenta los mismos tipos de errores que los clasificadores anteriores. Estos son errores cometidos al clasificar golpes con movimientos similares, además de mostrar la tendencia de confundir golpes de pádel realizados desde el mismo lado. Cabe destacar los 12 fallos cometidos al clasificar el revés como globo de revés y el revés después de pared con el revés.

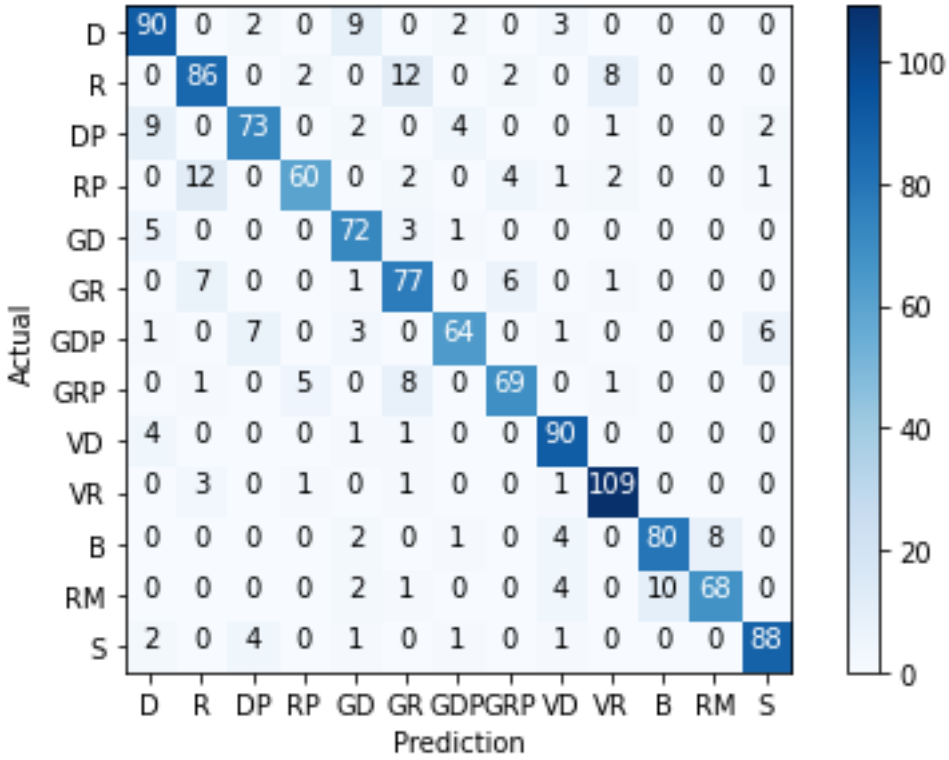

Figura 28. Algoritmo Bagging y clasificador base árbol de decisión para clasificación tipo de golpe. Matriz de confusión.

5.1.5.2 Estimador base: K vecinos más próximos

Al utilizar como estimador base el algoritmo KNN para el método de ensamble Bagging, se obtiene una exactitud del **89.391% (± 0.169)**. Para obtener esta exactitud se ha empleado los siguientes valores de los hiperparámetros: 500 estimadores bases, 1000 muestras por estimador y 100 características de entrada por clasificador.

Figura 29. Resultado Bagging con KNN utilizando la mejor configuración. Clasificación tipo de golpe.

En la Figura 30 se presenta la matriz de confusión aplicando dicha combinación de hiperparámetros. En cuanto a los errores, cabe resaltar los 11 fallos efectuados al clasificar el remate como bandeja, y los 10 fallos cometidos al catalogar el globo de revés con pared como globo de revés y el revés como globo de revés.

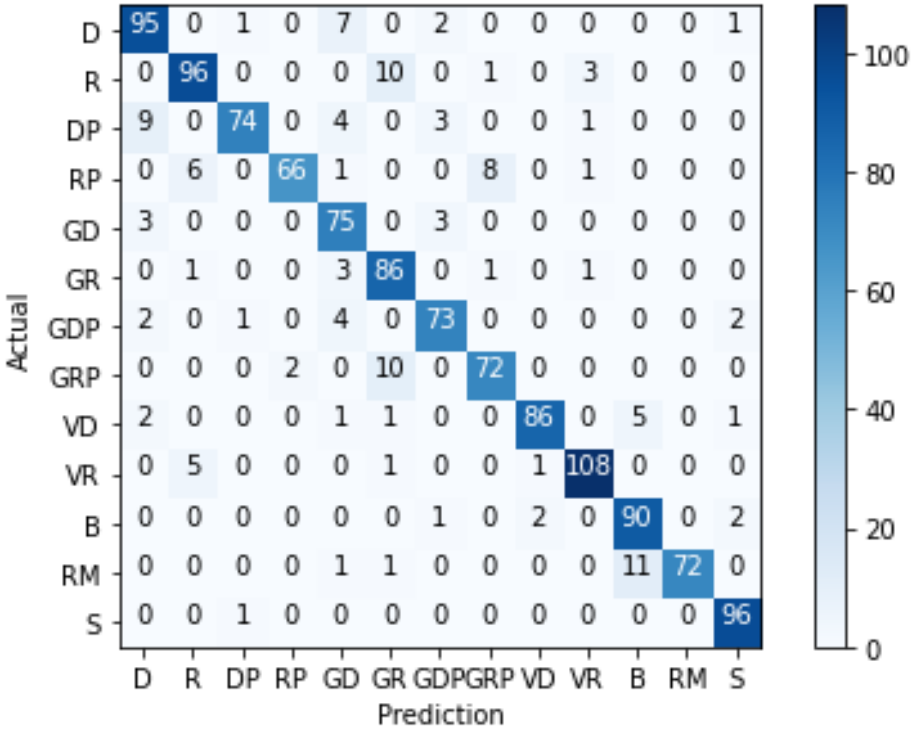

Figura 30. Algoritmo Bagging y clasificador base KNN para clasificación tipo de golpe. Matriz de confusión.

5.1.5.3 Estimador base: Máquina de Vector Soporte

Los mejores parámetros para este modelo son 500 estimadores bases, 1000 muestras por clasificador base y 200 características por clasificador. Aplicando estos valores se obtiene un **85.905% (±0.242)** de aciertos.

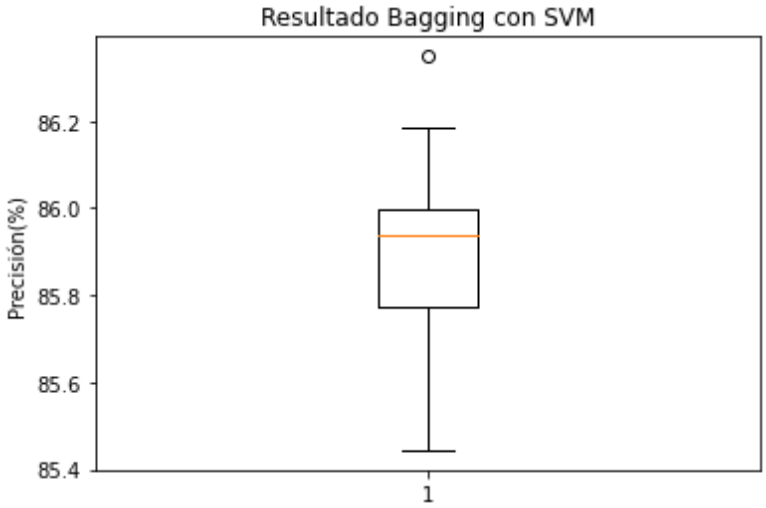

Figura 31. Resultado Bagging con SVM utilizando la mejor configuración. Clasificación tipo de golpe.

La matriz de confusión se puede observar en la Figura 32, donde se muestran los aciertos y fallos del clasificador. En este método, se ha clasificado erróneamente el revés con pared como revés 9 veces.

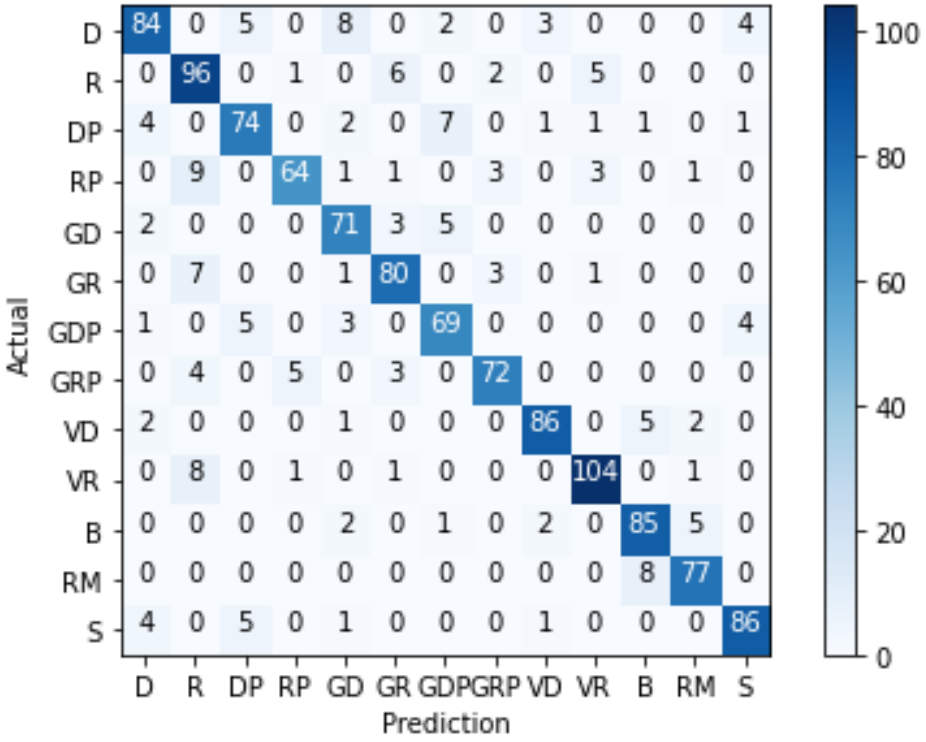

Figura 32. Algoritmo Bagging y clasificador base SVM para clasificación tipo de golpe. Matriz de confusión.

5.1.5.4 Resumen Bagging

A continuación, mostraremos una comparación entre los resultados obtenidos aplicando el método Bagging con el algoritmo base empleado. Como podemos observar en la Tabla 9, destaca la mejora del algoritmo árbol de decisión aplicando Bagging, aumentado su exactitud en casi un 30%. El algoritmo KNN muestra cierta mejora al emplear Bagging, aumentando la exactitud en un 2%. Por otro lado, el clasificador SVM disminuye su exactitud en casi un 3%.

Tabla 9. Exactitud (%) algoritmo Bagging en función del algoritmo base. Clasificación tipo de golpe.

	Clasificador base	Bagging
Árbol de decisión	54.09	83.84
KNN	87.42	89.39
SVM	88.40	85.905

5.1.6 Pasting

Al igual que en el método de Bagging, entrenaremos tres algoritmos débiles como estimadores bases y compararemos la diferencia entre aplicar este ensamble de clasificadores con entrenar únicamente el algoritmo base. Dividiremos los resultados en función del estimador base utilizado.

En cuanto a los hiperparámetros empleados, el algoritmo Pasting no permite utilizar el número de muestras por estimador, por lo que se estudiarán el número de estimadores y el número de características.

5.1.6.1 Estimador base: árbol de decisión

Las exactitudes obtenidas al variar los parámetros se presentan en la Tabla 10.

Tabla 10. Exactitud (%) algoritmo Pasting con árbol de decisión en función de los hiperparámetros. Clasificación tipo de golpe.

		N.º Estimadores			
		10	100	250	500
N.º características	25	75.58	87.75	89.06	89.14
	50	77.88	88.08	88.57	88.82
	100	78.13	86.10	85.94	86.51
	200	64.39	73.03	73.85	73.10

Como podemos observar, el mejor resultado se obtiene al aplicar 500 estimadores bases con 25 características por estimador. La accuracy aumenta hasta alcanzar los 100-250 estimadores bases, donde se estabiliza. Por lo tanto, al utilizar 250 se obtiene una accuracy similar empleando menos recursos, entrenando en menos tiempo el algoritmo. La exactitud media obtenida es del **89.032% (±0.281).**

Figura 33. Resultado Pasting con árbol de decisión utilizando la mejor configuración. Clasificación tipo de golpe.

Los errores cometidos por el algoritmo se pueden observar en la Figura 34, donde se presenta la matriz de confusión. Como podemos apreciar, se mantiene la tendencia de los errores cometidos por los anteriores algoritmos. Destaca los 10 fallos al clasificar el revés como globo de revés.

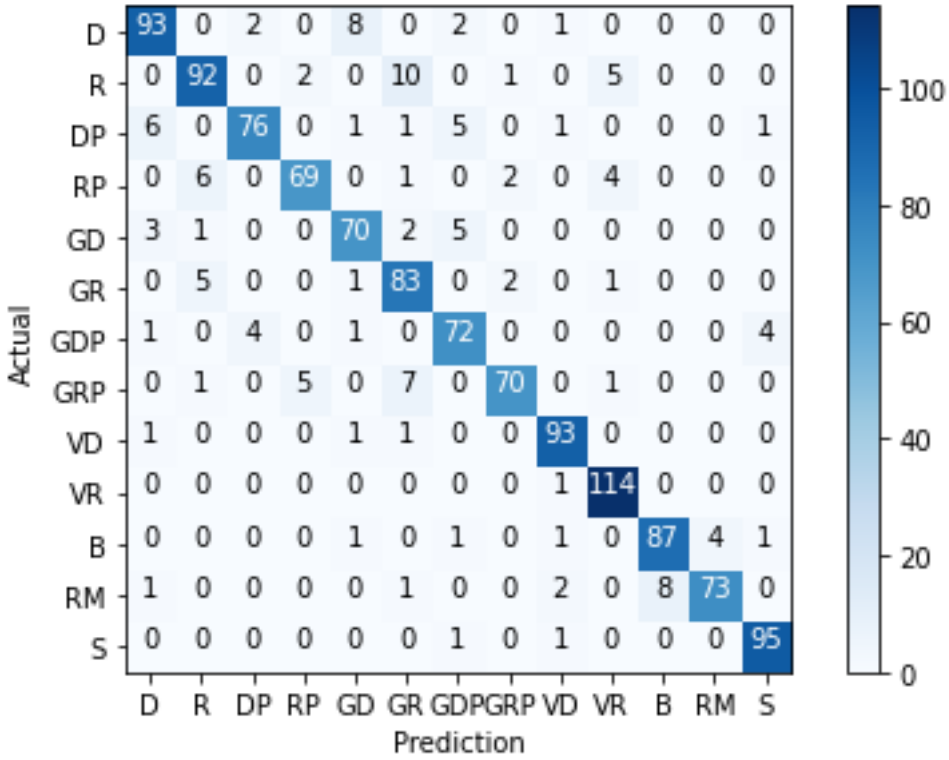

Figura 34. Algoritmo Pasting y clasificador base árbol de decisión para clasificación tipo de golpe. Matriz de confusión.

5.1.6.2 Estimador base: K vecinos más próximos

Utilizando como estimador base el algoritmo KNN, se obtienen los siguientes resultados, Tabla 11, al variar los hiperparámetros.

Tabla 11. Exactitud (%) algoritmo Pasting con KNN en función de los hiperparámetros. Clasificación tipo de golpe.

		N.º Estimadores			
		10	100	250	500
N.º características	25	88.08	90.54	90.38	90.46
	50	88.32	90.21	90.54	90.63
	100	87.99	89.64	90.13	90.05
	200	88.32	88.82	88.56	88.57

La mejor combinación de hiperparámetros es utilizar 500 estimadores bases con 50 características por estimador. Sin embargo, podemos observar que al utilizar 100 y 250 estimadores bases obtenemos un resultado similar empleando menos recursos. Se alcanza una accuracy media del **90.543% (± 0.224)** con dichos valores.

