UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y
SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE MEDICIÓN DE LAS VARIABLES CINEMÁTICAS DE LA MANO PARA LA CLASIFICACIÓN DE PRESAS

CARLOS GONZÁLEZ CARBALLO

2022

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE MEDICIÓN DE LAS VARIABLES CINEMÁTICAS DE LA MANO PARA LA CLASIFICACIÓN DE PRESAS

Autor

CARLOS GONZÁLEZ CARBALLO

Tutores

ÁLVARO GUTIÉRREZ MARTÍN
DIEGO FERNÁNDEZ VÁZQUEZ

2022

Resumen

Las praxias son procesos neurológicos que organizan, planifican y ejecutan un movimiento. Estos movimientos son acciones aprendidas que se realizan en función de la tarea realizar. Las praxias ideatorias están relacionadas con la capacidad de llevar a cabo una secuencia de gestos para el uso de objetos, como efectuar diferentes tareas manuales a través de presas y pinzas con la mano. El estudio de estos movimientos permite la evaluación de las destrezas manuales de diversas patologías, entre las que encuentran algunas enfermedades de origen neurológico. Este análisis consiste en seleccionar varios agarres para observar cómo se ejecuta la sujeción de los objetos escogidos.

Este Trabajo Fin de Grado desarrolla un sistema de detección y clasificación de pinzas y presas. Las pinzas y presas se diferencian en función de los dedos y su disposición. Se ha diseñado e implementado un sistema de grabación donde todas las muestras han sido tomadas con una perspectiva cenital. Lo que pretende es tener el mismo fondo y facilitar el estudio. Se han escogido cuatro tipos de pinzas y presas para ejecutar el asir de diferentes objetos. Los vídeos obtenidos son introducidos en unas redes neuronales profundas, DeepLabCut, que predicen las posiciones cinemáticas de los puntos de interés seleccionados.

Una vez conseguido las posiciones cinemáticas de los marcadores indicados se pretende aumentar la información adquirida. La caracterización ha consistido en extraer la distancia, ángulo, perímetro y área entre los marcadores. Algunos de los parámetros obtenidos no aportan información útil para la clasificación, por ello, se aplican técnicas de limpieza de datos. Por último, se estudian varios algoritmos de clasificación con el fin de encontrar aquel qué mejor clasifica las pinzas y presas capturadas en los vídeos.

Palabras clave: praxias, presas, pinzas, DeepLabCut, asir, capacidad motora, caracterización, clasificación y algoritmo.

Abstract

Praxes are the neurological processes that organize, plan, and execute a movement, regardless of its difficulty. These movements are learned actions that are executed according to the task to be performed. Ideational praxes are related to the ability to execute a sequence of gestures for the use of objects, such as grasping objects. The study of these manual tasks allows the evaluation of manual dexterity of different pathologies, including neurological diseases. The analysis consists of selecting different grips to observe how the grasping of the chosen objects is performed.

This Final Degree Project develops a system for the detection and classification of grips and pinches. Grips and pinches could be differentiated, depending on the fingers used and their position. A recording system has been designed and implemented where all the samples have been taken from the top to have the same background and facilitate the study. Four types of grippers and pinches have been selected to perform the grasping of different objects. The obtained videos are fed into the DeepLabCut system, based on deep neural networks, which predict the kinematic positions of the selected points of interest.

Once the kinematic positions of the indicated markers have been obtained, the aim is to increase the information acquired by characterizing the data. The characterization consisted of obtaining the distance, angle, perimeter and area between the different markers. Some of the parameters obtained do not provide useful information for classification, so data cleaning techniques are used. Finally, the various classification algorithms are studied in order to find the one that best classifies the grips and pinches captured in the videos.

Keywords: praxes, pinches, grasping, DeepLabCut, grasping, motor ability, characterization, classification, and algorithm.

Agradecimientos

En primer lugar, me gustaría agradecerles a mi tutor Álvaro Gutiérrez y a mi cotutor Diego Fernández el apoyo recibido a lo largo del desarrollo del Trabajo de Fin de Grado. Me han aportado grandes conocimientos acerca del estudio de la Inteligencia Artificial y su gran aplicación en el mundo sanitario.

También me gustaría agradecer la colaboración de todos los miembros del departamento del Robolabo de la Escuela Técnica Superior de Ingeniería de Telecomunicaciones. Por supuesto, me gustaría dar las gracias a todos mis amigos que han estudiado la carrera conmigo, que me han apoyado y motivado en las decisiones tomadas a lo largo de estos años.

Por último, me gustaría hacer una mención especial a las personas más importantes, a mi familia. A mis padres por haberme ofrecido siempre la mejor educación y por haberme motivado en seguir con mi desarrollo personal. Y a mis dos hermanos por seguir discutiendo cuál de las tres ingenierías es la mejor.

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

BBDD: Base de Datos.

CDR: Clinical Demencial Rating.

CNN: Convolutional Neuronal Network

DFTc: Demencia Frontotemporal variante conceptual.

DL: Deep Learning.

DLC: DeepLabCut.

DTA: Demencia Tipo Alzheimer.

GUI: graphical user interface.

IA: Inteligencia Artificial.

KNN: K Nearest Neighbor

ML: Machine Learning.

MLP: Multi-Layer Perceptron.

PC: Principal Component.

PCA: Principal Component Analysis.

RL: Representation Learning.

SVM: Support Vector Machine.

Capítulo 1

Introducción

Diariamente usamos nuestras manos para ejercer diferentes movimientos, entre ellos se encuentra la manipulación o sujeción de objetos. Varios estudios clínicos han hecho hincapié en el análisis de las praxias, ya que es el proceso que ocurre en el sistema nervioso central que planifica y organiza las acciones a realizar con el fin de coger objetos [1].

El sistema nervioso central planifica y organiza los movimientos y ajustes postulares necesarios para ejecutar la acción, además es ejecutada a través de la activación de los músculos involucrados en la tarea correspondiente. Un daño en el sistema nervioso central, producido por enfermedades o traumatismos, puede dar lugar a alteraciones en la ejecución y secuenciación de los movimientos, a estos trastornos se les denominan apraxias [2].

El desarrollo de un sistema capaz de clasificar varios agarres pretende ayudar a los análisis de las pinzas y presas. Con el fin de evaluar la destreza manual de diferentes enfermedades. Este estudio aporta información a aquellas personas que sufren de alguna enfermedad neurológica porque tienden a presentar alteraciones en la capacidad de elaborar un plan motor [3]. Se pretende dar apoyo en el actual Trabajo Fin de Grado con la creación de un sistema automatizado de clasificación de presas y pinzas.

1.1. Praxias

La praxia es la capacidad motora de analizar, planificar y ejecutar un movimiento. Existen diferentes tipos de praxias en función de la tarea a realizar, los cinco tipos principales son [4]:

- Ideomotoras: es la ejecución de movimientos simples, son movimientos que se producen ante un estímulo o una orden.
- Ideatorias: se centra en la ejecución del uso de los diferentes objetos.
- Visoconstructivas: son acciones que involucran la parte creativa.
- Faciales: gestos de boca, ojos o labios.
- Vestido: es aquella que se encarga de ejecutar la acción de vestirse.

1.1.1. Apraxias

Las apraxias surgen de la complicación de poder realizar los movimientos aprendidos. Las apraxias pueden darse en personas con demencia o daño cerebral. Esto impide la ejecución normal del movimiento o manipulación de cualquier objeto. Hay diferentes tipos de apraxias, una apraxia por cada tipo de praxia. Es decir, existe las apraxias ideomotoras, las apraxias ideatorias, las apraxias del vestido, etc. Las anomalías de las praxias ofrecen información de los pacientes, siendo importante a la hora de determinar la enfermedad que padecen y su gravedad.

En 2015 se realizó el estudio 'Déficit en la producción motora y severidad de la demencia en la enfermedad de Alzheimer y la demencia frontotemporal' [5]. En este estudio se analiza la correlación entre la demencia tipo Alzheimer (DTA) y demencia frontotemporal variante conductal (DFTvc) con las apraxias. Para ello, se trató de buscar una relación entre las apraxias y el grado de demencia en los pacientes.

En dicho estudio, se diseñó un modelo de sistema de procesamiento de praxias (ver Figura 1.1). El modelo encontró una relación en la que la capacidad motora depende del ingreso visual o táctil de objetos, del ingreso gestual y del ingreso auditivo verbal. El praxicón es la representación motora espacio-temporal, hay dos tipos de praxicones:

- Praxicón de entrada, se considera así cuando contienen gestos que han sido realizado previamente y están integrados con normalidad.
- Praxicón de salida: indica los pasos a seguir para realizar dichos gestos, está información ha sido aprendida.

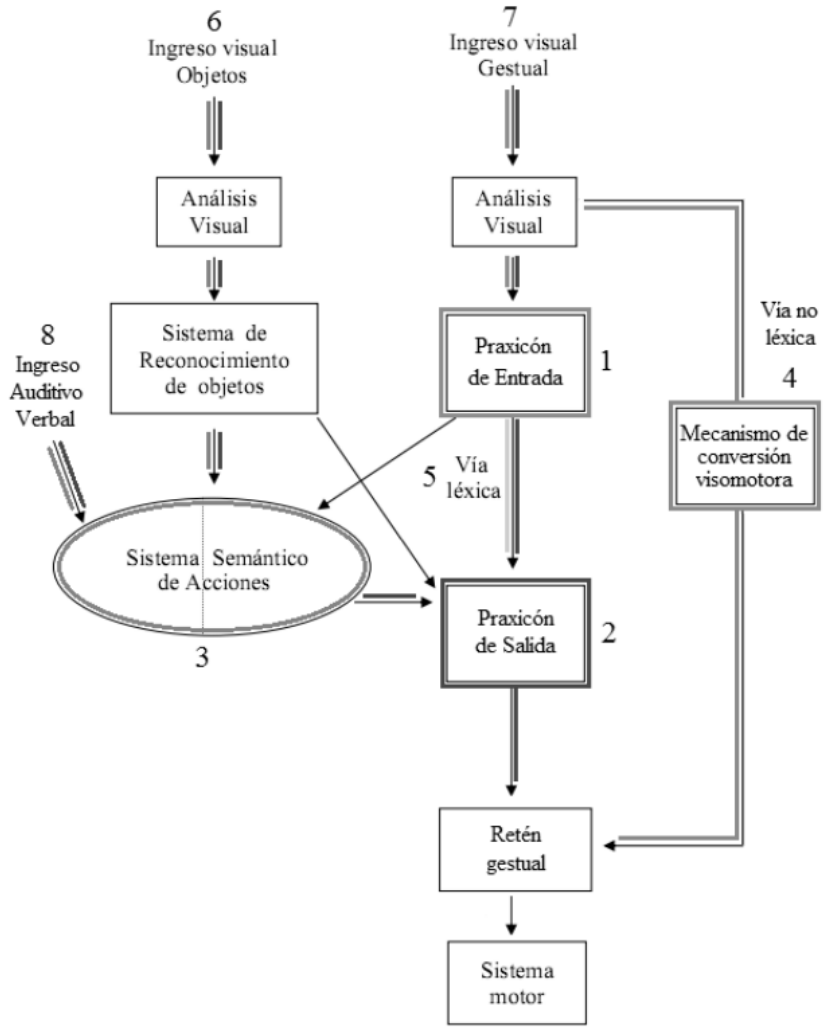


Figura 1.1: Modelo del sistema de procesamiento de praxias de González Rothi et al. (1991, 1997) con los posibles patrones de rendimiento práxico [5].

En el experimento un grupo de personas realizaron nueve actividades, se cuantificó el número de pacientes que presentaron alteraciones según el tipo de enfermedad que presentaban (ver Tabla 1.1). Los posibles tipos de alteraciones a clasificar eran:

- **Alteraciones**: porcentaje de personas que al menos presentan una alteración en una de las nueve pruebas realizadas.
- **Herramientas**: porcentaje de personas que presentaron una alteración en la ejecución del uso de herramientas.
- **Imitación de gestos**: porcentaje de personas que presentaron una alteración en la imitación de gestos familiares

Enfermedad	Nº de Personas	Alteraciones en algunas de las 9 pruebas	Alteraciones por el uso Herramientas	Alteraciones por Imitación de gestos
DTA	50	90 %	78 %	72 %
DFTvc	32	69 %	66 %	53 %

DTA Demencia de tipo Alzheimer
DFTvc Demencia Frontotemporal variante conductal.

Tabla 1.1: Porcentaje de alteraciones según el tipo de enfermedad.

Cada una de estas praxias realizadas se midieron mediante la escala Valoración de Clínica Demencial (CDR del inglés *Clinical Demential Rating*). La escala CDR puede tomar valores de 0 a 3, donde $CDR = 0$ significa que estas sano, $CDR = 0{,}5$ demencia cuestionable, $CDR = 1$ demencia leve $CDR = 2$ demencia moderada y $CDR = 3$ demencia grave [6]. Se llegó a la conclusión de que, a mayor desarrollo de demencia, mayor sería el grado de la apraxia. Por otro lado, la ejecución de gestos en la utilización de herramientas fue la praxia que más información proporcionaba para poder clasificar a los pacientes.

Junto a este hay otros estudios que demuestran la importancia de estudiar las praxias, las relacionadas con la manipulación de herramientas y objetos [7], [8]. Por ello, en este Trabajo Fin de Grado se estudian las praxias ideatorias.

1.1.2. Praxias ideatorias

Las praxias ideatorias son aquellas que hacen referencia a la capacidad de una secuencia de acciones motoras coordinadas cuando se hace uso de cualquier tipo de herramienta u objeto. Una praxia ideatoria está formada por tres fases [9]:

1. **Alcance del objeto**: se sufre una aceleración al comienzo de la acción y una deceleración cuando se acerca al objeto.

2. **Asir el objeto y alzarlo**: es el momento en el que se decide cómo colocar la mano y poder adaptarnos al objeto o situación correspondiente, es decir, esta acción se basa en analizar las dimensiones del cuerpo y la tarea a realizar con este.

3. **Transporte y suelta del objeto**: para poder aguantar los objetos hay que ejercer la tensión correcta para que someta al cuerpo la suficiente presión para que no se caiga.

El alcance y uso de los diferentes herramientas u objetos de forma correcta necesita que no exista ninguna alteración en las praxias ideatorias. Algunos de los ejemplos que implican una praxia ideatoria es peinarse, lavarse los dientes, cortar comida con un cuchillo y tenedor, entre otras. Al igual que en el resto de praxias, en las praxias ideatorias hay varias habilidades implicadas, entre las que cabe destacar: primero el

conocimiento de la función de las herramientas u objeto en cuestión, segundo conocer la acción a ejecutar y tercero el proceso que lleva realizar dicho movimiento.

Por tanto, la ejecución de una presa y pinza conlleva el proceso de una praxia ideatoria. En este Trabajo Fin de Grado se ha planteado la posibilidad de automatizar el proceso de clasificación de pinzas y presas con el fin de facilitar el trabajo a los evaluadores.

1.2. Objetivos

El principal objetivo de este Trabajo Fin de Grado es desarrollar un sistema automatizado capaz de analizar las diferentes presas y pinzas realizadas y poder clasificar y diferenciarlas entre sí.

De esta manera, se pretende facilitar la evaluación de la destreza manual en diferentes patologías, automatizando el proceso mediante el desarrollo de un algoritmo. Este podría ser el comienzo de un sistema capaz de detectar anomalías que pudiese acelerar el diagnóstico precoz.

1.3. Organización del documento

El actual Trabajo Fin de Grado está dividido en los siguientes Capítulos:

1. **Capitulo 1**: se define el concepto de las praxias y apraxias.
2. **Capitulo 2**: se ha introducido el contexto histórico del estudio de las presas y pinzas debido a su gran relación con las praxias ideatorias. También se incluye la definición de Inteligencia Artificial junto a las explicaciones de Machine Learning y Deep Learning.
3. **Capitulo 3**: primero se incluye el proceso de creación de la maqueta de grabación, donde se han grabado los vídeos. Segundo se explica el procedimiento seguido para el procesado de los vídeos. Tercero, se extrae y se analiza la información necesaria de los vídeos. Y por último, explicación de los diferentes algoritmos de clasificación empleados.
4. **Capitulo 4**: resumen de los resultados obtenidos para cada uno de los algoritmos, con una posterior comparación entre los resultados.
5. **Capitulo 5**: se han extraído las conclusiones del actual Trabajo Fin de Grado y sus futuras mejoras con el fin de mejorar el sistema.

Capítulo 2

Estado del arte

2.1. Estudio de las presas y pinzas

La manipulación de los objetos es una tarea que se realiza diariamente. El estudio de las presas fue iniciado por Alexander Graham Bell, en 1833, quien se preguntó si la mano era funcional sin contar con ninguna de las extremidades del cuerpo [9].

Las presas y pinzas son las manipulaciones de objetos y herramientas. Se clasifican en función de los dedos de la mano implicados a la hora de asir las herramientas y su posición. Cuando un objeto es agarrado se analiza la presión ejercida para mantenerlo en tensión y que este no se vea perjudicado.

Existen varios tipos de presas que se puede diferenciar entre presas digitales, palmares y centrales (Figura 2.1) [9].

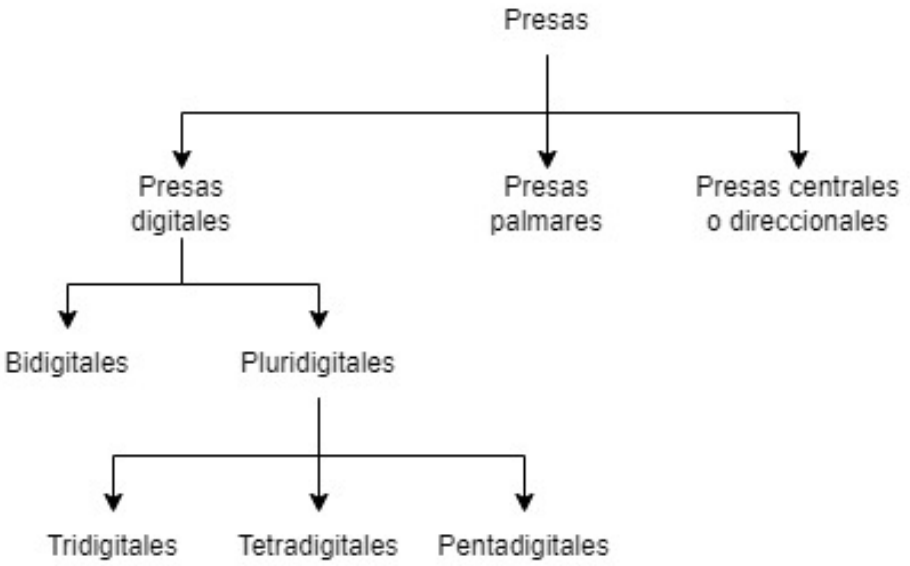


Figura 2.1: Esquema de presas.

- **Las presas digitales** son las que involucran los dedos. Un ejemplo es cuando se manipula los objetos de pequeñas dimensiones como la tapa de un bolígrafo o una aguja.
- **Las presas palmares** son aquellas que emplean los dedos y la palma de la mano. Por ejemplo, cuando se realiza el alcance de un vaso.
- **Las presas centradas o direccionales** son las usadas cuando los objetos son alargados como un destornillador o tenedor, en el que para facilitar la sujeción se efectúa por el centro.

Las **presas con gravedad** son clasificadas aparte, ya que se encargan de utilizar la mano como soporte, como cuando se mantiene por debajo un plato con la palma de la mano.

Las cuatro presas seleccionadas que se analizan a lo largo del Trabajo Fin de Grado, debido a su habitual estudio en diferentes escalas clínicas, son las siguientes [10]:

- **Presa tetradigital**: interviene el primer, segundo, tercer y cuarto dedo a la hora de asir objetos con firmeza. Se diferencian tres tipos de presas tetradigitales [11]:
  - tetradigital de pulpejo: se utiliza cuando el objeto es esférico.
  - tetradigital de pulpejolateral: se emplea cuando se desenrosca una tapa.
  - tetradigital de pulpejo pluridigital: se usa para sujetar objetos delgados como un bolígrafo.

  Esta última, es la escogida para estudiar este tipo de presas tetradigitales (ver Figura 2.2).

Figura 2.2: Presa tetradigital de pulpejo pluridigital.

- **Presa palmar o pluridigital**: este tipo de presas se centra en el agarre que usa la palma de la mano y todos los dedos, menos el primer dedo, que dependerá de la aplicación. Un ejemplo claro es el agarre de un vaso, en el que se realiza una presión hacia dentro con los dedos y la palma para que este no se caiga (ver Figura 2.3).

Figura 2.3: Presa palmar.

- **Presa subterminal**: involucra solo el primer y segundo dedo. Esta presa requiere una mayor precisión, ya que se trata de coger objetos delgados y pequeños, como una aguja o en nuestro caso, una tapa de un bolígrafo, como se muestra en la Figura 2.4.

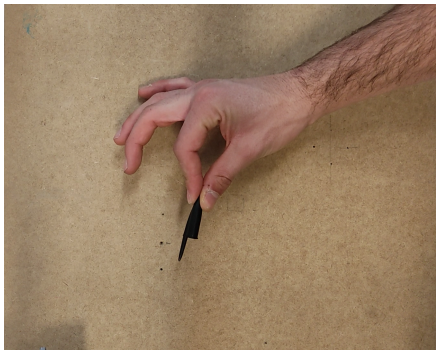


Figura 2.4: Presa subterminal.

- **Presa pulpolateral**: se realiza sujetando el objeto entre el primer dedo y la cara lateral del segundo dedo, un ejemplo claro es el de sujetar una llave (ver Figura 2.5).