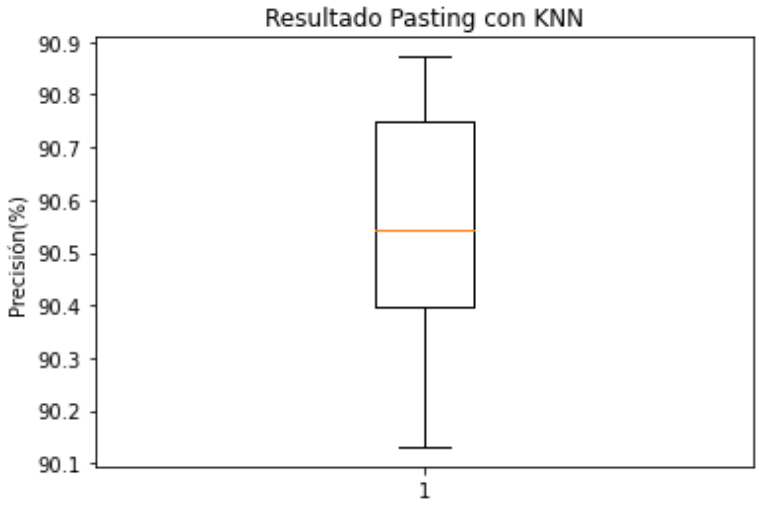

Figura 35. Resultado Pasting con KNN utilizando la mejor configuración. Clasificación tipo de golpe.

La matriz de confusión para este algoritmo se muestra en la Figura 36. Cabe destacar los 10 errores cometidos al confundir el revés como globo de revés y 9 fallos al confundir el remate con la bandeja.

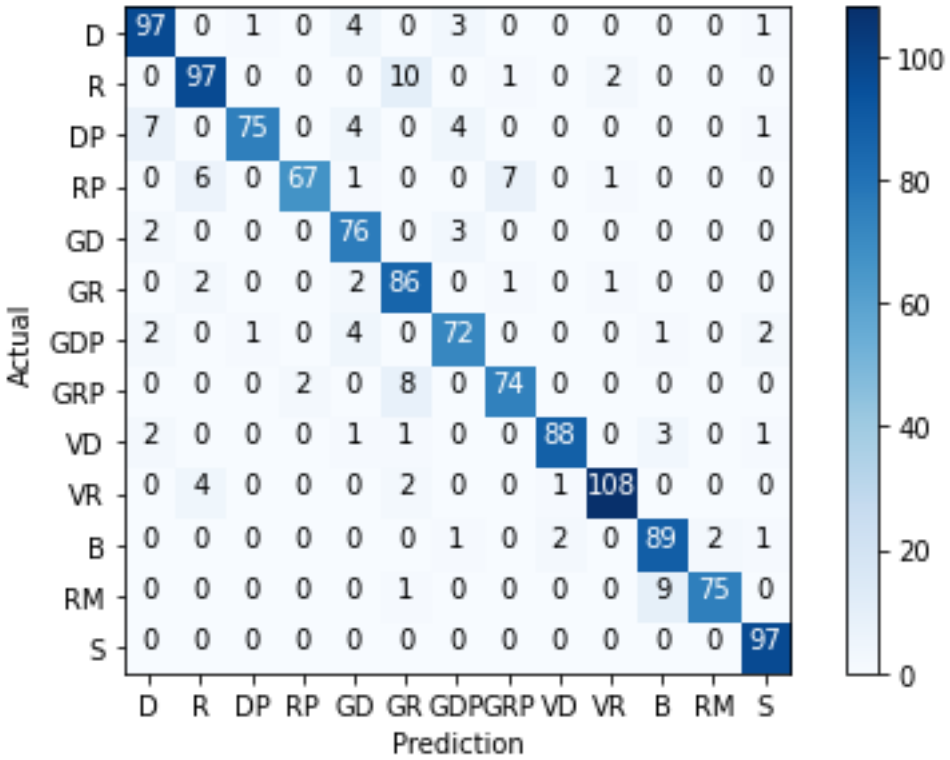

Figura 36. Algoritmo Pasting y clasificador base KNN para clasificación tipo de golpe. Matriz de confusión.

5.1.6.3 Estimador base: máquina de vector soporte

Al igual que en los estimadores bases anteriores, en la Tabla 12 se presenta los resultados obtenidos al variar los valores de los hiperparámetros.

Tabla 12. Exactitud (%) algoritmo Pasting con SVM en función de los hiperparámetros. Clasificación tipo de golpe.

		N.º Estimadores			
		10	100	250	500
N.º características	25	87.99	88.98	88.65	89.06
	50	89.39	89.56	90.13	90.05
	100	89.72	90.625	90.46	90.38
	200	90.46	90.21	90.38	90.21

Utilizando como clasificador base el algoritmo SVM podemos observar como la exactitud se establece antes que los anteriores al utilizar un número menor de estimadores. El mejor resultado obtenido es del **90.625%**, con un valor medio **90.428% (±0.143)** aplicando 100 estimadores bases con 100 características por estimador.

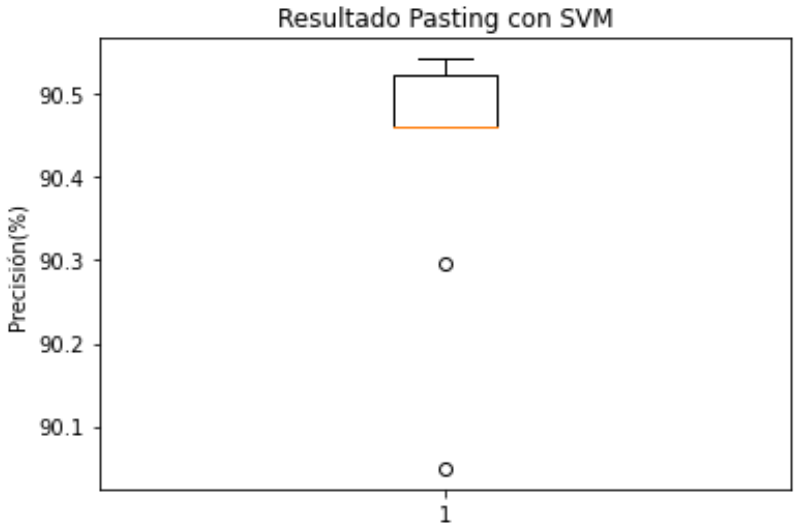

Figura 37. Resultado Pasting con SVM utilizando la mejor configuración. Clasificación tipo de golpe.

La matriz de confusión se puede observar en la Figura 38. En cuanto a los errores cometidos por el clasificador, destaca los 8 fallos al confundir el revés con el globo de revés, y 7 errores al clasificar la derecha como derecha con pared y el globo de revés con el globo de revés después de pared.

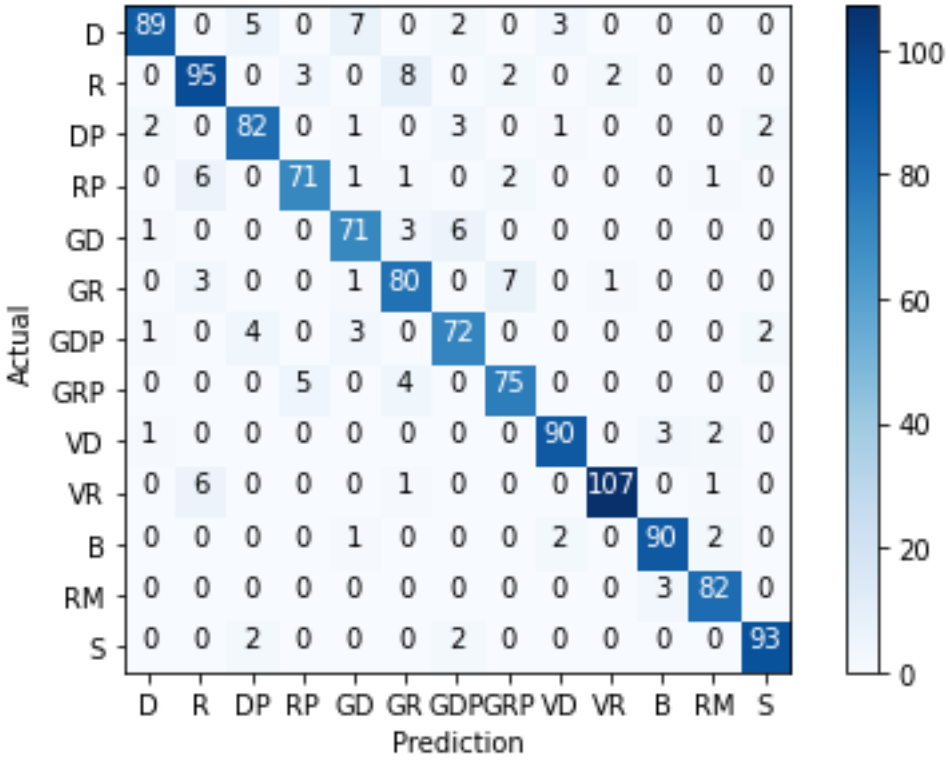

Figura 38. Algoritmo Pasting y clasificador base SVM para clasificación tipo de golpe. Matriz de confusión.

5.1.6.4 Resumen Pasting

Un resumen de los resultados obtenidos por el clasificador Pasting se puede ver en la Tabla 13. Se puede apreciar una gran diferencia entre los dos porcentajes del árbol de decisión, incrementando la exactitud en un 35%. Esta diferencia entre aplicar en el árbol de decisión y el ensamble es similar al del método Bagging. En cuanto al algoritmo KNN, también se observa un incremento de un 3% al utilizar el ensamble. Por otro lado, la máquina de vector soporte no ofrece mucha variación, aumentando su exactitud en un 2%.

Tabla 13. Exactitud (%) algoritmo Pasting en función del algoritmo base. Clasificación tipo de golpe

	Clasificador base	Pasting
Árbol de decisión	54.09	89.03
KNN	87.42	90.54
SVM	88.40	90.43

5.1.7 Random Forest

Para el clasificador Random Forest explicado en el apartado 4.4.3, se modificarán tres parámetros. Estos parámetros son el número de estimadores, el número de muestras y el método de entrenamiento (Bagging o Pasting). Debido a que el método Pasting no utiliza el hiperparámetro número de muestras, se utiliza el valor 'none' para referenciar que no se emplea este hiperparámetro. En cuanto a los hiperparámetros correspondientes al árbol de decisión, se utilizará la combinación que mejor resultado alcanzó. Los valores se muestran en la Tabla 14.

Tabla 14. Hiperparámetros Random forest.

N.º de estimadores	10, 50, 100, 250, 500
N.º de muestras	None, 100, 250, 500, 750, 1000
Bootstrap	True, False

Este clasificador obtiene como mejor resultado una exactitud del **88.495% (±0.422)**, utilizando los siguientes hiperparámetros: 500 estimadores bases y método pasting.

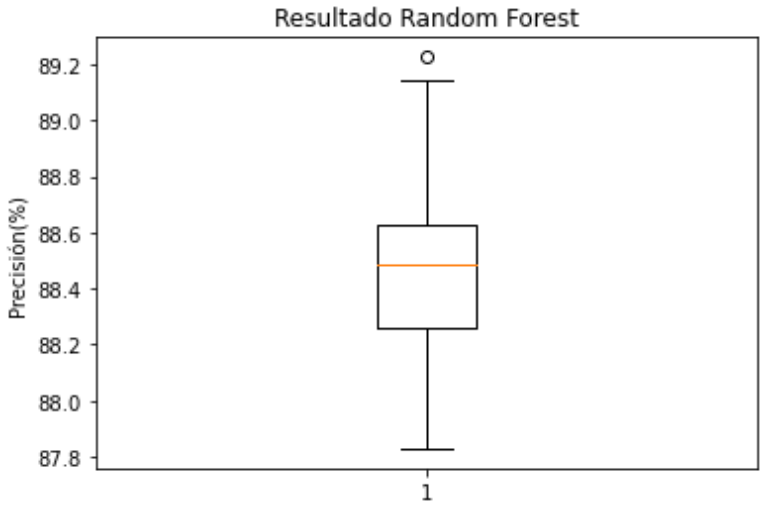

Figura 39. Resultado Random Forest con mejor configuración. Clasificación tipo de golpe.

Los errores cometidos por el clasificador se presentan en la Figura 40. La matriz de confusión nos muestra tres errores destacables: 12 fallos al clasificar el revés con el globo de revés y 8 fallos al confundir el revés después de pared con el revés.

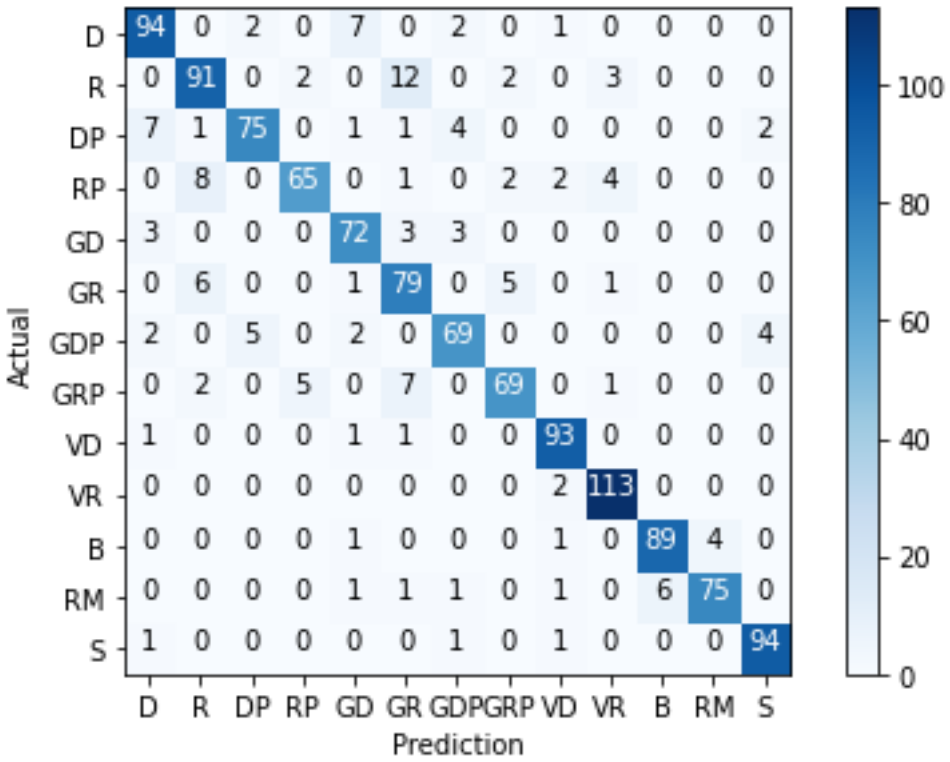

Figura 40. Algoritmo Random Forest para clasificación tipo de golpe. Matriz de confusión.

5.1.7.1 Extremely Randomized Trees

Este método de ensamble de clasificadores tiene los mismos hiperparámetros que el algoritmo Random Forest presentando anteriormente en la Tabla 14.