Figura 2.5: Presa pulpolateral.

2.2. Inteligencia Artificial

Se considera el comienzo de la Inteligencia Artificial (IA) en 1950 cuando Alan Turing se preguntó '¿Pueden las máquinas pensar?' en su libro de 'Cuputhing Machinery and Intelligence'. A lo largo del siglo XX se empezaron a plantear y desarrollar nuevas máquinas donde era implementada la IA. Éstas eran capaces de tomar decisiones a partir de la información que se le proporcionaba [12].

La IA es capaz de resolver problemas o tareas que las máquinas tradicionales son incapaces debido a que esta puede razonar, ya que aprende la tarea a resolver. La IA se puede organizar en niveles (ver Figura 2.6).

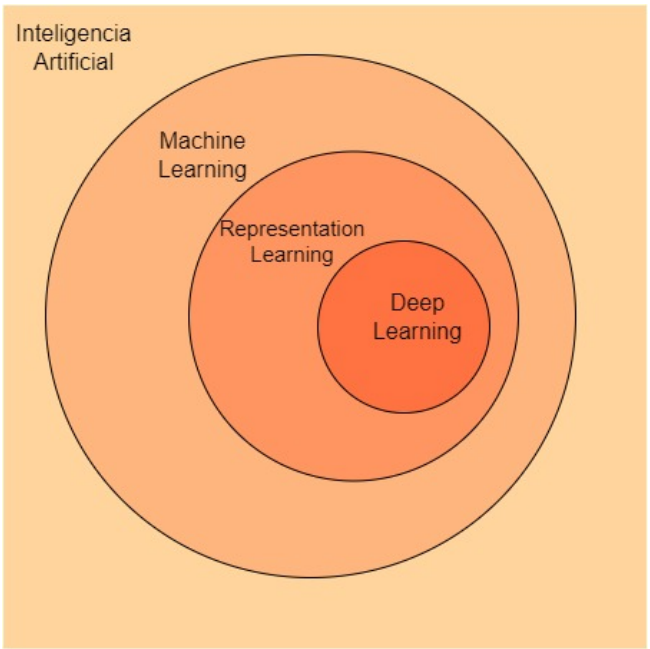


Figura 2.6: Esquema de inteligencia Artificial.

- Machine Learning (ML):es una rama de la IA que pertenece al lenguaje computacional, donde se implementan algoritmos con estadísticos matemáticos. Estos algoritmos son supervisados y generan una clasificación y predicción a partir de los datos [13].
- Representation Learning (RL): es la representación gráfica de la clasificación realizada en Machine Learning. [14]. En algunas situaciones proporcionan información adicional y permite obtener más conclusiones.
- Deep Leaening (DL), es la rama de Machine Learning que está basada en redes neuronales. En algunas situaciones este conjunto de neuronas llega a tener un comportamiento similar al comportamiento de la mente humana. Deep Learning ha conseguido mejorar la automatización de algunas aplicaciones y servicios, incluso evita hacer uso de la intervención humana.

Por una parte, se puede decir que el ML aprende a partir de datos, mientras que por la otra DL es capaz de aprender por sí solo sin la intervención humana. Uno de los inconvenientes que presenta DL frente a ML es que necesita una gran cantidad de datos para obtener buenos resultados [15].

En este Trabajo se han recogido un total de 157 vídeos, es decir, la Base de Datos cuenta con 157 instancias. La herramienta DeepLabCut a pesar de utilizar DL es capaz de lograr buenas predicciones de posiciones cinemáticas para un número bajo de vídeos entrenados. Para la posterior clasificación de pinzas y prisas grabadas se han utilizado diferentes algoritmos de ML y DL.

Capítulo 3

Proceso

En los anteriores Capítulos se ha visto la importancia del estudio de las praxias ideatorias (ver Capítulo 1) y la evaluación de las pinzas y presas (ver Capítulo 2), así como las ventajas encontradas en la IA. En este Capítulo se explica cómo se ha desarrollado el sistema automatizado de clasificación de presas y pinzas.

La Figura 3.1 hace referencia al proceso de creación del sistema. Se ha diseñado un entorno de grabación, en el cual se han tomado todas las muestras de los vídeos a clasificar, que ha permitido la Base de Datos (BBDD). El fin de desarrollar este entorno de grabación es reducir el ruido de los vídeos, ya que se ha empleado siempre el mismo fondo y facilitar su estudio.

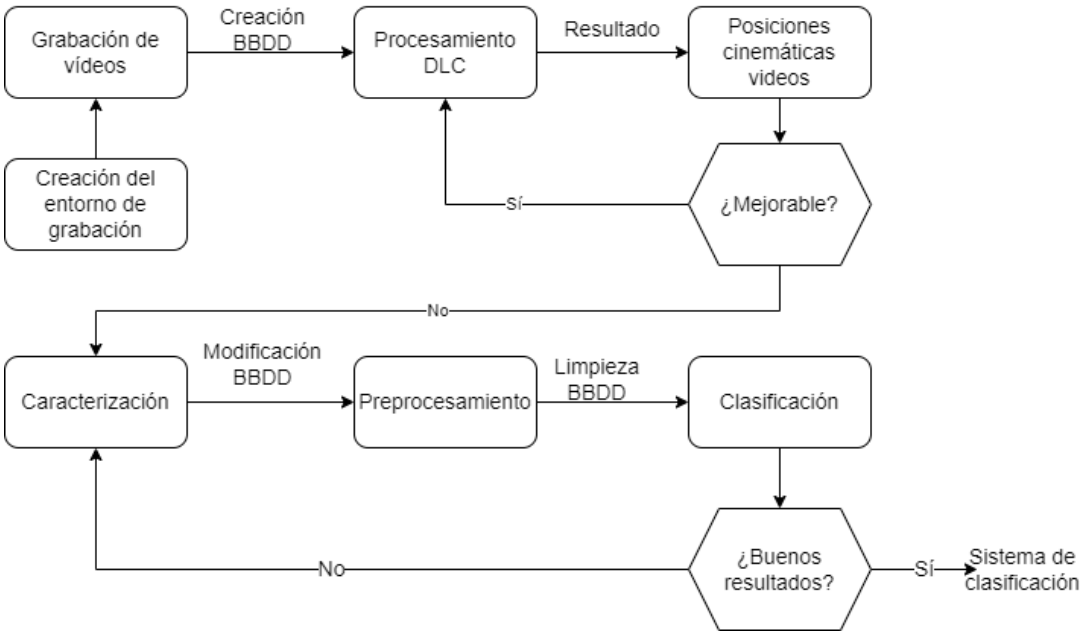


Figura 3.1: Esquema del proceso.

En cuanto a los vídeos procesados en la herramienta de DeepLabCut, se proporciona una estimación de las posiciones cinemáticas de los puntos de interés. Estos marcadores se han situado en las articulaciones de los dedos y en el centro de la muñeca. Con el fin de extraer una mayor información a partir de las posiciones

cinemáticas en cada instante del vídeo, se ha estimado la distancia, ángulo, área y perímetro entre los marcadores. Una vez extraídas estas características, en cada uno de los instantes de cada vídeo se ha obtenido la media, varianza o desviación típica que ha sido recopilado en la nueva Base de Datos (o Data Frame, estructura de datos bidimensional). Este nuevo Data Frame es utilizado para estudiar los diferentes algoritmos de clasificación y analizar aquel que proporciona un mejor resultado.

3.1. Grabación de vídeos

Se ha diseñado un sistema de grabación desde el cual se han realizado todos los vídeos. Se desarrolló con el objetivo de grabar los vídeos con una mayor facilidad con el móvil. El sistema de grabación cuenta de los siguientes elementos:

- La **Plataforma** se diseñó con la herramienta 'Autodesk Inventor' (ver Figura 3.2). Se ha construido con una tabla de madera ($50x50cm^2$) como superficie y perfiles de aluminio ($3x3cm^2$) en las esquinas. Además, estos perfiles se han unido entre sí, con el uso de más perfiles de aluminio, para proporcionar una mayor estabilidad.

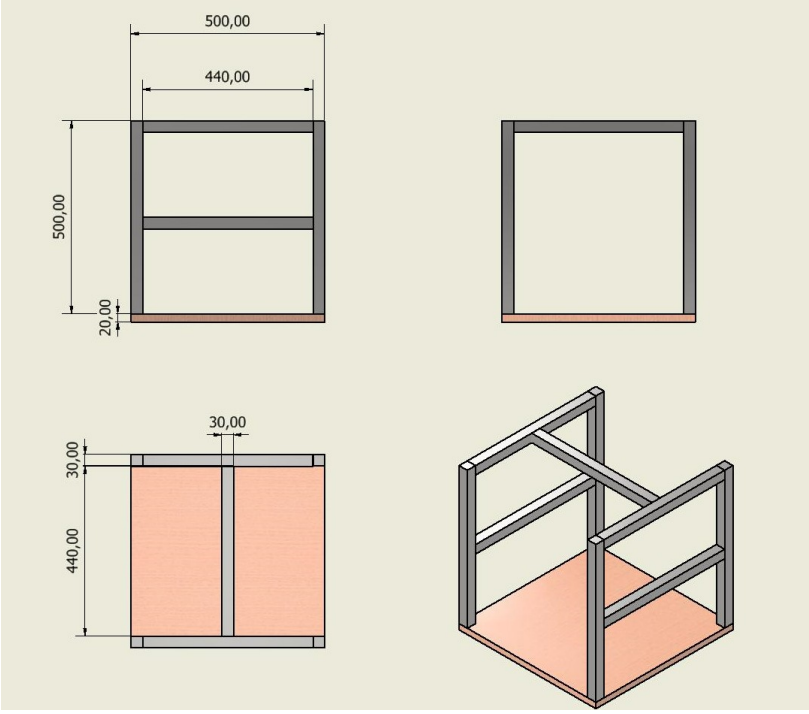


Figura 3.2: Plataforma de grabación.

El material total empleado, comprado en la tienda online de motedis se muestra en la Tabla 3.1 [16].

Material	Número
Tabla de madera	1
Perfiles de aluminio	9
Escuadra	14
Tornillos	28
Tuercas para ranuras	28

Tabla 3.1: Materiales.

- **Soporte del móvil**: este diseño pretende facilitar la sujeción del móvil para poder realizar la grabación de los vídeos desde la parte superior de la plataforma. Para su diseño también se utilizó la herramienta 'Autodesk Inventor' (ver Figura 3.3). La pieza fue creada en una impresora 3D, empleado el filamento como material.

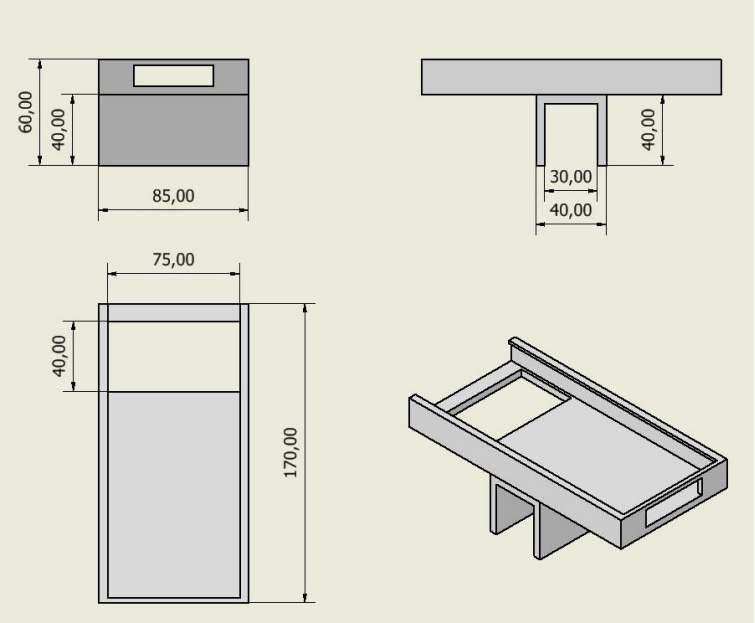


Figura 3.3: Soporte móvil.

El motivo de este diseño fue tener la posibilidad de tener un mayor grado de libertad y poder adaptar el soporte a cualquier dispositivo móvil. Para ello, se ha diseñado cumpliendo con las medidas estándares de los móviles. Se ha tenido en cuenta que no todos los dispositivos tienen la cámara en la misma posición por ello se ha dejado un hueco lo suficientemente amplia para evitar que esta esté obstaculizada. Se ha habilitado el acceso de un cable para permitir la carga en caso de batería baja y transferencia de vídeos sin tener que realizar el desmontaje. El soporte del móvil está diseñado para tener una perspectiva cenital, por tanto, su diseño incluye la adaptación a los perfiles de aluminio situados en la parte superior de la plataforma.

3.2. Procesamiento de los vídeos con DeepLabCut

DeepLabCut (DLC) es una herramienta desarrollada en el laboratorio Mathis Laboratory en 2018 por Alexander Mathis y Mackenize Mathis [17]. La herramienta ha sido diseñada para analizar los comportamientos de los animales sin hacer uso físico de marcadores y analizar así los comportamientos de biomecánica, genética y neurociencia [18]. DLC se basa en redes neuronales profundas. Para conseguir un resultado comparable con el etiquetado humano se necesita un entrenamiento donde la BBDD incluya varios vídeos etiquetados. Cuanto mayor sea el número de los datos de entrenamiento, mejor será la predicción que se obtienen.

DLC está basada en el lenguaje Python de código libre. En el Apéndice C.1.1 se explica cómo realizar su instalación. DLC es una aplicación capaz de trabajar con vídeos en 2D y 3D. La red generada puede ser utilizada en tiempo real. Esta aplicación es capaz de trabajar con el fondo variante e incluso si la cámara no se encuentra estática, aunque el estudio se hace un poco más complejo. En este Trabajo Fin de Grado se ha simplificado los datos por haber usado siempre el mismo fondo y con el móvil fijo, pero se han empleado la mano de diferentes sujetos para ofrecer variedad.

En la Figura 3.4 se muestra el proceso completo de un proyecto en DLC.

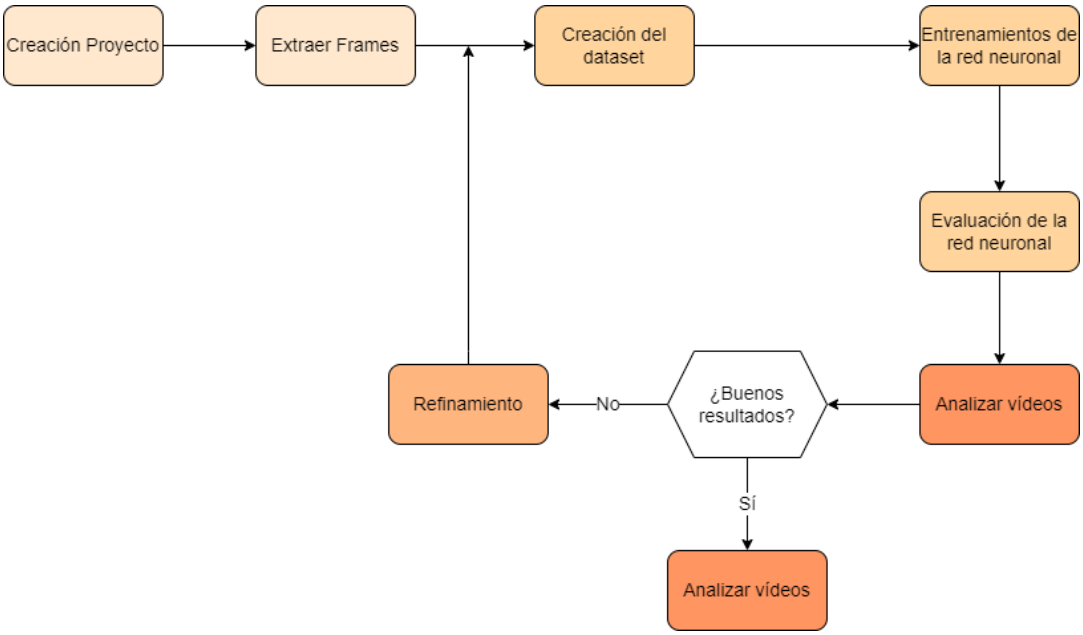


Figura 3.4: Esquema de funcionamiento de DLC [19].

3.2.1. Creación del proyecto

Existen dos maneras de crear un proyecto en DLC, mediante código en Python y el uso la GUI [20] (Apéndice C.1.2). En el proceso de creación del proyecto hay que identificarlo con el nombre del proyecto y el nombre del autor. Cuando el proyecto se crea se crean las siguientes carpetas: ***dlc-models***, ***evaluation-results***, ***label-data***, ***training-dataset*** y ***videos***. Por otro lado, se genera un fichero de configuración llamado ***config.yaml***.

El fichero de configuración ***config.yaml*** tiene una serie de atributos a tratar desde el inicio de la creación del proyecto.

- ***bodyparts***: en esta sección se identifica el nombre de los puntos a analizar.
- ***numframes2pick***: se decide cuantos frames se extraen de cada vídeo.
- ***dotsize***: identifica el diámetro de los puntos a la hora de realizar el etiquetado, como se verá más adelante.
- ***skeleton***: este atributo permite unir los puntos deseados de bodyparts.

3.2.2. Creación y etiquetado de frames

Una vez ha sido creado el proyecto y modificado el fichero de configuración, se procede la extracción de los frames de cada vídeo (Apéndice C.1.3). El número de frames extraídos de cada vídeo, es el especificado en *numframes2pick* del fichero *config.yaml*. Hay dos métodos de selección de frames:

- *uniform*: divide el vídeo y extrae los frames distribuidos en el tiempo.
- *kmeans*: se pretende extraer los frames que presenten una mayor diferencia entre ellos.

Cuando se obtienen los frames, estos son guardados en la carpeta del proyecto, ***label-data***, donde se encuentra una carpeta por cada vídeo con el número de frames especificado. Luego se procede al etiquetado de los marcadores seleccionados en el parámetro de ***bodyparts*** del fichero de configuración *config.yaml*.

A continuación se muestra una captura de las marcas que se han utilizado para identificar las partes con mayor importancia de la mano. Se han escogido la parte distal de la falange distal y las articulaciones interfalángicas de cada dedo y el centro de la muñeca (ver Figura3.5).

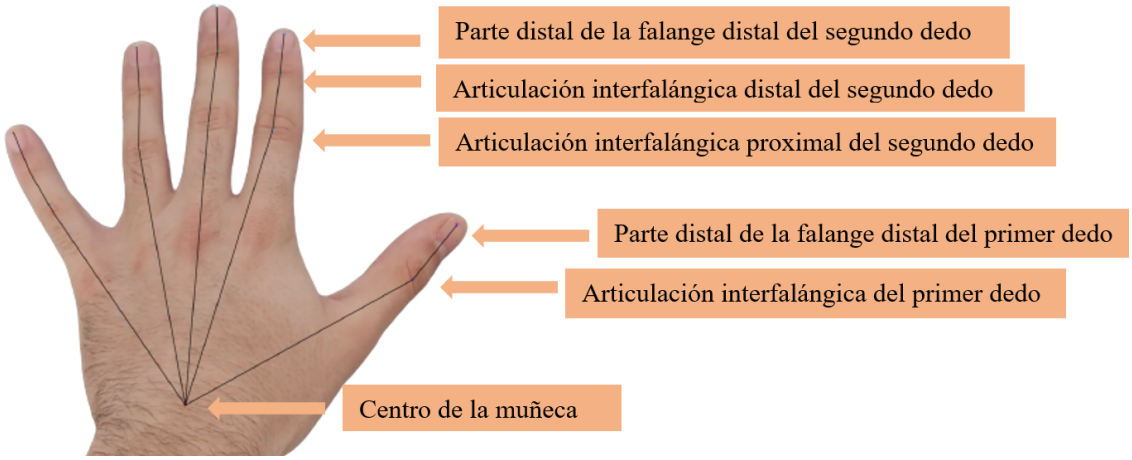


Figura 3.5: Marcadores utilizados en la mano.

3.2.3. Entrenamiento

DLC utiliza redes neuronales convolucionales preentrenadas, las cuales son entrenadas con los frames etiquetados (Apéndice C.1.4).

En este estudio se ha utilizado los siguientes métodos de entrenamiento:

- **ResNet-50** conocido como *Residual Network*, es un algoritmo desarrollado por Kaiming He [21]. Resnet está basado en *Convolutional Neuronal Network* (CNN), es decir, redes neuronales convolucionales, están formadas por varias capas de neuronas que cuentan con *skip connections*. *Skipp connnections* conecta las primeras capas con las posteriores. Esto permite que la información llegue a las neuronas de las capas ocultas con una mayor rapidez, así se consigue aprender desde el inicio del entrenamiento. Por último, ResNet-50 indica que contiene 50 capas ocultas.

- **MobileNet** es una adaptación de ResNet, que es más ligera, tiene una arquitectura de redes neuronales más sencilla. Los resultados tienden a ser buenos aunque no se tenga una alta resolución en los vídeos. Por ello, ante el uso de un dispositivo móvil para la grabación de los vídeos se recomienda utilizar este algoritmo.

Para el método de entrenamiento de ResNet es recomendable utilizar GPU frente a CPU, ya que consigue reducir el tiempo de ejecución del entrenamiento. Ante el algoritmo de MobileNet la GPU no es mucho más veloz que la CPU [22]. En este Trabajo Fin de Grado para el entrenamiento se ha usado la GPU del servidor que se encuentra alojado en el departamento del Robolabo en la Escuela Técnica Superior de Ingeniería de Telecomunicaciones en la Universidad Politécnica de Madrid.