Los resultados nos muestran un comportamiento similar al algoritmo Random Forest. El mejor resultado se obtiene al aplicar 500 estimadores bases y método Pasting, alcanzado una exactitud del **89.416% (±0.275).**

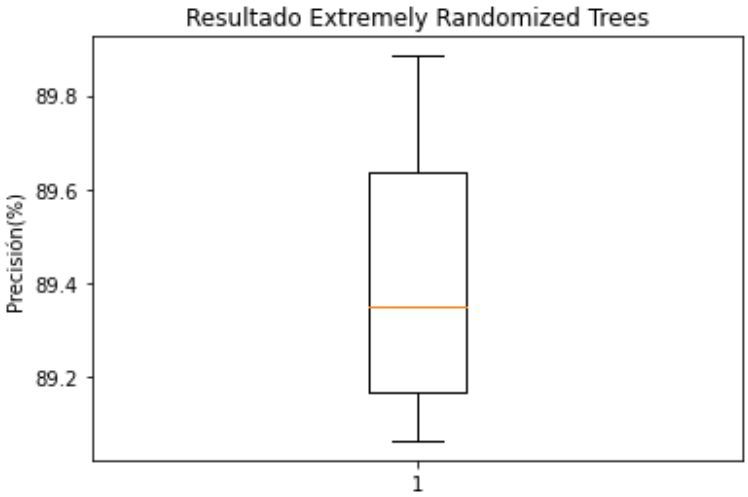

Figura 41. Resultado Extremely Randomized Trees con mejor configuración. Clasificación tipo de golpe.

La matriz de confusión se expone en la Figura 42. El patrón de los errores es igual que los algoritmos anteriores, es decir, este clasificador se confunde al clasificar tipos de golpes con una mecánica parecida. Sobresale los 11 fallos cometidos al clasificar el remate con la bandeja.

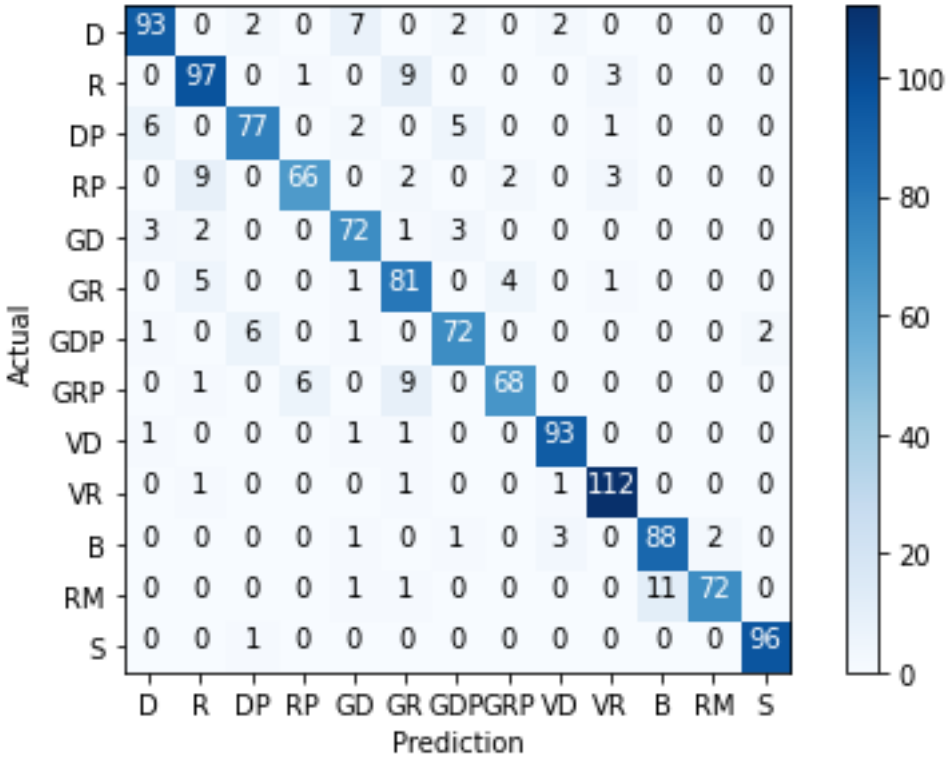

Figura 42. Algoritmo Extremely Randomized Forest para clasificación tipo de golpe. Matriz de confusión.

5.1.7.2 Resumen Random Forest

En la Tabla 15 podemos observar las diferentes exactitudes al utilizar los algoritmos árbol de decisión, Random Forest y Extremely Randomized Trees. El mejor resultado lo alcanza este último, con un 89.42% de aciertos. Esto supone una mejora de 35% con respecto al árbol de decisión utilizado como base.

Tabla 15 Exactitud (%) algoritmo árbol de decisión, Randon Forest y Extremely Randomized Trees. Clasificación tipo de golpe.

Árbol de decisión	Random Forest	Extremely Randomized Trees
54.09	88.50	89.42

5.1.8 AdaBoost

El método AdaBoost posee tres hiperparámetros importantes a modificar. En primer lugar, para el clasificador base se utilizarán los algoritmos árbol de decisión y SVM. No se utilizará el algoritmo KNN, ya que este algoritmo no proporciona pesos a las muestras, siendo imposible de implementarlo en AdaBoost. En cuanto al número de estimadores y learning rate, se probarán una serie de valores para obtener los mejores valores.

5.1.8.1 Estimador base: árbol de decisión

Los resultados obtenidos al variar los valores de los parámetros utilizando como estimador base el árbol de decisión se muestran en la Tabla 16.

Tabla 16. Exactitud (%) algoritmo AdaBoost con árbol de decisión en función de los hiperparámetros. Clasificación tipo de golpe.

		Learning rate				
		0.01	0.05	0.1	0.5	1
N.º estimadores	50	68.58	83.55	85.69	85.53	84.62
	100	76.89	85.03	86.35	87.58	86.10
	250	84.38	87.25	87.99	88.40	88.90
	500	85.94	87.58	86.68	87.01	89.31

La mejor accuracy alcanzada es de un **89.31%** de aciertos. Para ello, se ha empleado 500 estimadores bases y un coeficiente de aprendizaje de 1. Se puede observar cómo aumenta la exactitud obtenida al incrementar el número de estimadores bases. La exactitud media obtenida es del **88.832% (±0.408)**.

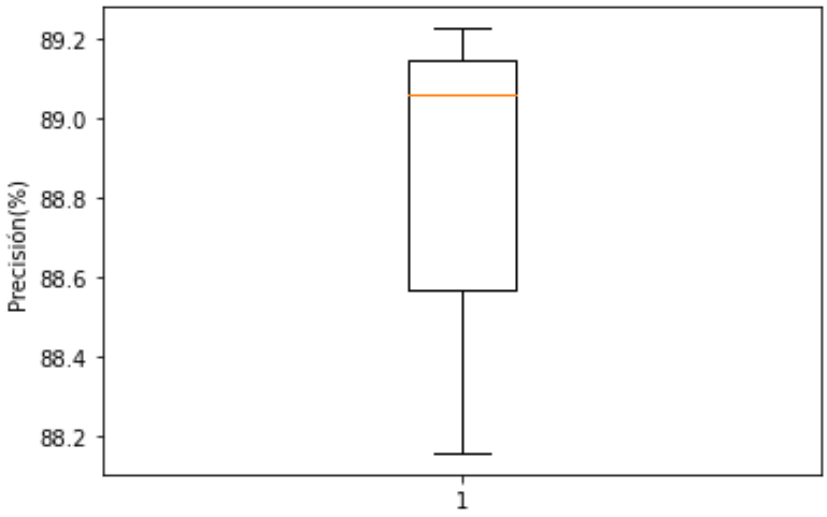

Figura 43. Resultado AdaBoost con árbol de decisión utilizando la mejor configuración. Clasificación tipo de golpe.

En la Figura 44 se muestran los errores cometidos por el clasificador. Entre los errores, sobresale los 10 fallos cometidos al clasificar el revés después de pared como revés y el revés como globo de revés, y 9 fallos al confundir el remate con la bandeja.

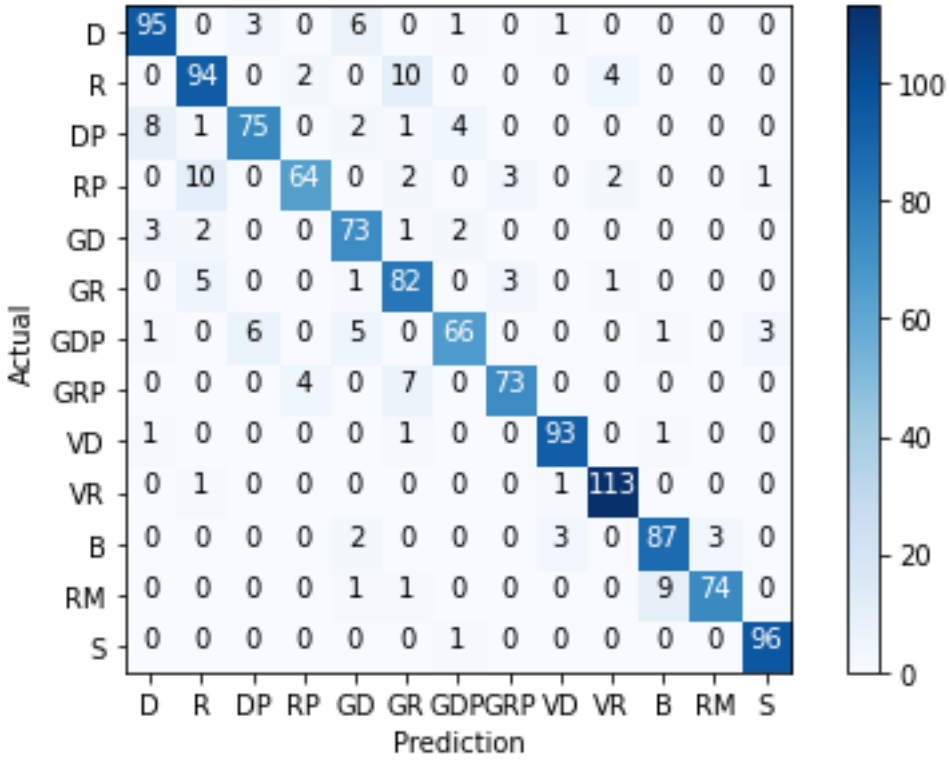

Figura 44. Algoritmo AdaBoost y clasificador base árbol de decisión para clasificación tipo de golpe. Matriz de confusión.

5.1.8.2 Estimador base: Máquina de vector soporte

En la Tabla 17 se presentan los resultados variando los hiperparámetros del método AdaBoost utilizando como estimador base la máquina de vector soporte.

Tabla 17. Exactitud (%) algoritmo AdaBoost con SVM en función de los hiperparámetros. Clasificación tipo de golpe.

		Learning rate				
		0.01	0.05	0.1	0.5	1
N.º estimadores	50	59.13	76.48	78.20	50.74	41.53
	100	69.82	77.30	76.56	40.95	26.64
	250	79.44	75.99	62.17	25.99	24.51
	500	79.35	61.35	59.54	20.88	16.69

La mejor configuración para este clasificar resulta en aplicar un coeficiente de aprendizaje 0.01 y 250 estimadores bases, alcanzado una accuracy del **79.44%**. La accuracy media obtenida que aplica dicha combinación es del **78.816% (±0.841)**.

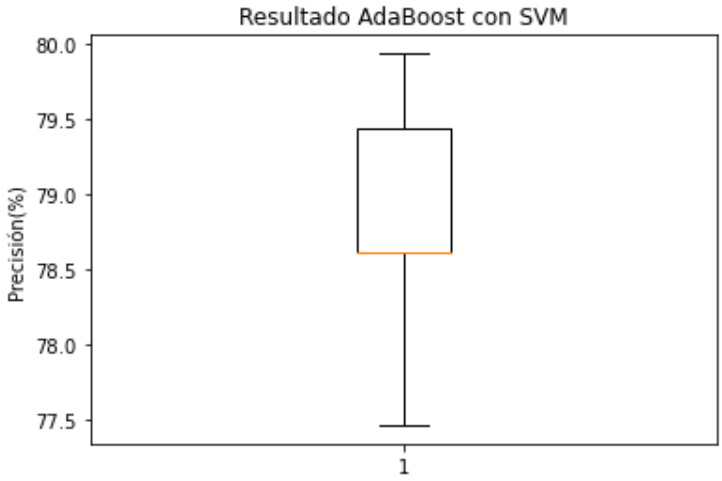

Figura 45. Resultado AdaBoost con SVM utilizando la mejor configuración. Clasificación tipo de golpe.

La matriz de confusión para dichos valores se puede ver en la Figura 46. Los mayores errores cometidos por el clasificador son 14 fallos al clasificar la derecha después de pared con el globo de derecha después de pared, así como los 13 fallos al confundir el globo de revés con el globo de revés después de pared.

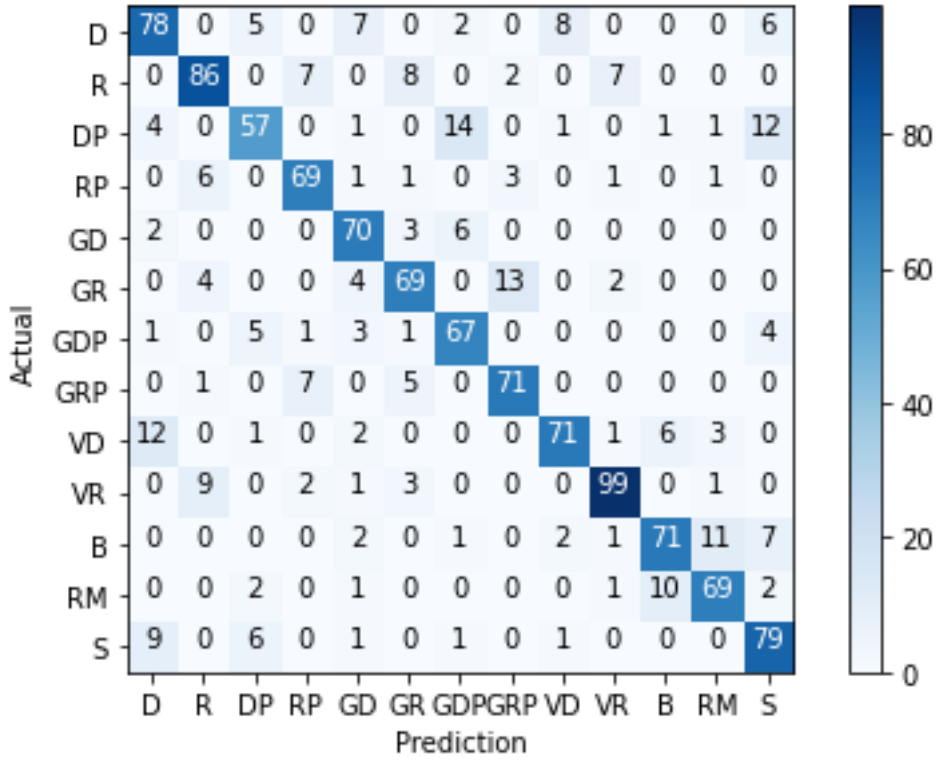

Figura 46. Algoritmo AdaBoost y clasificador SVM para clasificación tipo de golpe. Matriz de confusión.

5.1.8.3 Resumen AdaBoost

En la Tabla 18 se presenta una comparación entre el método de ensamble AdaBoost y los clasificadores bases utilizados. Se aprecia un incremento de casi 35% al utilizar AdaBoost en el árbol de decisión. Por otro lado, la máquina de vector soporte reduce su exactitud en más de un 10% al utilizar el ensamble.

Tabla 18. Exactitud (%) algoritmo AdaBoost en función del algoritmo base. Clasificación tipo de golpe.

	Clasificador base	Pasting
Árbol de decisión	54.09	88.83
SVM	88.40	78.82

5.1.9 Gradient Boosting

Los hiperparámetros modificados para el algoritmo Gradient Boosting, explicados en el apartado 4.4.5, son el número de estimadores, el coeficiente de aprendizaje y los parámetros correspondientes al árbol de decisión. Para los parámetros correspondientes al árbol de decisión, se utilizarán los mismos. En la Tabla 19 se presenta los resultados obtenidos variando los dos hiperparámetros.