Se genera una carpeta en **dlc-models** por cada entrenamiento realizado, y dentro de cada una de ellas, hay dos subcarpetas, **test** y **train**. Dentro de **train** se encuentra el fichero **pose_config.yaml** que incluye los parámetros de entrenamiento.

El parámetro más importante es el que indica el número máximo de iteraciones a realizar en cada entrenamiento.

3.2.4. Análisis de vídeo

Es uno de los últimos pasos a realizar para completar el desarrollo del sistema de análisis de vídeo. Una vez el sistema ha sido entrenado, es necesario analizar cómo reacciona ante nuevos datos de entrada. Para ello se le proporcionan nuevos vídeos que no han sido procesados antes (Apéndice C.1.5).

Se generan nuevos Data Frames por cada vídeo con las estimaciones de las posiciones cinemáticas de los marcadores indicados. DLC ofrece la posibilidad de crear nuevos vídeos que incluye las predicciones de los marcadores sobre el vídeo original. Esto facilita la inspección visual de las predicciones del sistema en los nuevos vídeos.

En el caso de tener que mejorar los resultados, se puede solucionar mediante la técnica de refinamiento. Es decir, se corrigen las posiciones de los marcadores que no están en el lugar adecuado. Después se genera una nueva iteración y se reentrena el sistema hasta obtener un resultado óptimo.

3.3. Caracterización

Cuando DLC analiza los vídeos devuelve las posiciones cinemáticas de los marcadores. Para poder realizar un buen clasificado de vídeos es necesario extraer más información ya que la obtenida no es suficiente.

Para conseguir obtener una mayor información se han extraído más características de cada frame (distancias, ángulos, perímetros y áreas entre los marcadores). Por último, como interesa el conjunto de vídeos para poder clasificarlo, se ha creado un resumen que recopila las medias de los atributos y desviaciones típicas de todos los vídeos.

3.3.1. Caracterización de los frames

La caracterización de los frames se realiza sobre los archivos generados en la carpeta de los vídeos analizados (los ficheros '.csv'). Estos vídeos son divididos en frames, entonces, para cada frame se estima con un valor de *likelihood* las posiciones x, y de cada marcador. El **likelihod** indica la probabilidad de error de predicción de la posición de cada marcador. Por ello, ante las impurezas encontradas en estos frames se ha aplicado una limpieza de los datos como paso previo a la caracterización.

3.3.1.1. Limpieza de los datos de frames

Se ha realizado una limpieza de los datos tratando de eliminar el ruido y minimizar los fallos de predicción de DLC.

- **Eliminación de información nula**. Se elimina la información proporcionada por la predicción de DLC cuando la mano se encuentra fuera del entorno visible de grabación. A continuación se muestra una gráfica en la que se han representado los valores obtenidos del centro de la muñeca cuando se realiza una presa tetradigital a lo largo del tiempo en el eje x tras aplicar esta limpieza de datos, Figura3.6.

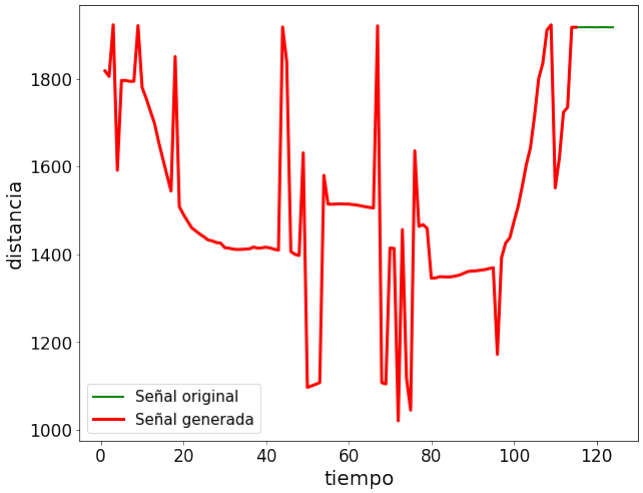


Figura 3.6: Filtrado de valores nulos.

- **Tiempo**. Para la extracción de datos cinemáticos interesa el instante cuando el sujeto ha agarrado el objeto deseado. Por lo tanto, la información que se obtienen sin que la pinza esté en ejecución no son relevantes, por ello se ha decidido eliminar el 25 % de los datos recibidos al principio y el 25 % del final, es decir, se ha utilizado el 50 % de los valores centrales (ver Figura 3.7).

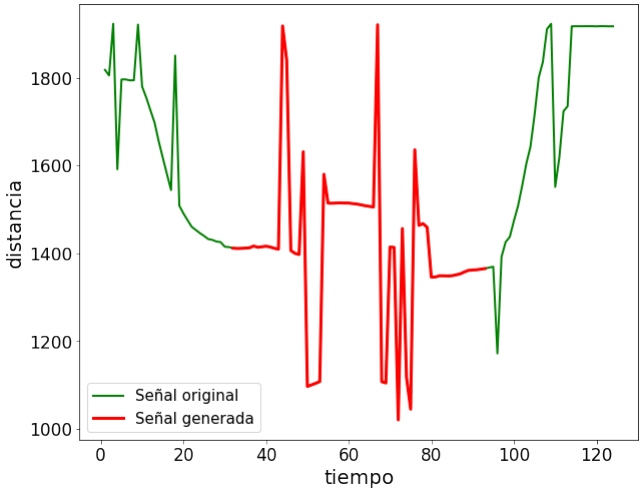


Figura 3.7: Filtrado del tiempo.

- ***Threshold***. El threshold es una probabilidad de haber acertado la posición x, y del marcador en cada instante. Se puede minimizar el error para intentar obtener mejores resultados. Se ha encontrado una condición y una ecuación que permite disminuir estos errores. Si el *threshold* $\leq 0{,}9$ se intenta corregir este fallo con la siguiente Ecuación 3.1:

$$pos_{marcador}[i] = pos_{marcador}[i] * 0{,}8 + pos_{marcador}[i-1] * 0{,}2 \qquad (3.1)$$

  Esto ha permitido corregir las anomalías presentadas por la predicción de datos, como cambios bruscos encontrados para un *Threshold* bajo (ver Figura 3.8).

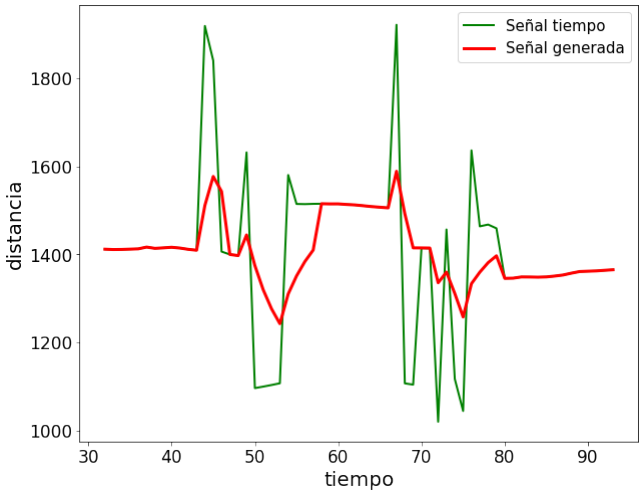


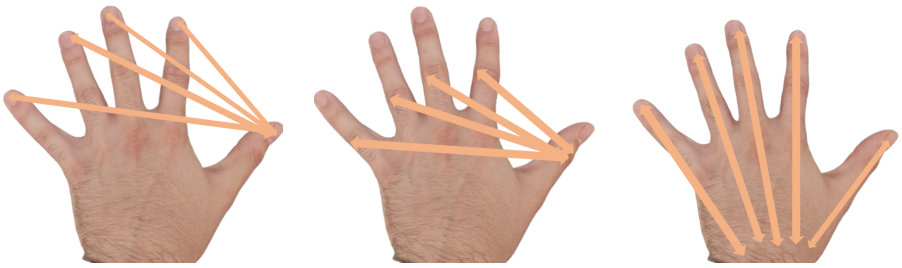
Figura 3.8: Filtrado del threshold.

Tras haber realizado la limpieza de datos, se han creado nuevos Data Frames por cada vídeo. Esto ha permitido mejor la clasificación de los datos.

3.3.1.2. Caracterización de los frames

Las posiciones cinemáticas de los marcadores nos permiten aumentar la información mediante la caracterización de los datos. Es decir, se busca extraer más características de los datos. Esto se hace que cada presa tenga un agarre diferente, que involucra diferentes partes de la mano, depende de si incluye o no la palma o si utiliza unos dedos u otros. Por ello, se ha escogido estudiar la distancia, ángulo, perímetro y área de algunos marcadores.

- **Distancia**. Se ha usado el cálculo de la distancia euclídea en un espacio bidimensional. Se ha obtenido la distancia entre la punta del primer dedo con el resto de los dedos, así como la distancia entre la punta del tercer, cuarto y quinto dedo con la punta del segundo dedo. Las mismas distancias fueron recogidas entre las articulaciones interfalángicas proximales: entre la articulación interfalángica del primer dedo con la articulación interfalángica proximal del resto de dedos, además de la distancia entre la articulación interfalángica proximal del tercer, cuarto y quinto dedo con la del segundo dedo. Otra distancia extraída fue entre la punta de cada dedo con el centro de la muñeca (ver Figura 3.9).



a) *b)* *c)*

Figura 3.9: Distancia entre los diferentes marcadores.

- **Ángulo**. El cálculo del ángulo pretende aportar información de la apertura de la pinza. Tomando como vértice la muñeca, se calcularon los ángulos entre la punta del primer dedo con la punta del segundo y también del tercer dedo, y entre la articulación interfalángica del primer dedo y la articulación interfalángica proximal del segundo y tercer dedo. Tomando como vértice la punta del primer dedo, se han obtenido los ángulos entre la punta y la articulación interfalángica proximal del segundo y tercer dedo.
- **Perímetro**. Se ha calculado el perímetro formado por el centro de la muñeca y los marcadores del primer y segundo dedo.
- **Área**. Se ha calculado el área formada por el centro de la muñeca y los marcadores del primer y segundo dedo.

La caracterización de los frames ha sido realizada mediante la programación de un script generado en Python. Este obtiene la información de los ficheros analizados

por DLC y realiza las operaciones comentadas en este apartado. Se genera un fichero por cada vídeo analizado, recopilando esta información.

No todas las manos son del mismo tamaño, por ello es recomendable realizar una normalización o estandarización de los datos. Las técnicas más utilizadas son [23]:

1. **Normalización Min-Max (feature scaling)**. Esta normalización transforma todos los datos entre [0,1]. La transformación se realiza con la siguiente expresión 3.2:
$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.2}$$
2. **Estandarización (z-score normalization)**. Se estandarizan los datos, para que estos pasen a tener una media igual a 0 y desviación típica igual a 1. Para está normalización se utiliza la Ecuación 3.3:
$$\hat{x} = \frac{x - \mu_x}{\sigma_x} \tag{3.3}$$

A pesar de ello, para el problema de clasificación de presas y pinzas se aplicó la normalización y estandarización de los datos, pero no aportó mucha información. Esto se debió a que se decidió que la caracterización del conjunto de vídeos dependía de la media, varianza y desviación típica de los frames normalizados o estandarizados, no aportaban información útil, por ello, se descartó su uso.

3.3.2. Caracterización de los vídeos

Una vez realizada la caracterización de cada frame de cada vídeo interesa el conjunto de ellos. De cada atributo calculado en los frames de distancia, perímetro y área se han extraído algunos valores estadísticos:

- Media: $\mu = \frac{\sum_i^N x_i}{N}$
- Varianza: $\sigma^2 = \frac{\sum_i^N x_i - \mu}{N}$
- Desviación típica: $\sigma = \sqrt{\sigma^2}$

3.3.2.1. Preprocesamiento de los vídeos

Después de haber obtenido los valores estadísticos de las distancias, ángulos, perímetros y ángulos, se obtiene 67 atributos. Ante esta gran cantidad de datos se reduce la dimensión para mejorar el tratamiento de esta BBDD. Las dos técnicas utilizadas son: eliminación de variables correladas y la reducción de las dimensiones, en concreto con *Principal Component Analysis* (PCA).

- **Eliminación de variables correladas**. Se ha estudiado la relación que existen entre todas las variables, donde se analiza la correlación que existe entre estas. La corrrelación se calcula como:
$$\rho = \frac{E[X * Y] - E[X] * E[Y]}{\sqrt{Var(X) * Var(Y)}} \tag{3.4}$$

Donde X, Y son las variables a estudiar y donde $\rho \in [-1, 1]$. Si $\rho = 0$ las variables son linealmente independientes, mientras que si $\rho = 1$ las variables son linealmente dependientes. Aquellas variables con una ρ cercana a -1 o 1 no aportan información nueva. Con el fin de disminuir la dimensión una de ellas es eliminada. En este caso se ha impuesto la condición cuando ρ sea superior a 0,8. Por consiguiente, se ha transformado la BBDD de 67 atributos a un nuevo Data Frame de 38 variables.

Se calcula la matriz de correlación, donde la diagonal es la varianza de cada una de las variables y el resto son la covarianza entre las diferentes variables. En la Figura 3.10 se muestra un mapa de calor que representa la matriz de correlación tras realizar la transformación de la BBDD.

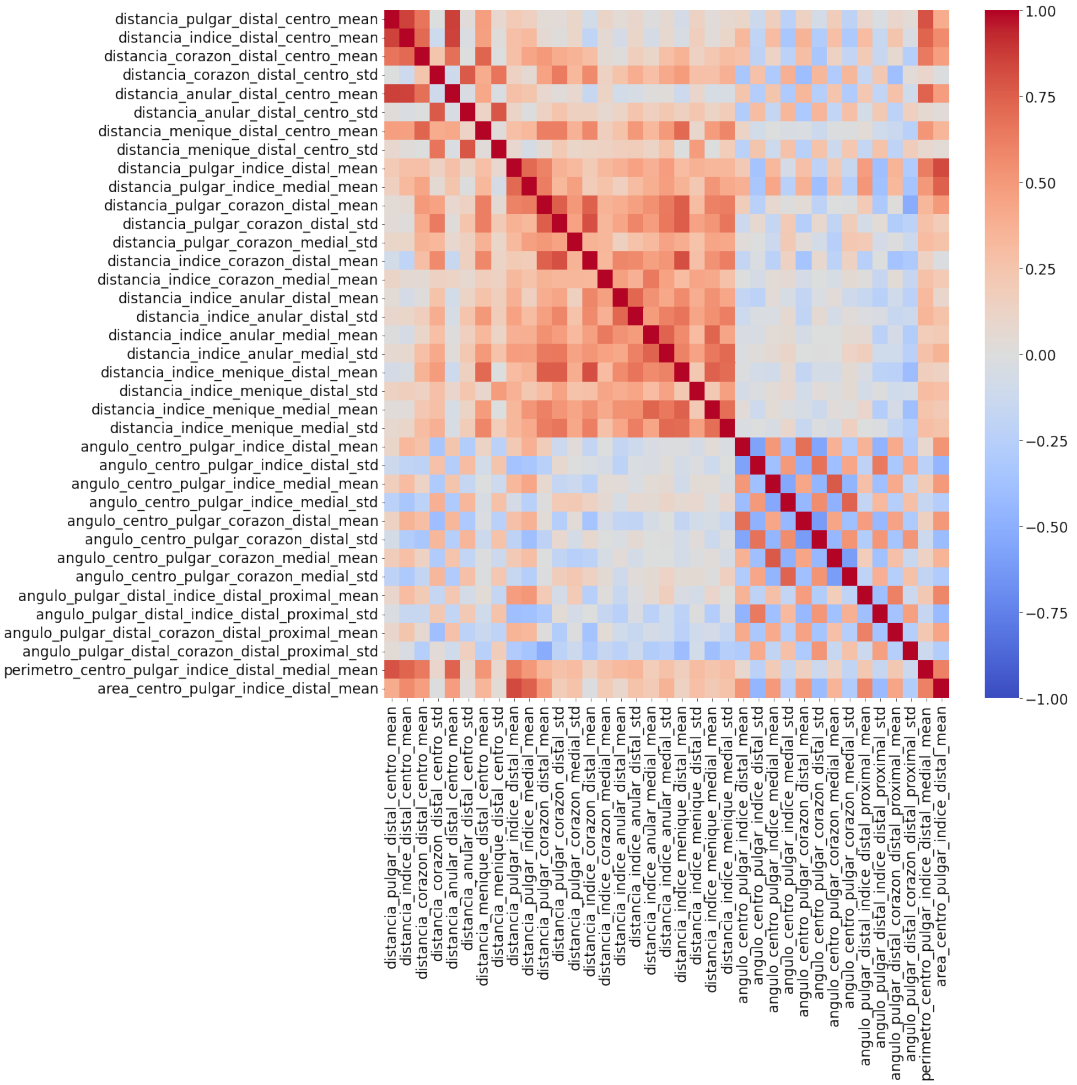
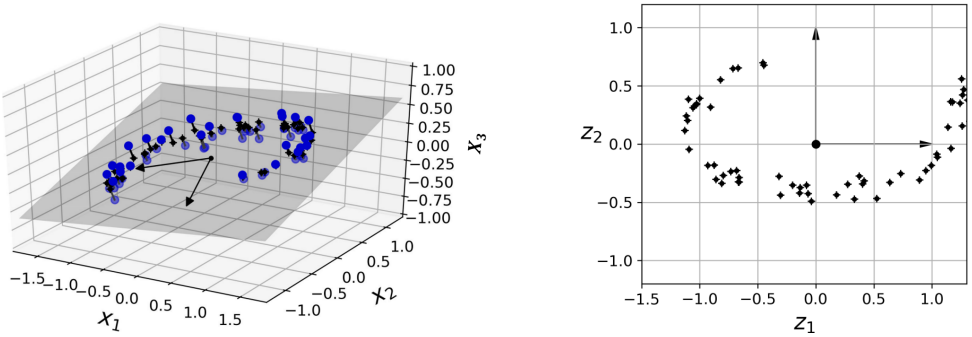


Figura 3.10: Matriz de correlación tras la eliminación de variables.

- **Reducción de dimensiones**: a pesar de haber hecho la reducción anterior, en ML es común que las BBDD tengan demasiados atributos. Tener una gran abundancia de datos puede hacer que el sistema sea lento y complicar la clasificación. Por ello, se utiliza la técnica de reducción de dimensiones. Las BBDD no suelen estar distribuidas uniformemente en todas las dimensiones (es decir, no todos los parámetros participan por igual en las clases) [24]. Por esto se busca un subespacio de la base del espacio en el que las muestras son proyectadas perpendicularmente. Esto se realiza haciendo el producto escalar de las muestras con la matriz de proyección.

  El siguiente ejemplo muestra una reducción del espacio de una base en 3D a 2D (ver Figura 3.11). En este caso se ha buscado un plano al que se le ha realizado la proyección perpendicular de cada muestra que ha generado el nuevo dataset en 2D.



a) A 3D dataset lying close to a 2D subespace. *b) The new 2D dataset after projection.*

Figura 3.11: Reducción de dimensión 2D a 3D [24].

PCA es uno de los métodos más empleado para tratar de reducir la dimensión. La ídea de PCA es buscar aquel hiperplano que es capaz de mantener las direcciones de máxima varianza y por tanto reducir el error cuadrático medio de la distancia entre el dataset original y su proyección. Si las muestras de una variable tienen baja varianza, no se puede extraer mucha información de ellas. Se denomina *Principal Component* (PC) al vector que determina la dirección del plano. Es conveniente conocer cuanta información se pierde con la reducción de dimensiones. La Figura 3.12 representa la varianza capturada para las diferentes dimensiones de subespacio probadas. Escogiendo un subespacio de 15 dimensiones, se mantiene un 91 % de la información. De esta manera se crea un nuevo dataset que en vez de tener 38 atributos ha sido reducido a 15 atributos.

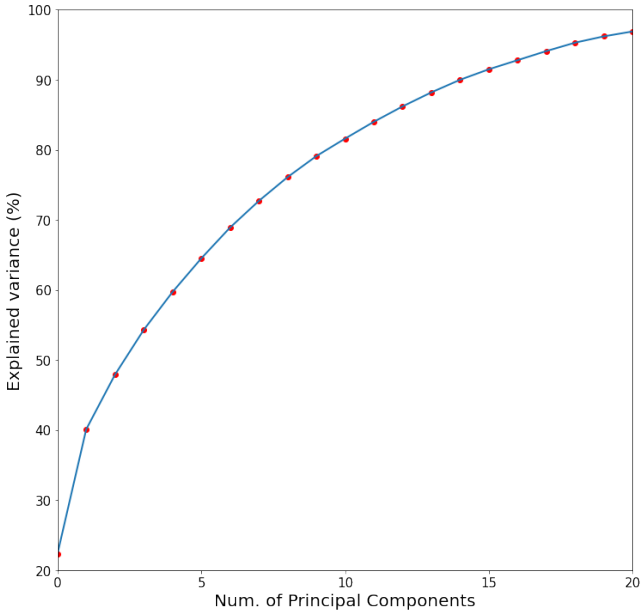


Figura 3.12: Porcentaje de varianza capturada por los PC.

3.4. Clasificación

Tras la caracterización de los datos con su correspondiente preprocesamiento se ha realizado la clasificación. En ML hay muchos algoritmos de clasificación, se ha escogido estudiar algunos de los más comunes: ***Logistic Regresion***, ***K Nearest Neighbors***(KNN), ***Decision Tree***, ***Random Forest***, ***Support Vector Machine*** (*SVM*) y ***Multi-Layer Perceptron*** (*MLP*). Estos algoritmos se encuentran dentro de las librerías de Python, en este caso, las mencionadas pertenecen a Scikit-Learn.