Tabla 19. Exactitud (%) algoritmo Gradient Boosting en función de los hiperparámetros. Clasificación tipo de golpe.

		Learning rate				
		0.01	0.05	0.1	0.5	1
N.º estimadores	50	59.13	62.66	62.91	74.67	66.86
	100	60.53	64.14	65.46	75.33	67.52
	250	62.42	65.79	65.05	74.26	67.19
	500	62.34	65.54	66.61	76.32	68.01

Como podemos observar, la exactitud máxima alcanzada es del 76.46%, al emplear 500 estimadores bases y un coeficiente de aprendizaje de 0.5. Se observa una pequeña mejora al aumentar el número de estimadores bases, al igual que en los anteriores clasificadores. La exactitud media obtenida que aplica esta configuración es del **75.395% (±0.798)**.

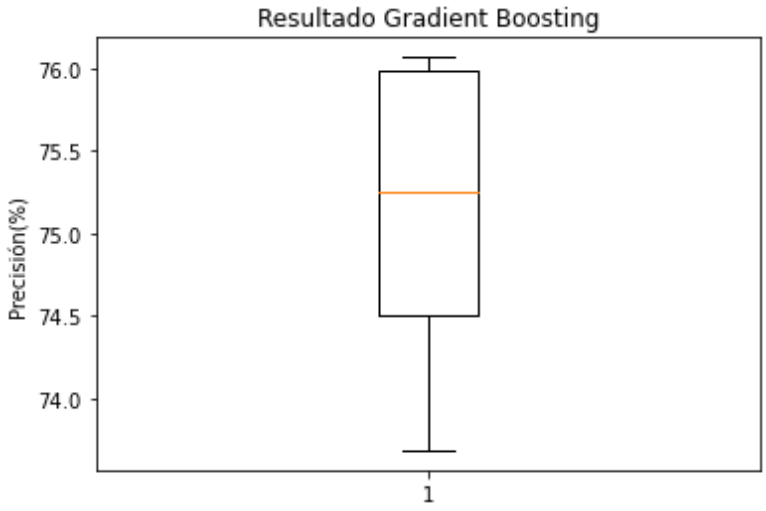

Figura 47. Resultado Gradient Boosting utilizando la mejor configuración. Clasificación tipo de golpe.

La matriz de confusión presentada en la Figura 48 nos muestra los errores cometidos por el clasificador. Se destacan los 15 fallos cometidos al confundir el revés con el globo de revés, y los 11 fallos al confundir el globo de revés con el revés, así como la bandeja con el remate.

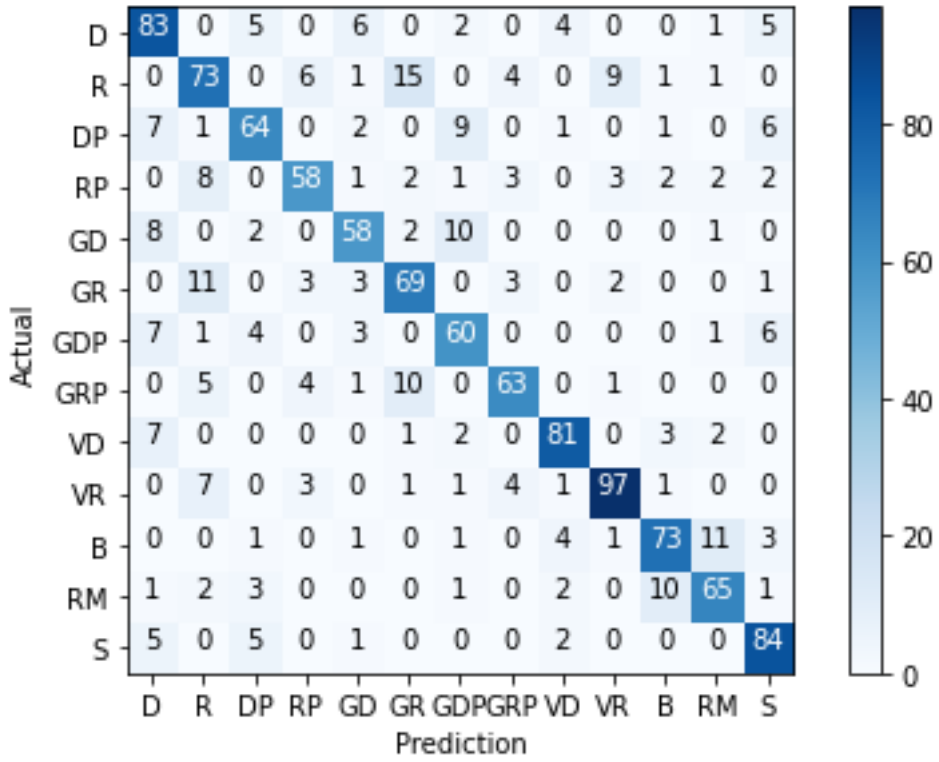

Figura 48. Algoritmo Gradient Boosting para clasificación tipo de golpe. Matriz de confusión.

5.1.10 Clasificador por votación

Por último, estudiaremos los resultados producidos por el algoritmo de clasificación por votación. Uno de los parámetros a modificar por este clasificador son los estimadores. En este hiperparámetro podremos utilizar todos los algoritmos desarrollados previamente, incluso los algoritmos de ensamble de clasificadores. Sin embargo, como se explicó en el apartado 4.4.1, este algoritmo funciona mejor al utilizar algoritmos independientes entre sí. Por lo tanto, se entrenarán los cuatros algoritmos bases desarrollados (MLP, árbol de decisión, KNN, SVM). Se estudiarán los resultados obtenidos tanto para el método hard voting como soft voting, variando el peso asignado a cada clasificador base entre 1 y 3.

5.1.10.1 Hard voting

La búsqueda en rejilla nos indica que la mejor combinación de pesos asignados a cada clasificador es: 3 para los estimadores MLP, KNN, SVM, y 1 para el árbol de decisión. La exactitud media obtenida es del **91.990% (±0.084)**.

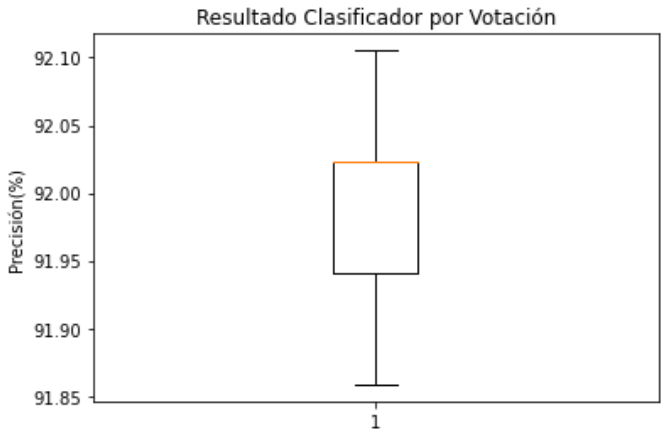

Figura 49. Resultado clasificador por votación con hard voting utilizando la mejor configuración. Clasificación tipo de golpe.

La matriz de confusión se puede observar en la Figura 50. En esta figura los errores cometidos son menores, ya que tiene una mejor precisión. El error máximo cometido es de 7 fallos al clasificar el revés como globo de revés. Se aprecia la misma tendencia de confundir golpes con una mecánica similar.

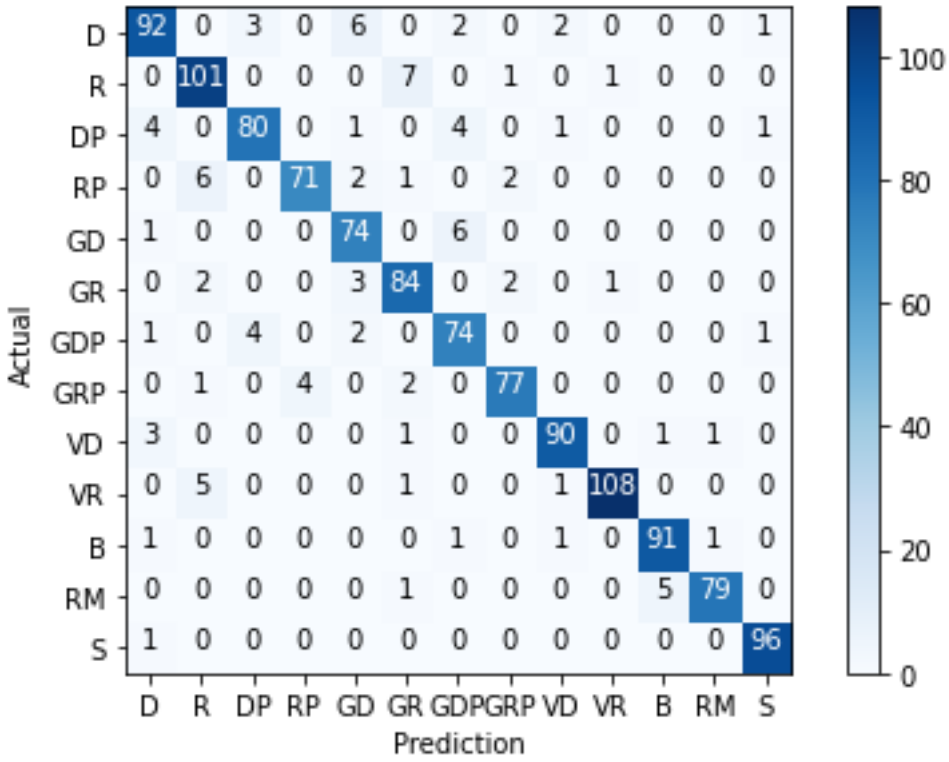

Figura 50. Algoritmo clasificador por votación con hard voting para clasificación tipo de golpe. Matriz de confusión.

5.1.10.2 Soft voting

En cuanto al método soft voting, la configuración de los pesos es igual que el anterior, es decir, 3 para los estimadores MLP, KNN, SVM, y 1 para el árbol de decisión. La exactitud media obtenida es del **92.451% (±0.176)**.

Figura 51. Resultado clasificador por votación con soft voting utilizando la mejor configuración. Clasificación tipo de golpe.

En la Figura 52 se muestra la matriz de confusión. Sobresale los 7 errores cometidos al confundir el revés con el globo de revés y el remate con la bandeja.

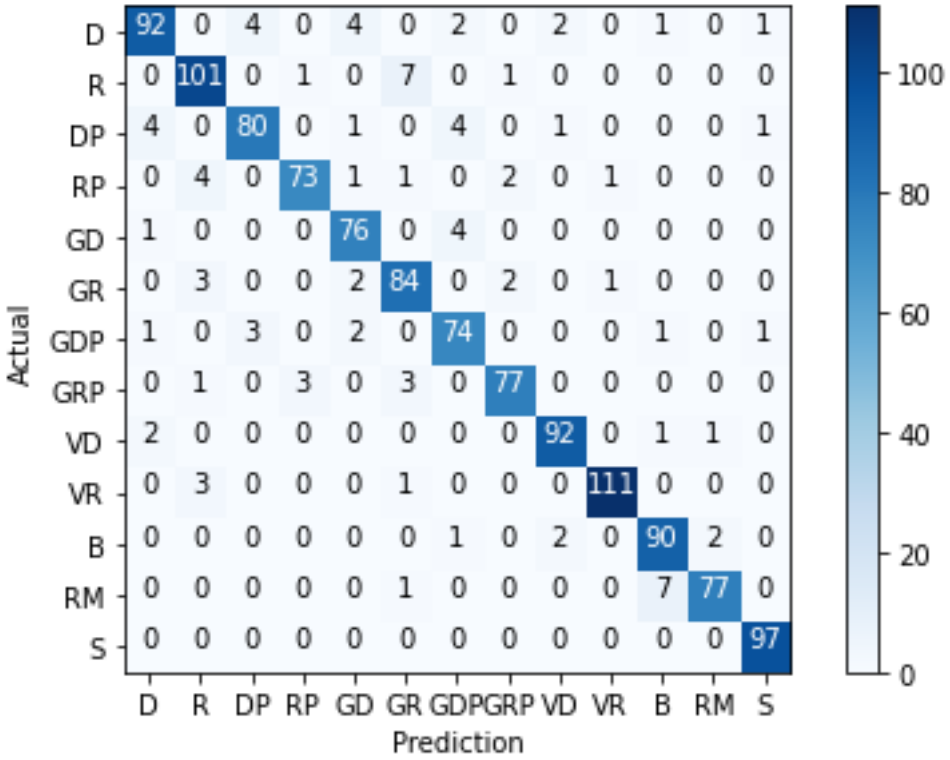

Figura 52. Algoritmo clasificador por votación con soft voting para clasificación tipo de golpe. Matriz de confusión.

5.1.11 Comparación métodos de ensamble

En la Tabla 20 se presenta una comparación de la exactitud y el tiempo empleado para cada ensamble desarrollado. En esta tabla, se muestra la exactitud media, exactitud máxima, la mejora de exactitud (en comparación al estimador base utilizado), el tiempo de entrenamiento y el tiempo de predicción empleado en cada algoritmo. Se debe aclarar que la mejora de exactitud en el caso del clasificador por votación se realizará en comparación con el estimador base que mayor precisión obtenga.

Tabla 20. Comparación métodos de ensamble

		Exactitud media (%)	Exactitud máxima (%)	Mejora exactitud (%)	Tiempo entrenamiento (s)	Tiempo predicción (s)
Bagging	Árbol de decisión	83.840	84.375	29.753	7.67	0.33
	KNN	89.391	89.638	1.973	2.75	9.61
	SVM	85.905	86.349	-2.493	82.59	55.49
Pasting	Árbol de decisión	89.032	89.556	34.945	33.26	0.81
	KNN	90.543	90.872	3.125	4.53	15.56
	SVM	90.428	90.625	2.030	60.10	14.20
Random Forest		88.495	89.227	34.408	16.79	0.23
Extremely Randomized Trees		89.416	89.885	35.329	1.51	0.20
AdaBoost	Árbol de decisión	88.832	89.227	34.745	1654.54	0.64
	SVM	78.816	79.934	-9.582	2107.78	180.86
Gradient Boosting		75.395	76.727	21.308	382.56	0.40
Clasificador por votación	Hard voting	91.990	92.105	2.401	126.74	1.11
	Soft voting	92.451	92.763	2.862	127.96	0.94

Atendiendo a la exactitud obtenida y al tiempo de entrenamiento y predicción requerido, el mejor clasificador es el **clasificador por votación** utilizando el método de soft voting. Este algoritmo logra una exactitud media de 92.451%, alcanzando un máximo de 92.763% de aciertos. Estas cifras son las máximas obtenidas entre todos los métodos de ensamble desarrollados. Para lograr dichos resultados, emplea un tiempo de entrenamiento de 127.97 segundos y un tiempo de predicción de 0.94 segundos. Al tener un tiempo de predicción menor de un segundo, este modelo puede ser implementado en sistemas de tiempo real para clasificar los golpes realizados durante un partido o un entrenamiento.

El segundo método con mejor resultado es el clasificador por votación utilizando el método de hard voting. Como era de esperar, este método obtiene una exactitud menor que el anterior, ya que únicamente utiliza las etiquetas de los estimadores bases en lugar de sus probabilidades. Aun así, el resultado que obtiene es de 91.99% de accuracy, casi 0.5% inferior al método de soft voting.

El algoritmo que emplea menos tiempo, tanto para el entrenamiento como para la predicción, es el Extremely Randomized Forest. Esta técnica dedica únicamente 1.51 segundos al entrenamiento del algoritmo y 0.20 segundos a la predicción. Además, este algoritmo presenta una alta exactitud, del 89.416, y es el mejor algoritmo en cuanto a la comparación con el estimador base utilizado. Por lo tanto, los ensambles Random Forest y Extremly Randomized Trees logran una mejora significativa en la precisión manteniendo tiempos de entrenamiento y predicción relativamente bajos. Esto los hace atractivos en términos de eficiencia y rendimiento.

En cuanto al tiempo empleado para el entrenamiento y predicción de los modelos, hay 2 aspectos destacables. En primer lugar, los tiempos de entrenamientos utilizados para los métodos de Boosting son considerablemente superiores al resto. Además, el tiempo requerido para el entrenamiento de los ensambles que utilizan como algoritmo base la Máquina de Vector Soporte es considerablemente superior a los otros dos. Al combinar estos dos aspectos, el algoritmo con el tiempo de entrenamiento más prolongado es el AdaBoost con SVM, con un periodo de entrenamiento de 35 minutos. Asimismo, el tiempo de predicción más extenso se observa en este ensamble, con 3 minutos empleados. Además, este algoritmo presenta la peor mejora de exactitud, reduciendo en un 9.582% la exactitud obtenida con respecto al estimador base.

Con respecto a la mejora de la exactitud frente al estimador base, destaca las mejoras obtenidas al emplear el árbol de decisión como estimador base. El árbol de decisión no es un algoritmo muy robusto, por lo que al incluirlo en los métodos de ensamble obtenemos un mejor rendimiento. Por otro lado, los algoritmos KNN y SVM son más complejos y robustos de manera individual, lo que resulta en mejoras no tan pronunciadas como las obtenidas con el árbol de decisión.

Cabe desatacar la diferencia obtenida en la exactitud entre los métodos de Bagging y Pasting. La única diferencia entre ambos algoritmos radica el uso de los datos. Como ya sabemos, el algoritmo Bagging reutiliza muestras de datos en el entrenamiento de los estimadores bases, mientras que el ensamble Pasting no lo permite. Esto provoca una gran diferencia en sus resultados. Empleando como estimador base el árbol de decisión, el algoritmo Pasting obtiene un 89.032% de aciertos, mientras que el método Bagging alcanza el valor del 83.840%, una diferencia de más del 5%. Esta disparidad también se observa al utilizar la Máquina de Vector Soporte como clasificador base, con una diferencia algo menor al 5%.

Como se mencionó en el apartado 4.3.4, el algoritmo KNN es un algoritmo de “aprendizaje perezoso”. Esto se representa en los tiempos de entrenamiento y predicción obtenidos al utilizar este modelo. Como podemos observar, los métodos de ensamble que utilizan este algoritmo como estimador base emplean más tiempo en la fase de predicción que en la fase de entrenamiento.

Por último, al analizar los errores de los clasificadores, se observa una tendencia a confundir golpes con un movimiento de brazos similares. Esto resulta evidente, dado que los datos captados por el sensor del brazo tienden a ser más parecidos. Por ejemplo, existe poca diferencia entre el revés y el revés con pared, siendo la única distinción que en un golpe la bola rebota antes con la pared. Además, la mayoría de los errores ocurren al clasificar golpes realizados desde el mismo lado, es decir, golpes de derecha se clasifican incorrectamente como otros golpes de derecha, al igual que el revés.

5.2 Clasificación nivel del jugador

En la clasificación del nivel del jugador, dado que no ha sido abordada en trabajos previos, nos enfocaremos en desarrollar modelos capaces de clasificar los datos de manera precisa y en revisar los errores cometidos. Por este motivo, solo utilizaremos los algoritmos de ensamble de clasificadores que han demostrado un mejor rendimiento en la clasificación previa.

En cada algoritmo entrenado, se mostrarán la accuracy obtenida, así como la matriz de confusión correspondiente. Además, para los algoritmos que muestren un mejor rendimiento, se presentará una tabla que incluirá el tipo de golpe y la identidad del jugador vinculados a los errores cometidos. Dado que el nivel del jugador puede estar influenciado por ambos tipos de datos, se realizará un análisis para comprender las razones detrás de estos errores. Para el tipo de golpe, se utilizará la misma nomenclatura que para la matriz de confusión. En cuanto a la identidad del jugador, se emplea el número de identidad asociado al jugador del conjunto de datos.

5.2.1 Perceptrón multicapa

El algoritmo perceptrón multicapa para la clasificación del nivel del jugador contará con una capa de entrada de 240 neuronas, al igual que el anterior, y una capa de salida de 5 neuronas, que contendrán las etiquetas correspondientes a los niveles de los jugadores.

Como realizamos en el mismo algoritmo previamente, dividiremos el estudio en dos partes, dependiendo del número de capas ocultas empleadas para la clasificación. Así, entrenaremos el modelo para una capa oculta y para dos capas ocultas.

5.2.1.1 Perceptrón multicapa con una capa oculta

Los hiperparámetros modificados se muestran en la Tabla 3. Estos son el número de neuronas por capa oculta, el tamaño de datos, la función de activación y el número de iteraciones.

El mejor resultado se alcanza empleando 1000 neuronas en la capa oculta, 70 iteraciones, tamaño de paquete de 10 datos y función de activación ReLU. Aplicando esta combinación de obtiene una exactitud del **91.990% (±0.323)**, que se alcanza en la iteración 70.

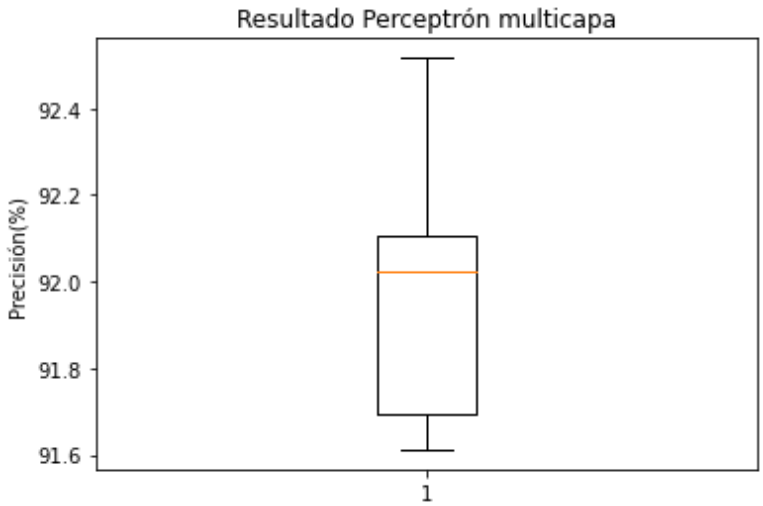

Figura 53. Resultado MLP con una capa oculta. Clasificación nivel del jugador.

La matriz de confusión se presenta en la Figura 54. Se puede apreciar que el mayor error cometido es confundiendo el nivel avanzado con el nivel intermedio, con 17 fallos. Estos resultados se pueden deber a que la diferencia entre el nivel intermedio y el nivel avanzado resulta de la opinión de los entrenadores, lo cual

puede ser más subjetivo. Además, la competición a la que pertenecen estos dos niveles es la misma, por lo que puede existir poca diferencia entre un nivel y el otro.

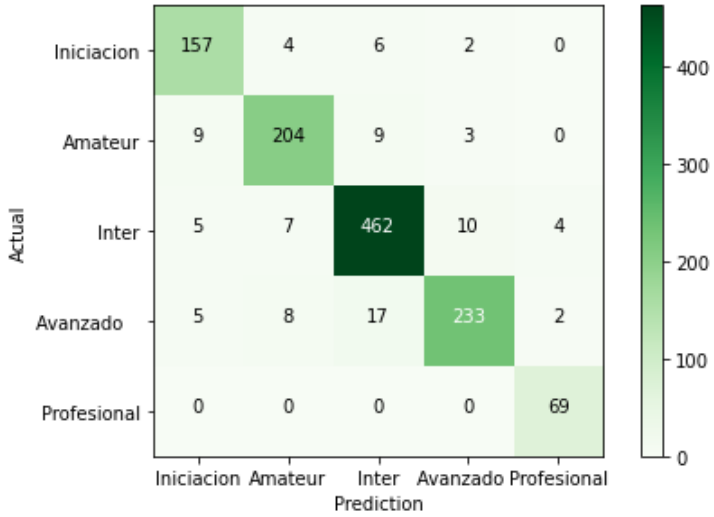

Figura 54. Algoritmo MLP con una capa oculta para clasificación nivel del jugador. Matriz de confusión.

5.2.1.2 Perceptrón multicapa con dos capas ocultas

Para el perceptrón multicapa se modificarán los mismos parámetros que en el apartado anterior, los cuales se presentan en la Tabla 4.

La mejor combinación de los hiperparámetros está compuesta por 500 neuronas la primera capa oculta, 250 la segunda, 70 iteraciones y un tamaño de paquete de 10 datos. Se alcanza una exactitud del **94.507% (±0.373)** en 28 iteraciones.

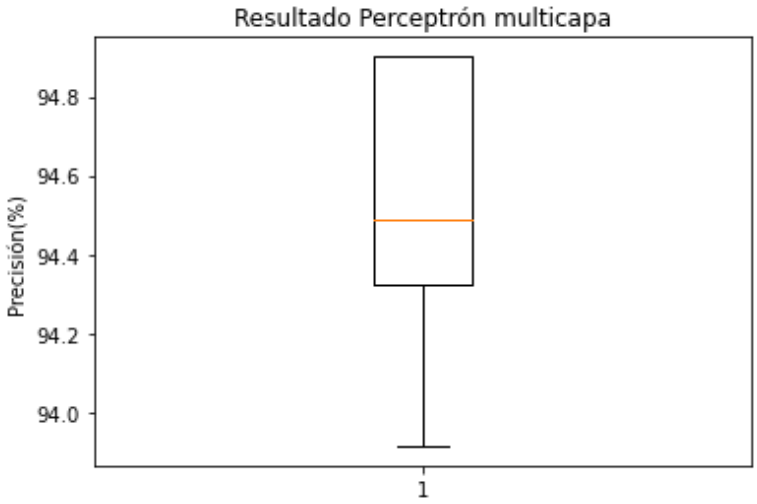

Figura 55. Resultado MLP con dos capas ocultas. Clasificación nivel del jugador.

En la Figura 56 se presentan los errores cometidos por el clasificador. Al igual que el caso anterior, el mayor error cometido son los 15 fallos al confundir el nivel avanzado con el nivel intermedio.

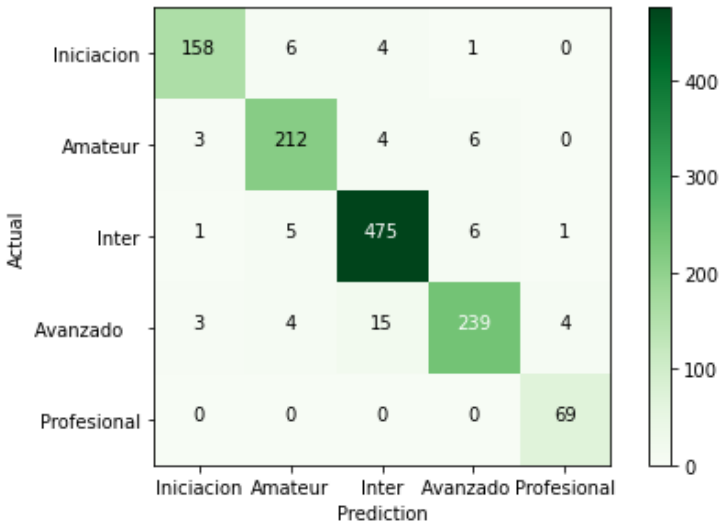

Figura 56. Algoritmo MLP con dos capas ocultas para clasificación nivel del jugador. Matriz de confusión.

Tabla 21. Tipos de errores cometidos por clasificador MLP con dos capas ocultas.

MLP		Predicción				
		Iniciación	Amateur	Intermedio	Avanzado	Profesional
Actual	Iniciación		-D [13] -S [8] -D [15] -RP [13] -RP [13] -R [1]	-RM [1] -RM [1] -D [15] -RP [8]	-RM [15]	
	Amateur	-D [2] -R [7] -GRP [20]		-DP [19] -VR [7] -GRP [19] -GD [19]	-GRP [2] -VR [7] -DP [7] -DP [19] -DP [2] -VD [7]	
	Intermedio	-D [4]	-B [4] -VD [17] -VR [17] -B [17] -GR [21]		-B [21] -GDP [14] -RP [17] -GRP [21] -VD [21] -GR [17]	-B [9]
	Avanzado	-VD [18] -GD [11] -GRP [5]	-GD [22] -GDP [11] -GD [11] -GD [11]	-VR [18] -VD [11] -B [16] -D [11] -GD [16] -R [16] -R [5] -VD [11] -RP [16] -GDP [16] -D [16] -GR [18] -VR [18] -GR [11] -S [18]		-RM [11] -GRP [11] -GDP [11] -GDP [11]
	Profesional					

En la Tabla 21 se pueden observar los tipos de errores cometidos por el clasificador. Entre los golpes realizados por jugadores de nivel iniciación que han sido clasificados con mayor grado, destacan el remate realizado por el jugador 1, el revés de pared del jugador 13 y el fondo de derecha ejecutado por el jugador 15. Estos fallos pueden deberse al hecho de que estos jugadores tienen una mecánica sólida en dichos golpes, ya que los fallos se repiten em más de una ocasión. Además, se observa que los tipos de golpes son bastante básicos, es decir, no se detectan golpes más avanzados como las voleas o las bandejas.

En el caso de los jugadores de nivel amateur, se observa una clara tendencia a cometer errores en los golpes de derecha con niveles superiores, especialmente por parte de los jugadores 19 y 7. En concreto, el movimiento que genera más confusión en este caso es la derecha después de pared, con 4 fallos registrados. En este nivel aparecen golpes más avanzados como los globos y las voleas.

A medida que aumenta el nivel de habilidad, los errores en la clasificación se vuelven más diversos. En el nivel intermedio, se destaca la aparición de la bandeja con 4 fallos, tanto en niveles inferiores como superiores, y los globos, con 4 errores.

Finalmente, en el nivel avanzado se aprecia un mayor número de errores. Los resultados reflejan una clara complejidad al clasificar los globos, en especial los globos de derecha realizados por el jugador 11. Existen 9 errores al clasificar los globos de este jugador, tanto para niveles inferiores como superiores.

5.2.2 Árbol de decisión

Los hiperparámetros utilizados junto a sus valores se muestran en la Tabla 5. Para encontrar la mejor combinación de los parámetros, se realizará una búsqueda en rejilla.

La mejor composición de los hiperparámetros es aplicar el criterio de entropía, una profundidad máxima de 30, un número mínimo de 2 muestras para dividir un nodo y 2 muestras en el nodo hoja, obteniendo una exactitud del **65.946% (± 0.517).**

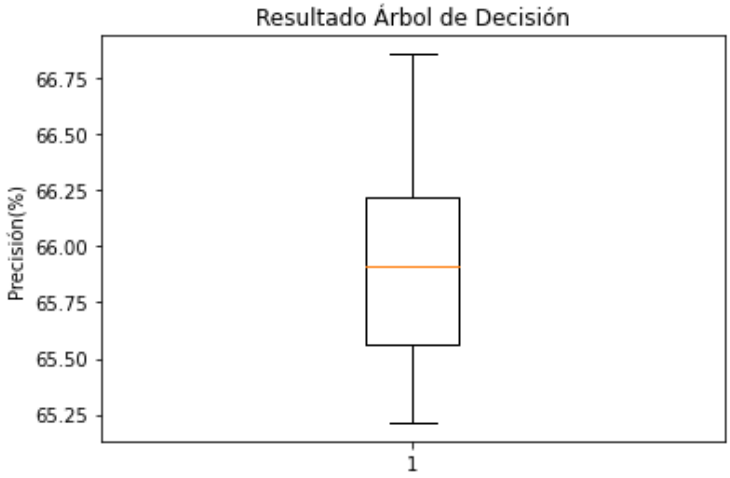

Figura 57. Resultado árbol de decisión con mejor configuración. Clasificación nivel del jugador.