Como paso previo a la clasificación hay que intentar evitar el sesgo en los datos. Primero se mezclan los datos. Segundo, se realiza una división de datos entre Test y Train. Como la BBDD cuenta con pocas instancias, se divide los datos en 70 % Train y 30 % Test.

3.4.1. Logistic Regresion

Este algoritmo de clasificación estima la probabilidad de pertenecer a una clase. La Ecuación 3.5 representa la clasificación binaria en *Logistic Regresion*:

$$\sigma(x) = \frac{1}{1 - e^{-x}} \in [0, 1] \tag{3.5}$$

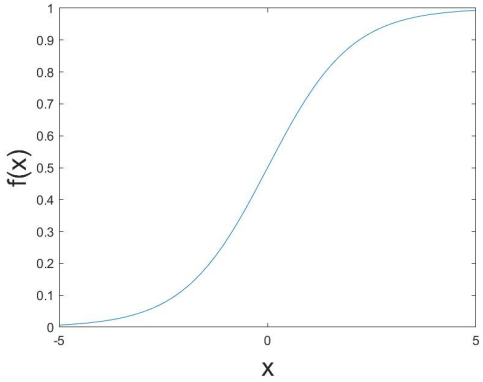


Figura 3.13: Sigmoide.

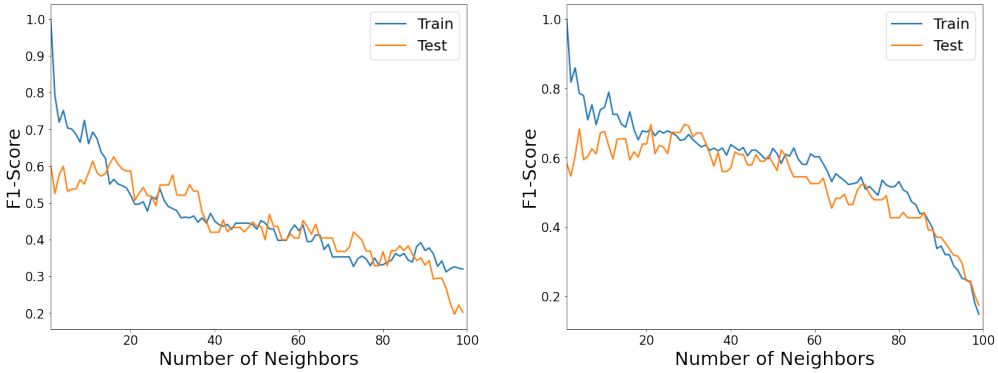
Esto implica que aquellos valores superiores a 0.5 pertenecen a la clase 1 y los que se encuentren por debajo a la clase 0. En otras palabras, realiza una clasificación binaria. En el actual Trabajo el problema se encuentra ante la clasificación de 4 clases (presa tetradigital, presa palmar, presa subterminal y presa pulpolateral), por lo que no es posible utilizar el método anterior. Ante esta situación se desarrolló ***Multonimal Logistic Regresion***, que es una mejora de *Logistic Regresion* que permite la clasificación de más de dos clases [25] (Apéndice C.2.1).

3.4.2. K Nearest Neighbors

KNN trata buscar los K vecinos con una distancia euclídea menor. Para identificar la clase a la que pertenece se hace una votación entre los K vecinos más cercanos y se escoge la que tenga una mayor participación [26].

Hay que tener en cuenta que para $K = 1$ la probabilidad de acertar en Train siempre va a ser el 100 %, ya que 'eres tu propio vecino', pero ante nuevos datos de Test la predicción es baja. Por otro lado, para K grandes, el valor de acertar en Train disminuye notablemente, ya que clasificas en función de la participación de más vecinos. Se ha tratado de buscar el número de vecinos óptimo para esta BBDD, a continuación se muestra cómo evoluciona los porcentajes de acierto tanto de Test cómo de Train para los diferentes números de vecinos tanto para ResNet como para MobileNet (ver Figura 3.14).

El valor óptimo de K vecinos depende de aquel que proporcione un mejor acierto



a) *b)*

Figura 3.14: KNN óptimo para a) ResNet y b) MobileNet.

de los datos de Test. *F1-Score* es la precisión de los datos acertados. Pero, ante varios valores de K vecinos con porcentajes de aciertos muy similares se escogerá aquel que tenga menos vecinos, ya que su estructura es más simple. Para ResNet se ha escogido el número óptimo de vecinos igual 10 y para MobileNet es cercano a 20.

3.4.3. Árboles de Decisión

En la Figura 3.15 se muestra una estructura de un **Árbol de decisión** o ***Decision Tree***.

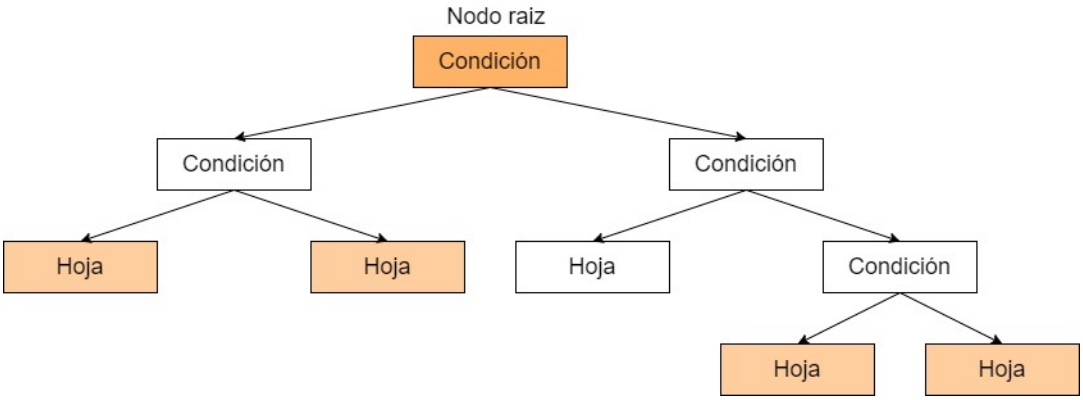


Figura 3.15: Decision Tree.

Un árbol de decisión está formado de un nodo raíz, las ramas, nodos de decisión y las hojas. En cada nodo hay una condición que, en función de si se cumple o no, se hace una división binaria y te desplazas a través de una rama u otra hasta otro nodo o a una hoja [27]. Ante problemas de clasificación, como el planteado, la hoja determina a qué clase se pertenece.

El principal objetivo es simplificar las ramas cuanto antes para llegar a una hoja. El tipo de división de los nodos se realiza especificando el tipo de criterio:

- **Índice de Gini**: indica la pureza de cada nodo, indica en cada nodo como de igual es la participación de las clases en sus nodos hijos.

$$g(K) = 1 - \sum_i [p(C_i|n)]^2 \in [0, 1] \tag{3.6}$$

  Si $g(K) = 1$ entonces se dice que hay una impureza, que no tiene una desigualdad de participación entre el nodo y sus nodos hijos. Por el contrario, $g(K) = 0$ indica que el nodo hijo es puro, ya que mantienen la igualdad de participación de las clases.

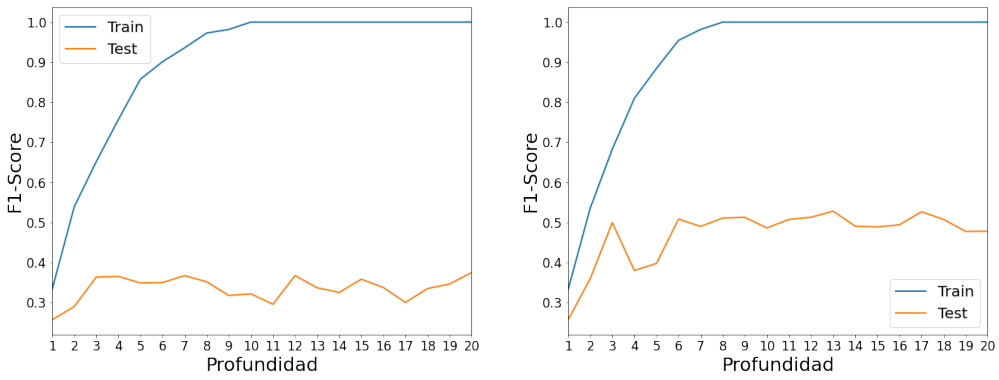
- **Entropía**: indica el desorden que existe en cada nodo, es decir, la aleatoriedad de los datos. Por ello, se busca aquel nodo que minimice la entropía.

$$S(K) = -\sum_i p(C_i|n) * logp(C_i|n) \tag{3.7}$$

- $p(C_i|n)$ indica la probabilidad de pertenecer a una de las clases cuando se sabe que pertenece en al nodo n.

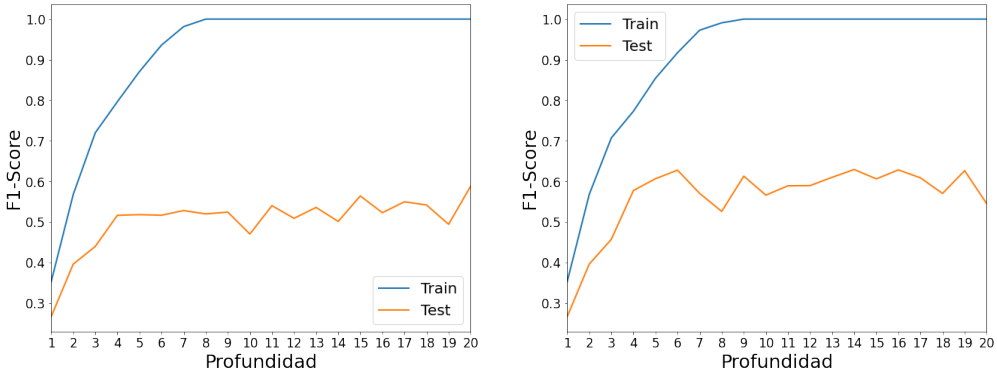
Para la implementación de este algoritmo se ha utilizado tanto el criterio de Gini como el de entropía para poder realizar una comparativa de resultados entre ambos criterios. Además, hay que especificar la profundidad máxima del árbol (Apéndice C.2.3).