En la Figura 58 se puede observar la matriz de confusión para la combinación que mejor resultados obtiene. Este algoritmo, al tener una menor precisión, presenta un número mayor de errores cometidos. Al igual que en los resultados anteriores, el error más destacado se produce al confundir el nivel avanzado con el nivel intermedio, con 61 fallos.

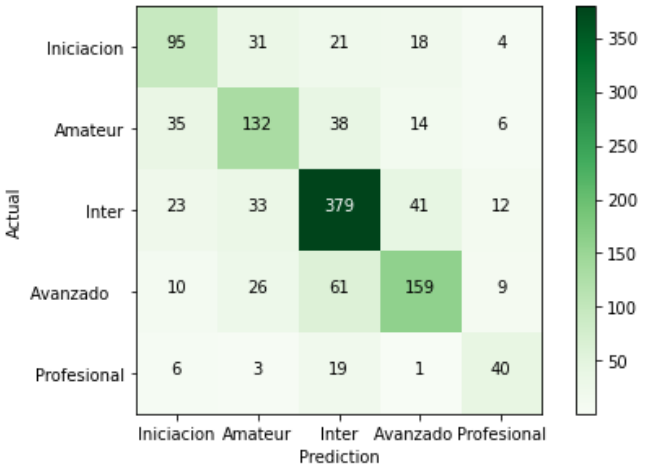

Figura 58. Algoritmo árbol de decisión para clasificación nivel de jugador. Matriz de confusión.

5.2.3 K vecinos más próximos

Los valores en función de los hiperparámetros modificados, la métrica de la distancia y el número de vecinos a tener en cuenta, se muestran en la Tabla 22.

Tabla 22. Exactitud (%) algoritmo KNN en función de los Hiperparámetros. Clasificación nivel de jugador.

		K									
		1	2	3	4	5	6	7	8	9	10
Distancia	Manhattan	93.83	92.76	91.53	92.43	92.11	91.61	91.37	91.12	90.63	90.13
	Euclidiana	91.04	88.65	88.40	87.66	87.34	86.18	86.68	85.03	84.79	84.62

Al igual que en la clasificación del tipo de golpe, se observa una diferencia entre usar la función Manhattan y la euclidiana, ofreciendo mejores resultados la función Manhattan. Las variables que ofrecen un mejor resultado son k=1 y la función Manhattan. El mejor rendimiento obtenida para esta combinación es del **93.83%**. Como ya sabemos, este valor es constante para una misma división de datos, por lo que se probará esta combinación para distintas separaciones de datos. La exactitud media obtenida es del **95.173% (± 0.686).**

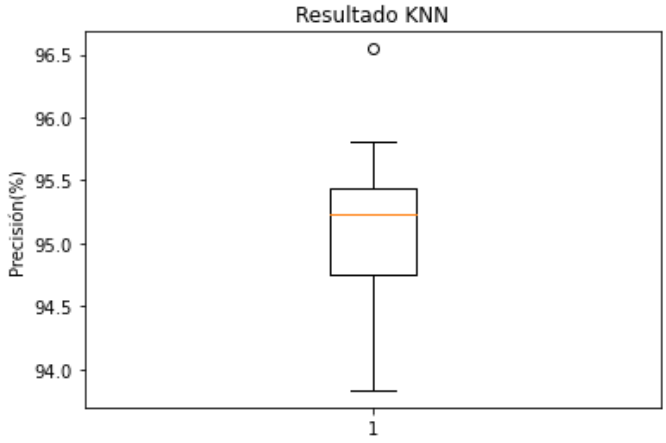

Figura 59. Resultado KNN con mejor configuración. Clasificación nivel del jugador.

La matriz de confusión se puede observar en la Figura 60. Sobresale en este caso los 15 errores al clasificar el nivel avanzado como nivel profesional. Este tipo de error no es tan acentuado en los anteriores algoritmos.

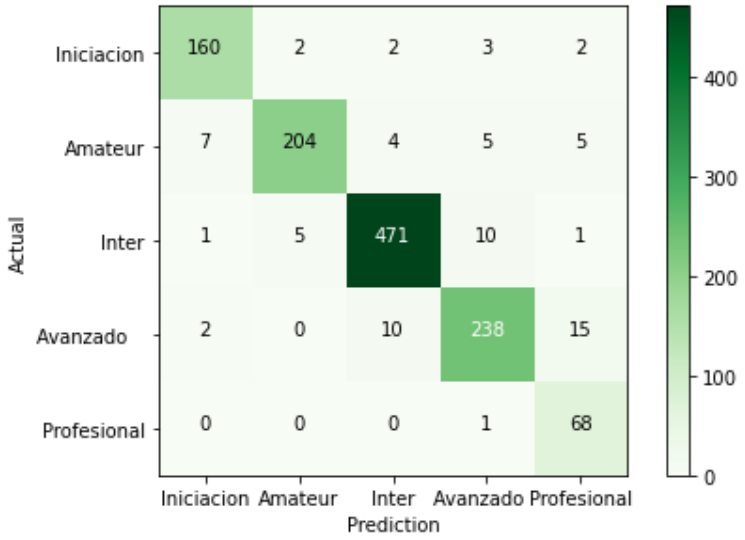

Figura 60. Algoritmo KNN para clasificación nivel de jugador. Matriz de confusión.

Tabla 23. Tipos de errores cometidos por clasificador KNN.

KNN		Predicción				
		Iniciación	Amateur	Intermedio	Avanzado	Profesional
Actual	Iniciación		-D [13] -DP [13]	-R [8] -RP [13]	-RM [1] -RP [13] -RM [15]	-VD [13] -R [13]
	Amateur	-VD [20] -RP [2] -DP [2] -B [2] -GRP [7] -GR [7] -R [3]		-B [3] -RP [2] -R [3] -GD [19]	-RP [20] -D [2] -DP [2] -GRP [20] -DP [2]	-D [2] -GDP [2] -D [2] -R [2] -B [2]
	Intermedio	-VR [17]	-VD [17] -RP [14] -VR [17] -B [17] -GR [17]		-RP [14] -GRP [14] -GRP [21] -GDP [21] -VR [21] -RM [9] -GR [21] -VD [9] -GR [17] -DP [21]	-VR [12]
	Avanzado	-VD [18] -GRP [5]		-VD [5] -RM [5] -D [11] -RP [16] -R [16] -R [5] -RP [16] -R [5] -GR [18] -S [18]		-GD [11] -RM [11] -D [11] -VD [11] -GDP [11] -GD [11] -GRP [11] -GD [11] -RP [11] -D [11] -GD [11] -GDP [11] -GD [11]

						-GDP [11] -GDP [11]
	Profesional				-GRP [6]	

En la Tabla 23 se presentan los tipos de golpes e identidades de los jugadores en los fallos. En el nivel de iniciación, se observa la misma tendencia del clasificar anterior, donde se cometen errores en golpes básicos, como son la derecha, el revés y el remate. Además, se aprecia un alto número de errores cometidos por el jugador 13. Este jugador podría tener una mayor habilidad en comparación con otros jugadores categorizados en el mismo nivel.

En los siguientes niveles, se refleja una clara tendencia a confundir los golpes de revés en niveles inferiores y los golpes de derecha en niveles superiores, especialmente en los niveles amateur y avanzado. Esta tendencia podría deberse al hecho de que los golpes de derecha son más fáciles de ejecutar que los golpes de revés. En el caso del nivel amateur, destacan los golpes de derecha realizados por el jugador 2, clasificados en niveles superiores, y los golpes de revés ejecutados por el jugador 7, clasificados en niveles inferiores.

En cuanto al nivel avanzado, los únicos golpes clasificados como nivel profesional son ejecutados por el jugador 11. Además, se observa una gran cantidad de fallos de este tipo, entre los que destaca los globos de derecha. La razón de este tipo de error podría ser que el jugador tenga muy mecanizado este movimiento, además de no ser un golpe muy complejo.

5.2.4 Máquina de vector soporte

Los resultados obtenidos por la Máquina de Vector Soporte al variar los valores de los distintos parámetros se presentan en la Tabla 24. Se modifican la regulación y la función de Kernel. Al igual que en la clasificación del tipo de golpe, el hiperparámetro para la separación multiclase (decisión_fuction_shape) no produce ninguna variación en la exactitud alcanzada, por lo que no se muestra en la tabla.

Tabla 24. Exactitud (%) algoritmo SVM en función de los hiperparámetros. Clasificación nivel de jugador.

		C									
		0.1	1	2	5	8	10	12	15	20	50
Kernel	Lineal	62.34	59.62	59.21	58.06	57.73	57.65	57.32	57.57	57.57	57.57
	Polinómico	54.44	83.55	87.91	90.87	91.37	91.61	91.78	91.37	91.04	90.87
	Radial	53.04	86.18	90.95	93.50	93.91	94.08	93.75	93.67	93.83	93.75
	Sigmoide	39.64	29.69	28.95	25.99	26.89	26.97	26.81	26.64	26.97	25.32

En este caso, el mejor resultado lo alcanza la configuración C=10 y función de Kernel radial, logrando un **94.08%** de aciertos. Los peores valores obtenidos utilizan la función sigmoide, alcanzando como máximo un 39.64%. Al variar la división de los datos, se obtiene una exactitud media del **93.890% (±0.668)**.

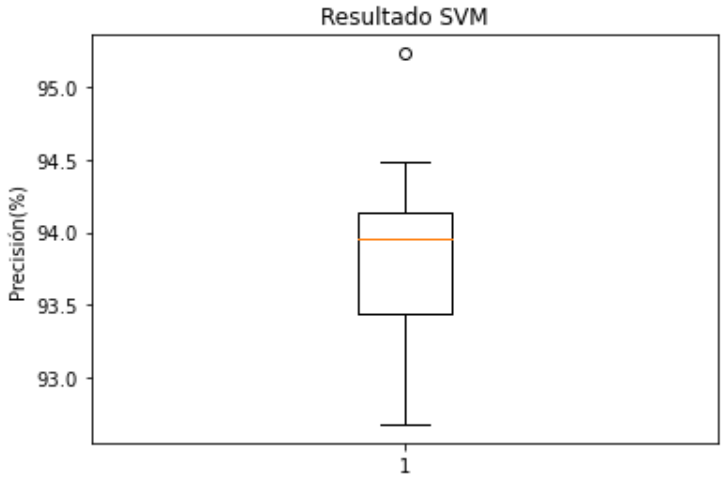

Figura 61. Resultado SVM con mejor configuración. Clasificación nivel del jugador.

A continuación, en la Figura 62 se muestra la matriz de confusión para dicha configuración. El mayor error cometido al clasificar el nivel del jugador es confundir el nivel avanzado con el nivel intermedio, con 17 fallos. Este tipo de error es el más común.

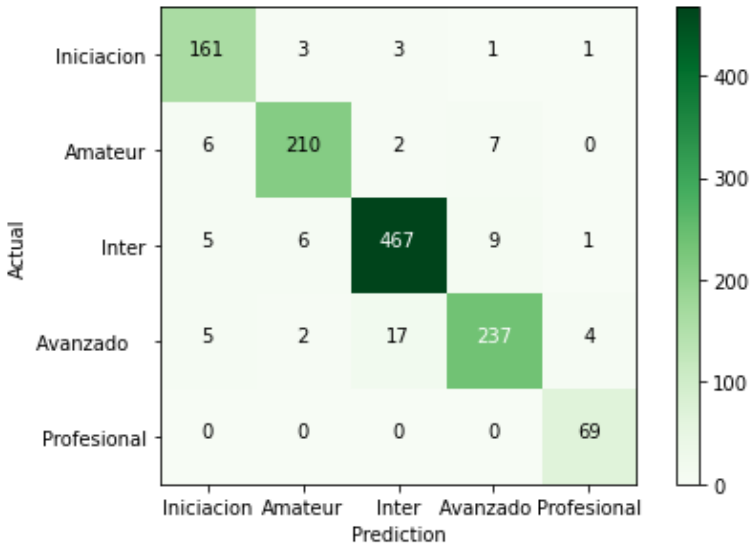

Figura 62. Algoritmo SVM para clasificación nivel de jugador. Matriz de confusión.

Tabla 25. Tipos de errores cometidos por clasificador SVM.

SVM		Predicción				
		Iniciación	Amateur	Intermedio	Avanzado	Profesional
Actual	Iniciación		-RM [1] -RP [13] -RP [13]	-R [8] -RM [1] -D [15]	-RM [15]	-R [13]
	Amateur	-R [7] -VR [7] -B [2] -GR [7] -S [2] -R [3]		-R [2] -GD [19]	-D [2] -D [2] -D [2] -DP [7] -D [2] -DP [19] -VD [7]	
	Intermedio	-VR [10] -GR [21] -DP [9] -R [21] -VD [10]	-GDP [14] -VD [17] -VR [17] -GR [17] -B [17] -VR [17]		-B [10] -GDP [4] -B [21] -B [21] -VR [12] -RP [17] -RM [9] -GRP [9] -VR [4]	-B [9]
	Avanzado	-VD [18] -VD [18] -GRP [5] -VD [11] -GDP [11]	-B [16] -GDP [11]	-GD [11] -RM [11] -GR [11] -D [11] -RM [5] -R [16] -VD [11] -RM [18] -R [5] -VD [11] -RP [16] -R [5] -GDP [16]		-GRP [11] -GD [11] -GDP [11] -GDP [11]

				-GR [11] -D [16] -GR [18] -S [18]		
	Profesional					

Los tipos de errores se especifican en la Tabla 25. En el nivel de iniciación, los errores son menos diversos y corresponden a golpes básicos. Se destaca el remate realizado por el jugador 1 y el revés del jugador 13. En el nivel amateur, se observa la misma tendencia que el clasificador anterior de confundir los golpes de revés en niveles inferiores y los golpes de derecha en niveles superiores. En este nivel, también aparecen golpes más avanzados como las voleas y los globos. El error más común es la clasificación incorrecta de la derecha del jugador 2 en niveles superiores, con 4 fallos.

En los niveles intermedio y avanzado, se aprecia una gran cantidad de errores diversos. En el nivel intermedio, se destacan las voleas clasificadas por debajo de este nivel y las bandejas por encima del nivel, ambos golpes complejos. Por otro lado, en el nivel avanzado, se producen una gran cantidad de errores al clasificar los golpes realizados por el jugador 11, una característica que también se presentaba en los anteriores clasificadores. En especial, los globos realizados por este jugador muestran una variación considerable en su etiquetado.

5.2.5 Extremely Randomized Trees

Las combinaciones de los distintos hiperparámetros probados para obtener el mejor rendimiento del clasificador puede observarse en la Tabla 14.

La accuracy obtenida es del **92.484 (±0.308)** al emplear 500 estimadores bases y el método pasting para la división de las muestras.

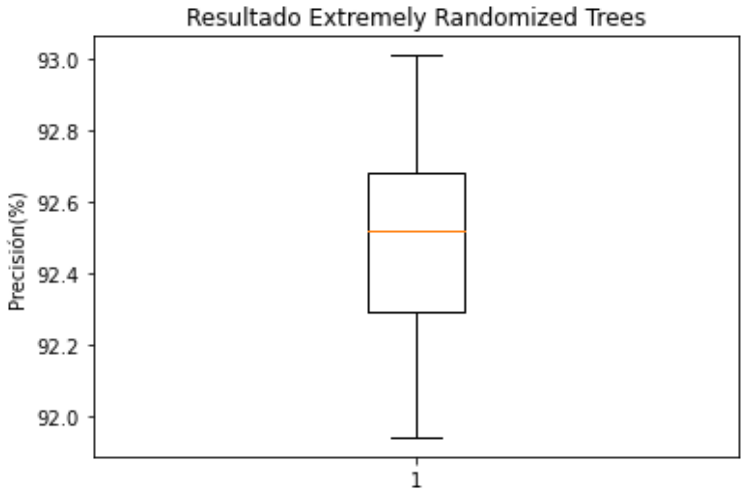

Figura 63. Resultado Extremely Randomized Trees con mejor configuración. Clasificación nivel del jugador.

En la Figura 64 se muestra la matriz de confusión al emplear la mejor combinación de hiperparámetros del clasificador. En cuanto a los errores, destacan los 38 fallos al clasificar el nivel avanzado como intermedio y 17 fallos al clasificar el nivel amateur como nivel intermedio.

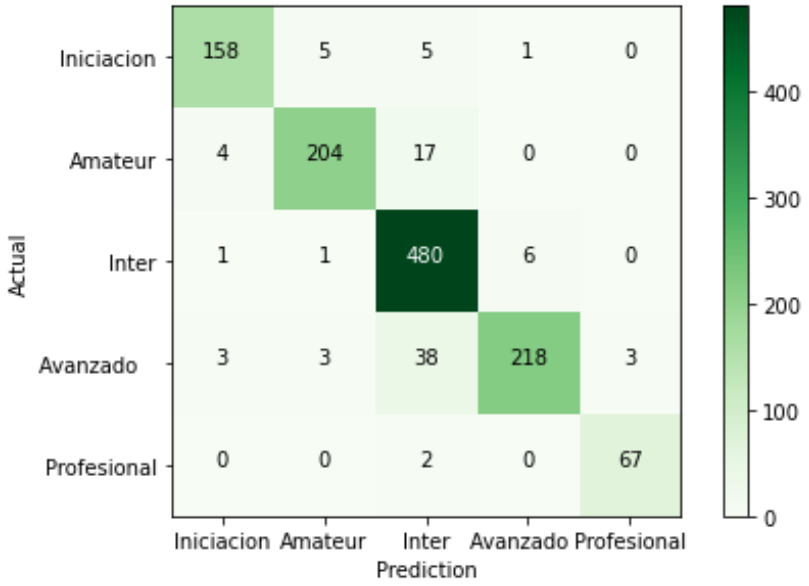

Figura 64. Algoritmo Extremely Randomized Trees para clasificación nivel de jugador. Matriz de confusión.

5.2.6 AdaBoost

Para el algoritmo AdaBoost escogeremos como algoritmo base el clasificador árbol de decisión por los resultados obtenidos en la clasificación previa. En la Tabla 26 se presenta la exactitud obtenida al variar los dos parámetros asociados al método Boosting: el coeficiente de aprendizaje y el número de estimadores.

Tabla 26. Exactitud (%) algoritmo AdaBoost con árbol de decisión en función de los hiperparámetros. Clasificación nivel del jugador.

		Learning rate				
		0.01	0.05	0.1	0.5	1
N.º estimadores	50	85.86	90.38	89.14	90.30	88.16
	100	88.98	90.79	92.02	91.04	89.88
	250	89.39	91.69	92.11	91.53	90.95
	500	90.79	91.94	91.78	90.87	91.20

Como podemos observar, la mejor combinación de hiperparámetros se da al utilizar 250 estimadores bases y un coeficiente de aprendizaje de 0.1. En esta tabla podemos apreciar como aumenta la accuracy hasta los 250 estimadores bases. La exactitud media obtenida aplicando la mejor combinación de hiperparámetros es del **91.65% (±0.189)**.

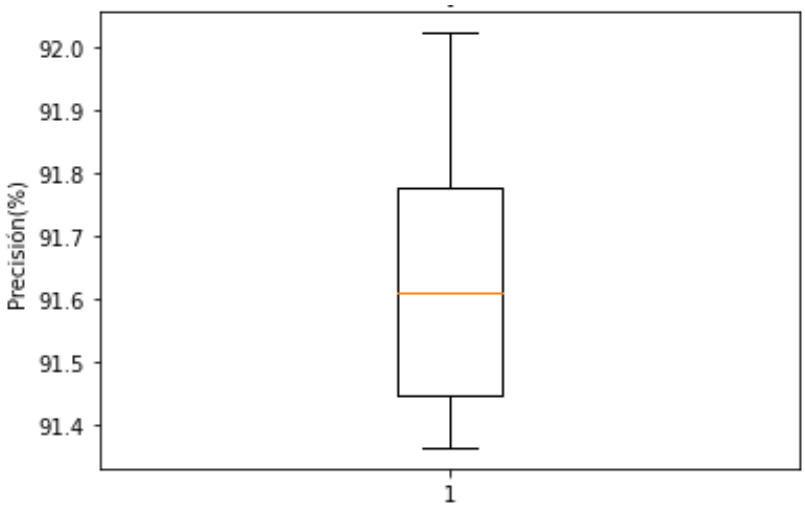

Figura 65. Resultado AdaBoost con árbol de decisión utilizando la mejor configuración. Clasificación nivel del jugador.

Los distintos errores cometidos por el clasificador se pueden observar en la Figura 66. Se aprecia una mayor cantidad de errores en las predicciones del nivel intermedio, con 34 errores al clasificar el nivel avanzado como nivel intermedio y 20 fallos al clasificar el nivel amateur como nivel intermedio.

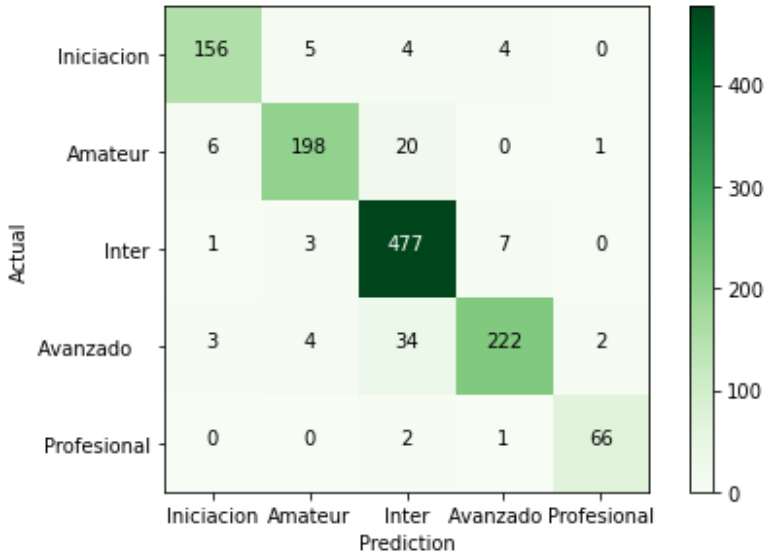

Figura 66. Algoritmo AdaBoost y clasificador base árbol de decisión para clasificación nivel de jugador. Matriz de confusión.

5.2.7 Clasificador por votación

Al igual que en el clasificador por votación previo, emplearemos cuatro estimadores bases: árbol de decisión, SVM, KNN, MLP. Para la predicción final se empleará el método soft voting. Por lo tanto, el hiperparámetros a variar será el peso otorgado a cada estimador. El mejor resultado se obtiene al configurar los siguientes pesos: 3 para los algoritmos MLP, KNN y SVM y 1 para el árbol de decisión. La exactitud alcanzada es del **96.74% (±0.084)**.

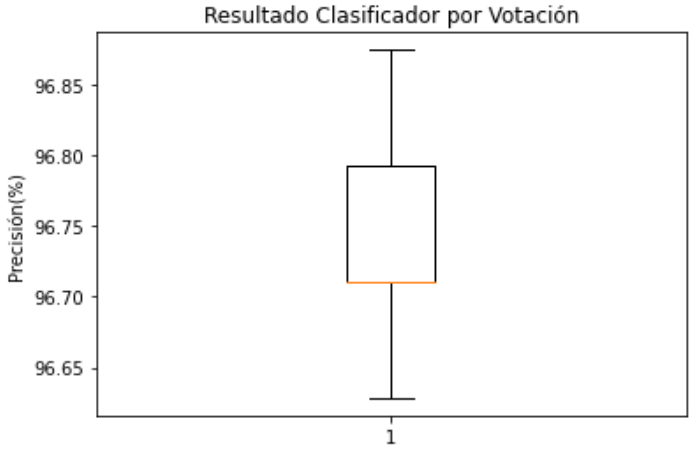

Figura 67. Resultado clasificador por votación utilizando la mejor configuración. Clasificación nivel del jugador

La matriz de confusión mostrada en la Figura 68 nos enseña los errores cometidos por el clasificador. Destacan los 11 fallos cometidos al confundir el nivel avanzado con el nivel intermedio y los 6 fallos al confundir el nivel amateur con el nivel avanzado.

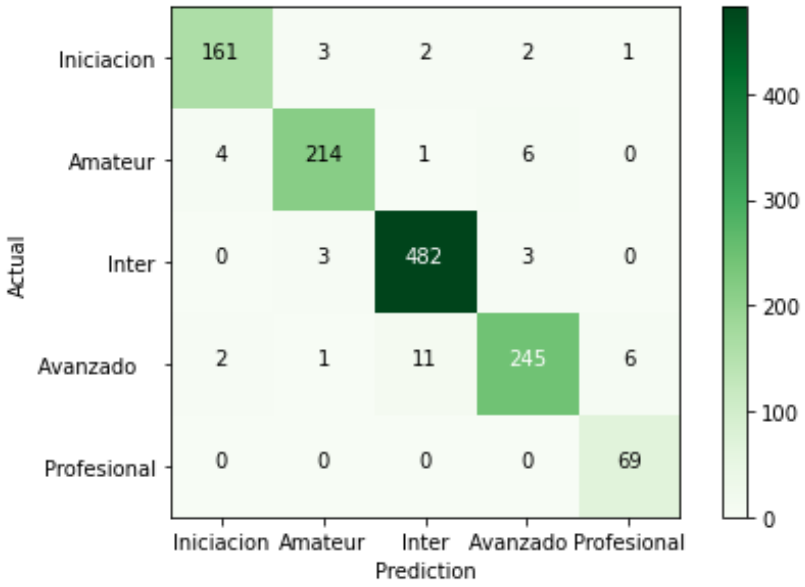

Figura 68. Algoritmo clasificador por votación para clasificación nivel de jugador. Matriz de confusión.

Tabla 27. Tipos de errores cometidos por clasificador por votación.

Clf_Votacion_Soft		Predicción				
		Iniciación	Amateur	Intermedio	Avanzado	Profesional
Actual	Iniciación		-D [13] -RP [13] -RP [13]	-R [8] -RM [1]	-RM [1] -RM [15]	-R [13]
	Amateur	-R [7] -DP [2] -B [2] -GRP [7] -GR [7]		-GD [19]	-D [2] -VR [7] -DP [7] -DP [19] -DP [2]	
	Intermedio		-VD [17] -VR [17] -B [17]		-RP [17] -RM [9] -GR [17]	
	Avanzado	-VD [18] -GRP [5]	-GDP [11]	-B [16] -D [11] -RM [5] -R [16] -R [5] -VD [11] -GDP [16] -D [16] -GR [18] -S [18]		-GD [11] -RM [11] -GRP [11] -GD [11] -GDP [11] -GDP [11]
	Profesional					

El clasificador por votación nos muestra los errores más comunes producidos por los estimadores bases, ya que utiliza las predicciones de estos. En el nivel de iniciación, este clasificador presenta errores en el remate del jugador 1 y en el revés del jugador 13. Como ya se ha mencionó previamente, estos son golpes básicos.

La tendencia a confundir los golpes de revés en niveles inferiores y los golpes de derecha en niveles superiores, que se observó en los clasificadores anteriores, se repite en los niveles amateur y avanzado. También se aprecia la aparición de golpes más avanzados a medida que aumenta el nivel de los jugadores. Esto podría deberse a la experiencia de los jugadores, ya que, a un nivel más alto, los jugadores irán desarrollando y mejorando golpes más avanzados.

Por último, el jugador 11 vuelve a aparecer en los errores de nivel avanzado, ya que ocupa todos los errores cometidos al etiquetar sus golpes, en particular los globos de derecha, en el nivel profesional. Cada jugador tiene golpes más desarrollados que otros, por lo que no es sorprendente que este jugador tenga buena técnica en los globos.

5.2.8 Comparación clasificación nivel del jugador

En la Tabla 28 se presenta una comparación entre los distintos algoritmos desarrollados para la clasificación del nivel del jugador. En esta tabla se disponen la exactitud media, la exactitud máxima, el tiempo de entrenamiento y el tiempo de predicción obtenido por los clasificadores.

Tabla 28. Comparación clasificadores para el nivel del jugador

		Exactitud media (%)	Exactitud máxima (%)	Mejora exactitud (%)	Tiempo entrenamiento (s)	Tiempo predicción (s)
Perceptrón multicapa	Una capa oculta	91.99	92.52	-	267.961	0.014
	Dos capas ocultas	94.51	94.90	-	41.219	0.05
Árbol de decisión		65.95	66.86	-	1.233	0.003
K vecinos más próximos		95.17	96.55	-	0.013	0.417
Máquina de Vector Soporte		93.89	95.23	-	3.108	0.533
Extremely Randomized Trees		92.48	93.01	26.53	0.950	0.171
AdaBoost		91.65	92.11	25.70	653.472	0.370
Clasificador por votación		96.74	96.88	1.57	58.642	1.577

Exceptuando el algoritmo árbol de decisión, podemos observar un gran rendimiento y eficiencia de los demás clasificadores. En particular, destaca la exactitud obtenida por el clasificador por votación, con una exactitud media del 96.74% y una exactitud máxima del 96.88%. También, los resultados logrados con K vécinos más próximos, la Máquina de Vector Soporte y el perceptrón multicapa con dos capas ocultas son notables, alcanzando una accuracy media del 95.17, 93.89% y 94.53% respectivamente.

El tiempo de entrenamiento transcurrido en el algoritmo AdaBoost y perceptrón multicapa es considerablemente superior al resto. Además, destaca que se emplee más tiempo para la red neuronal con una capa oculta que para dos capas ocultas. Esta diferencia se puede deber a las iteraciones realizadas. Como se mencionó anteriormente, el perceptrón multicapa de una capa oculta alcanza su resultado en la iteración 70, mientras que la red neuronal de dos capas ocultas lo consigue en la iteración 28.

En cuanto al árbol de decisión, la exactitud alcanzada es razonablemente inferior al resto de clasificadores. Sin embargo, al aplicar los algoritmos de ensamble de clasificadores podemos alcanzar exactitudes similares a los demás clasificadores. Al utilizar los ensambles AdaBoost y Extremely Randomized Trees se obtienen una accuracy del 91.65% y 92.48% respectivamente, lo que implica una mejora alrededor del 26%.

Para finalizar, podemos observar en los errores cometidos que se producen clasificaciones incorrectas de golpes más avanzados a medida que aumenta el nivel del jugador. Esto resulta evidente, ya que a medida que los jugadores adquieren más experiencia, desarrollan golpes más complejos de realizar. Además, es notable que los clasificadores etiquetan incorrectamente los golpes de derecha en niveles superiores con mayor frecuencia que los golpes de revés. Este fenómeno se debe a que los golpes de derecha suelen ser más sencillos de ejecutar en comparación con los golpes de revés.

6 CONCLUSIÓN Y TRABAJOS FUTUROS

6.1 Conclusiones

En primer lugar, con respecto a la clasificación del tipo de golpe, se ha observado una mejora significativa en el rendimiento al emplear los algoritmos de ensamble de clasificadores. Al entrenar estos ensambles, hemos obtenido nueve algoritmos que superan o se aproximan al umbral del 90% de aciertos. Es importante destacar que los únicos algoritmos que muestran una disminución en el rendimiento son el método de Bagging y AdaBoost, ambos utilizando el clasificador SVM como estimador base. En consecuencia, queda claramente demostrado que la utilización de algoritmos de ensamble aporta un valor significativo a la tarea de clasificación del tipo de golpe.

Por otro lado, se ha demostrado que los algoritmos de aprendizaje automático son altamente efectivos para la clasificación del nivel del jugador. A excepción del algoritmo de árbol de decisión, todos los demás clasificadores han logrado superar el 90% de aciertos. Además, hemos demostrado que el uso de ensambles de clasificadores también contribuye al incremento del rendimiento de los algoritmos en esta tarea.

En ambos tipos de clasificación, el clasificador por votación ha sobresalido al lograr los mejor resultados en cuanto a porcentaje de aciertos. Cabe destacar que este algoritmo no solo ofrece resultados precisos, sino que también presenta tiempos de predicción y entrenamiento relativamente bajos, lo que lo convierte en una opción adecuada en términos de eficiencia. Por otro lado, si se busca un algoritmo con bajos tiempos de entrenamiento y predicción a costa de una menor precisión, el algoritmo Extremely Randomized Forest es una opción adecuada.

Además de los resultados obtenidos en términos de exactitud en la clasificación del tipo de golpe y del nivel del jugador, es importante destacar que hemos identificado ciertas tendencias en los errores de clasificación. Estas tendencias pueden proporcionar información valiosa para la comprensión de las dificultades asociadas a la clasificación y pueden servir como guía para futuras investigaciones en busca de mejoras.

En resumen, este estudio ha demostrado la efectividad de los algoritmos de aprendizaje automático, en especial los algoritmos de ensamble de clasificadores, para llevar a cabo la clasificación de los golpes de pádel, tanto en la clasificación del tipo de golpe como en la clasificación del nivel de jugador.

6.2 Trabajos futuros

Los trabajos futuros pueden enfocarse en varias vertientes, la mejora de los algoritmos de clasificación, la ampliación de la base de datos o la implementación de un sistema en tiempo real. Para mejorar los algoritmos de clasificación, se podría implementar una combinación de ensambles de clasificadores. En este proyecto, los ensambles de clasificadores utilizados empleaban únicamente algoritmos individuales como estimadores bases. Sin embargo, es posible aplicar una combinación de ensambles de clasificadores, donde el estimador base de un ensamble sea otro ensamble diferente. Un ejemplo sería aplicar el algoritmo Extremely Randomized Forest como estimador base en el clasificador por votación.

Para la ampliación y mejora de la base de datos, existen varias opciones. Se puede añadir nuevos golpes y los efectos con los que se realiza. Existen algunos golpes que no están incluidos en la base de datos, como la contrapared, la dejada, la chiquita, la víbora, el remate por tres. Algunos de estos golpes difieren con otros en el efecto otorgado a la bola. Por ejemplo, la bandeja y la víbora son golpes muy similares, que dependen de la velocidad y efecto aplicada a la bola, al igual que el remate y el remate por tres.

Otra mejora para el conjunto de datos sería la separación de las muestras en función de la posición del jugador. En el pádel, existen dos posiciones principales: los jugadores de revés, que juegan en el lado izquierdo de la pista y los jugadores de derecha, que ocupan el lado derecho de la pista. Aunque en niveles bajos este aspecto a menudo se pasa por alto, cada tipo de jugador desempeña una función específica. Los jugadores de revés tienden a dominar más el juego aéreo, utilizando bandejas y remates con mayor frecuencia. Por otro lado, los jugadores de derecha suelen ser jugadores más defensivos, empleando golpes de fondo o globos con mayor

frecuencia. Además, como su nombre indica, los jugadores de revés utilizan más golpes de revés en comparación con los jugadores de derecha, lo que también influye en la dirección y técnica de los golpes. Estos hechos pueden generar fuentes de error en la clasificación del nivel del jugador y tipo de golpe, dado que cada posición se especializa en ciertos tipos de golpes. Se propone, por lo tanto, la creación de dos bases de datos separadas según la posición del jugador, con el fin de evaluar el rendimiento de los clasificadores en la clasificación del tipo de golpe y nivel del jugador.

El tamaño de la base de datos utilizada para esta investigación consta de 4002 golpes. Cuantas más extensa sea una base de datos, mejores modelos de los algoritmos se obtendrán. La mayoría de las bases de datos utilizan miles de datos para los algoritmos de aprendizaje. Por lo tanto, el aumento de muestras de golpes de pádel supondría una mejora significativa.

Las muestras tomadas para la creación de la base de datos atienden únicamente al movimiento del brazo o de la mano, siendo el movimiento de las piernas y torso fundamental para llevar a cabo un buen golpe. La flexión de las rodillas, junto al giro del tronco, permite al jugador efectuar el golpe de la pelota con mayor velocidad y precisión. La recogida de datos del movimiento de las piernas y tronco podría facilitar la clasificación de golpes y permitiría evaluar de manera más precisa la técnica utilizada por los deportistas en cada golpe.

Finalmente, la implementación de un sistema en tiempo real con un modelo ya entrenado supondría un gran avance. Este sistema permitiría clasificar el tipo de golpe realizado y el nivel del jugador que lo ha realizado en entrenamientos o competiciones, lo que sería de gran ayuda para el desarrollo del jugador por parte de los entrenadores, además de facilitar futuras investigaciones. Este sistema deberá ser capaz de aislar los golpes del resto del tiempo en el que el jugador no está realizando ningún movimiento, para su posterior clasificación.

REFERENCIAS

[1] B. J. S.-A. Martínez, «Historia del padel,» *Materiales para la Historia del Deporte,* nº 11, p. 4, 2013.

[2] J. C. Ibáñez, «Evolución del pádel en España en función del género y edad de los practicantes,» 2016. [En línea]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=5978842. [Último acceso: 4 Mayo 2023].

[3] A. T. F. S. A. S. E. U. A. C. E. S. Y. J. T. M. D. T. F. J. O. J. A. H. C. Z. N. V. y. O. P. Julio Cesar, «Inteligencia Artificial,» Iniciativa Latinoamericana de Libros de Texto Abiertos, 2014, pp. 16-18.

[4] V. M. J. L. H.-L. E. I. A.-J. y. E. F. V.-A. Diana Nancy, «Avances de la inteligencia artificial en salud,» *Ciencias de la Salud,* vol. 5, nº 3, pp. 603-613, 2019.

[5] M. M.-P. V. Q.-V. J.F. Avila-Tomás, «La inteligencia artificial y sus aplicaciones en medicina: introducción antecedentes en la IA y robótica,» *ScienceDirect,* vol. 52, nº 10, pp. 778-784, 2020.

[6] P. Sandonnini, «Robot Sophia: La revolución de los robots humanoides,» Innovación digital 360, 03 07 2023. [En línea]. Available: https://www.innovaciondigital360.com/i-a/presentamos-a-sophia-la-robot-humanoide-que-expresa-emociones/. [Último acceso: 30 08 2023].

[7] J. Torres, Deep Learning, Introducción práctica con Keras (PRIMERA PARTE), Barceloma: WATCH THIS SPACE, 2018.

[8] Lalit, «encora,» 30 Septiembre 2021. [En línea]. Available: https://www.encora.com/es/blog/aplicaciones-de-aprendizaje-autom%C3%A1tico. [Último acceso: 4 Mayo 2023].

[9] Y. L. y. W. Chen, «A Comparative Performance Assessment of Ensemble Learning for Credit Score,» *MDPI,* vol. 8, nº 10, p. 1756, 2020.

[10] G. C. Domínguez, «Comparación de algoritmos de aprendizaje automático para clasificación de golpes de pádel,» Sevilla, 2021.

[11] C. M. Valerio, «Comparación de algoritmos de aprendizaje automático para la clasificación de golpes de pádel: Dominio temportal versus frecuencial,» Sevilla, 2022.

[12] T. Perri, M. Reid, A. Murphy, K. Howle y R. Duffield, «Sensors,» 16 Noviembre 2022. [En línea]. Available: https://doi.org/10.3390/s22228868. [Último acceso: 7 Julio 2023].

[13] P. K. N. E. O. M. G. M. W. a. C. O. D. Connaghan, «Multi-sensor classification of tennis strokes,» de *Sensors*, Limerick, 2011.

[14] D. B. P. y. C. O. U. Luis Benages Pardo, «Detection of Tennis Activities with Wearable Sensors,» *Sensors,* p. 19, 2019.

[15] B. J. ,. C.-I. J. M. D. I.-C. P. S. d. Z. F. S.-P. A. Sánchez-Alcaraz Martínez, «Análisis de las acciones de

5] ataque en el pádel masculino profesional,» *Apunts Educación Física y Deportes,* vol. 36, nº 142, p. 7, 2020.

[1 J. Courel-Ibáñez, B. J. Sánchez-Alcaraz Martinez y D. Muñoz Marín, «Exploring Game Dynamics in
6] Padel: Implications for Assessment and Training,» *Journal of Strength and Conditioning Research,* vol. 33, nº 7, pp. 1971-1977, 2019.

[1 M. Á. G.-R. S. J. I. B. J. S.-A. y. D. M. Adrían Escudero-Tena, «Importance of Maintaining Net Position in
7] Men's and Women's Professional Padel,» *Perceptual and Motor Skill,* vol. 0, nº 0, pp. 1-16, 2023.

[1 A. Delgado, «Geeknetic,» 21 Noviembre 2020. [En línea]. Available: https://www.geeknetic.es/Raspberry-
8] Pi/que-es-y-para-que-sirve. [Último acceso: 7 Mayo 2023].

[1 «Wikipedia,» 2 Julio 2021. [En línea]. Available:
9] https://es.wikipedia.org/wiki/Unidad_de_medici%C3%B3n_inercial. [Último acceso: 7 Mayo 2023].

[2 «Parlamento Europeo,» 26 03 2021. [En línea]. Available:
0] https://www.europarl.europa.eu/news/es/headlines/society/20200827STO85804/que-es-la-inteligencia-artificial-y-como-se-usa?at_campaign=20234-Digital&at_medium=Google_Ads&at_platform=Search&at_creation=DSA&at_goal=TR_G&at_audience=&at_topic=Artificial_Intel. [Último acceso: 04 08 2023].

[2 «Instituto de ingeniería del conocimiento,» [En línea]. Available: iic.uam.es. [Último acceso: 04 08 2023].
1]

[2 «Talento,» [En línea]. Available: https://talentocorporativo.com/diferencias-entre-machine-learning-
2] inteligencia-artificial-y-robotica/. [Último acceso: 04 08 2023].

[2 TensorFlow, [En línea]. Available: 2023. [Último acceso: 08 08 2023].
3]

[2 «Keras,» TensorFlow, [En línea]. Available: https://www.tensorflow.org/guide/keras?hl=es-419. [Último
4] acceso: 08 08 2023].

[2 Keras, [En línea]. Available: https://keras.io/. [Último acceso: 08 08 2023].
5]

[2 Pedregosa, «Scikit-learn: Machine Learning in Python,» Scikit.learn, 2011. [En línea]. Available:
6] https://scikit-learn.org/stable/index.html. [Último acceso: 08 08 2023].

[2 V. Morales-Oñate, «Bookdown,» 11 04 2023. [En línea]. Available:
7] https://bookdown.org/victor_morales/TecnicasML/. [Último acceso: 05 08 2023].

[2 D. Rodríguez, «Analytics Lane,» 16 12 2019. [En línea]. Available:
8] https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/. [Último acceso: 05 08 2023].

[2 A. Géron, Hands on Machine Learning with Scikit-Learn, Keras and TensorFlow, Sebastopol: O'Reilly
9] Media, 2019.

[3 «Aprende Machine Learning,» 12 12 2017. [En línea]. Available:
0] https://www.aprendemachinelearning.com/que-es-overfitting-y-underfitting-y-como-solucionarlo/. [Último acceso: 05 08 2023].

[31] J. Ponce, «Funciones de activacion y como puedes crear la tuya usando python y tensorflow,» [En línea]. Available: https://jahazielponce.com/funciones-de-activacion-y-como-puedes-crear-la-tuya-usando-python-r-y-tensorflow/. [Último acceso: 14 08 2023].

[32] «Sklearn.neural_network.MLPClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. [Último acceso: 14 08 2023].

[33] «¿Qué es un árbol de decisión?,» IBM, [En línea]. Available: https://www.ibm.com/es-es/topics/decision-trees. [Último acceso: 06 08 2023].

[34] «Sklearn.tree.DecisionTreeClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Último acceso: 06 08 2023].

[35] «¿Qué es el algoritmo de k vecinos más cercanos?,» IBM, [En línea]. Available: https://www.ibm.com/es-es/topics/knn#:~:text=El%20algoritmo%20de%20k%20vecinos%20m%C3%A1s%20cercanos%2C%20tambi%C3%A9n%20conocido%20como,un%20punto%20de%20datos%20individual.. [Último acceso: 07 08 2023].

[36] L. Salcedo, «Introducción al Machine Learning#9 - k vecinos más cercanos,» Mi Diario Python, 22 12 2020. [En línea]. Available: https://pythondiario.com/2018/01/introduccion-al-machine-learning-9-k.html. [Último acceso: 07 08 2023].

[37] «Sklearn.neighbors.KNeighborsClassifier,» Scikit learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier. [Último acceso: 07 08 2023].

[38] J. A. Rodrigo, «Máquinas de vector soporte(Support Vector Machines, SVMs),» Ciencia de datos, 04 2017. [En línea]. Available: https://cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines#Hiperplano_y_Maximal_Margin_Classifier. [Último acceso: 08 08 2023].

[39] «Sklearn.svm.SVC,» Scikit Learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC. [Último acceso: 08 08 2023].

[40] J. M. Heras, «Ensambles: voting, bagging, boosting, stacking,» IArtificial, 31 05 2019. [En línea]. Available: https://www.iartificial.net/ensembles-voting-bagging-boosting-stacking/#Votacion_por_mayoria. [Último acceso: 09 08 2023].

[41] «sklearn.ensemble.VotingClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html#sklearn.ensemble.VotingClassifier. [Último acceso: 2023 08 2023].

[42] «¿Qué es Bagging?,» IBM, [En línea]. Available: https://www.ibm.com/es-es/topics/bagging. [Último acceso: 09 08 2023].

[43] «sklearn.ensemble.BaggingClassifier,» Scikit-leanr, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html. [Último acceso: 09 08 2023].

[44] S. E. R, «Understand Random Forest Algorithms With Examples,» Analytics Vidhya, 05 07 2023. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/. [Último acceso: 10 08 2023].

[45] «Sklearn.ensemble.RandomForestClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier. [Último acceso: 10 08 2023].

[46] «Sklearn.ensemble.ExtraTreesClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier. [Último acceso: 10 08 2023].

[47] A. Saini, «Master the AdaBoost Algorithm: Guide to Implement and Understanding AdaBoost,» Analytics Vidhya, 15 09 2021. [En línea]. Available: https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/. [Último acceso: 11 08 2023].

[48] Raman, «Boosting in Machine Learning,» Geeksforgeeks, 23 05 2023. [En línea]. Available: https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/. [Último acceso: 13 08 2023].

[49] «Sklearn.ensemble.AdaBoostClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier. [Último acceso: 11 08 2023].

[50] J. A. Rodrigo, «Gradient Boosting con Python,» Ciencia de datos, 10 2020. [En línea]. Available: https://cienciadedatos.net/documentos/py09_gradient_boosting_python.html. [Último acceso: 13 08 2023].

[51] «Sklearn.ensemble.GradientBoostingClassifier,» Scikit-learn, [En línea]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier. [Último acceso: 13 08 2023].