A continuación, en la Figura 3.16 y Figura 3.17 se analiza la reacción del sistema ante la evolución de la profundidad del árbol para los dos criterios. En la Figura 3.16, para ResNet se obtienen resultados bastante bajos, el mejor resultado se obtiene cuando se selecciona el criterio de entropía y se escoge una profundidad del árbol de 3, con el fin de minimizar la sobreespecialización. En el caso de usar MobileNet también presenta mejores resultados con el criterio de entropía y con una profundidad de 4 (ver Figura 3.17). Se observa, que MobileNet predice merjo que ResNet.



a) *b)*

Figura 3.16: Decision Tree óptimo para ResNet con a) Gini b) Entropía.



a) *b)*

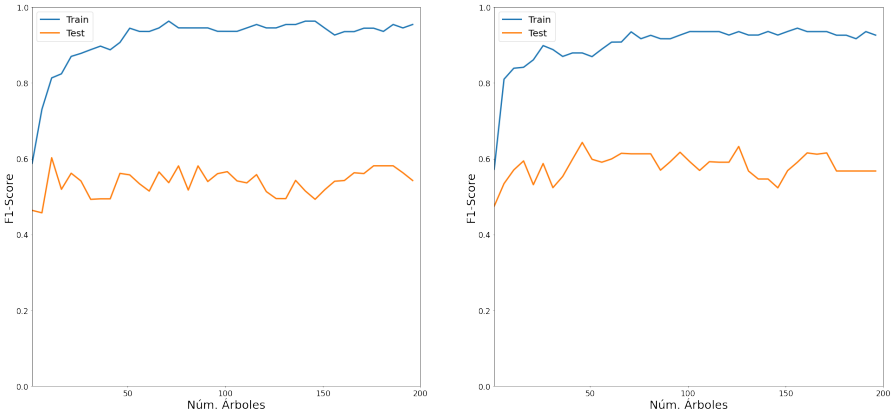
Figura 3.17: Decision Tree óptimo para MobileNet con a) Gini b) Entropía.

Se observa, que con el algoritmo, *Árboles de Decisión*, se obtienen resultados bajos para una profundidad baja y una sobreespecialización para una profundidad alta, por ello, se buscó un compromiso entre la precisión de los aciertos de los datos de Train y Test. La sobreespecialización es debido a que el algoritmo genera los nodos y ramas a partir de los datos de entrenamiento y esto puede generar que los datos se encuentren sesgados.

3.4.4. Random Forest

El ***Random Forest*** o **Bosque aleatorio** es un conjunto de árboles de decisión, cada uno diferente. Cada árbol utiliza un conjunto de variables aleatorias de la BBDD. Esto consigue eliminar el problema encontrado en *Decision Tree* de que los datos estén sesgados. Para mejorar esta situación, los árboles de Random Forest utilizan distribuciones homogéneas, es decir, para un mismo número de instancias en todas las clases [28]. En particular cuando se trata de un problema de clasificación, como el abordado, cada árbol del bosque da un resultado de clasificación, que predice a que clase pertenece la muestra. Luego se hace una votación entre todos los resultados de cada árbol y la clase que obtenga una mayor participación será la elegida. Donde es necesario indicar el número máximo de árboles en el bosque, la máxima profundidad del árbol y el número máximo de hojas (Apéndice C.2.3).

Para los valores de $max\_depth = 8$ y $max\_leaf\_nodes = 8$, el valor óptimo de árboles es 11 para ResNet y 14 para MobileNet , tal y como se contempla en la Figura 3.18.



a) *b)*

Figura 3.18: Random Forest para a) Resnet y b) MobileNet.

3.4.5. Support Vector Machine

Support Vector Machine o **SVM** es un algoritmo que busca aquel hiperplano que mejor separa las instancias [29].

La Figura 3.19 muestra un ejemplo en el que se han representado dos clases, la verde y azul y SVM realiza una separación mediante un plano lineal. Existen infinitos planos que cumplen con la separación de las dos clases, cuanto más lejos esté, mejor generalizará. Los vectores de soporte o *suppport vector* son las instancias que ha considerado el algoritmo como las adecuadas para crear la frontera de decisión o *Optimal hiperplane*.

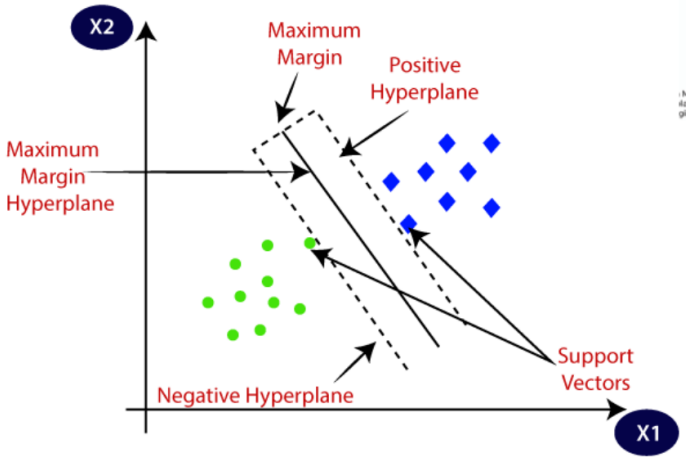


Figura 3.19: Hiperplano en SVM [30].

La Figura 3.19 sirve como ejemplo, pero en la práctica la BBDD tienen una mayor dispersión y buscar aquel hiperplano puede ser complejo y a pesar de encontrarlo no

ser de gran utilidad. Esto es debido a que el hiperplano se busca a partir de los datos de entrenamiento. Una manera de solucionar este problema es con aquel hiperplano que separe los datos y mantenga la estabilidad a pesar de que cometa errores, *penalties*.

Por ello se trata de encontrar un hiperplano con unos *Support Vector* que delimitan el margen máximo de error, *Soft Margin* y permite un número máximo de *penalties* (ver Figura 3.20).

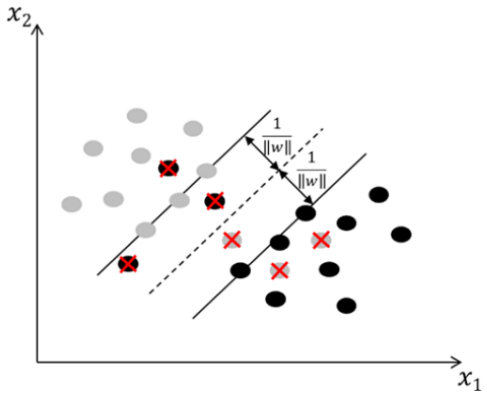


Figura 3.20: Soft Margin en SVM [29].

Existen clasificadores que no pueden ser separados linealmente. Por ello, SVM realiza transformaciones de los datos para aumentar la dimensión de la BBDD y buscar un hiperplano n-dimensional que permita realizar una separación lineal. Este aumento de dimensión tiene como fin encontrar espacios convexos y que así el algoritmo no se quede en mínimos locales. El problema radica en que el aumento de estas dimensiones puede provocar que los tiempos de ejecución incrementan notablemente. Se han planteado diversas soluciones para intentar mejorar y evitar aumentar el espacio, esto se realiza mediante técnicas de **Kernel** [31]. Se aplican técnicas matemáticas que permite saber como esa muestra reacciona ante dimensiones mayores sin tener que aumentarla. Hay diferentes tipos de transformaciones de Kernel:

- **Kernel polinómico**: utilizado para entornos lineales.

$$K(x_1, x_2) = (x_1^T X_2 + b)^d \tag{3.8}$$

- **Kernel radial basis**: el más usado en entornos no lineales.

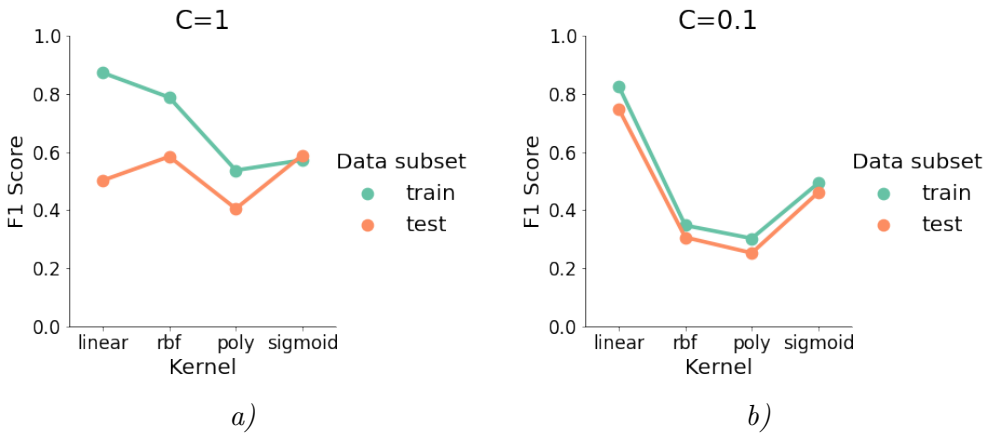
$$K(x_1, x_2) = e^{-(\frac{||x_1 - x_2||^2}{2\sigma^2})} \tag{3.9}$$

- **Kernel sigmoidal**: este es utilizado para distribuir de manera más uniforme los resultados, aunque este Kernel no suele dar buenos resultados.

$$K(x_1, x_2) = tanh(x_1^T x_2 + b) \tag{3.10}$$

Para proceder a la implementación de este algoritmo se especifica el Kernel a utilizar y el número de *penalties* que se permiten dentro del margen máximo de error. También se debe indicar si se desea aplicar la transformación de Kernel y cual. Por último, hay un parámetro *gamma* que controla cómo de cerca tienen que estar las muestras de los vectores de soporte para que sean tenidos en cuenta (Apéndice C.2.5).

Se ha ajustado los parámetros tanto para ResNet como para MobileNet, donde *gamma* es el único parámetro que se mantiene fijo. Con el resto se han entrenado la red para diferentes valores de C y Kernel. En la Figura 3.21 se presentan las mejores combinaciones para ResNet y MobileNet.



a) *b)*

Figura 3.21: SVM óptimo para a) ResNet y b) MobileNet.

Para ambos se ha escogido un Kernel lineal, pero para ResNet hay que utilizar una $C = 1$ y en MobileNet con $C = 0{,}1$ para obtener los mejores resultados en cada uno de los casos.

3.4.6. Multi-Layer Perceptron

Antes de entender en que consiste un **MLP** hay que entender en que consiste las neuronas y el perceptrón [32].

Una **neurona** en IA, está formada por un conjunto de valores de entrada y unos valores que determinan el comportamiento de esta. En la Figura 3.22 se representa una neurona con más de una entrada, llamadas *input*. Cada *input* tiene un enlace con un peso **p**, a los cuales se les aplica el producto con la matriz de pesos **W** de dimensión 1xR, a las cuales se le suma un sesgo, **b** y se les aplica una función, **f**, que aportará un resultado que será la salida de la neurona.

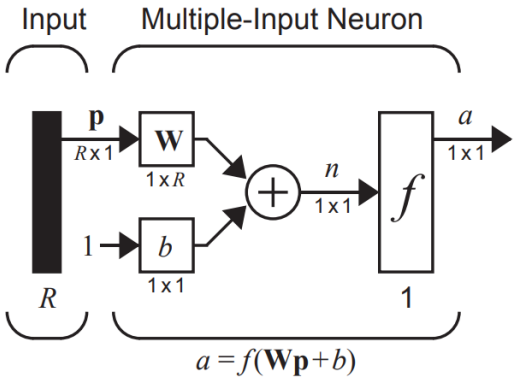


Figura 3.22: Representación de una neurona [32].

Un **perceptrón** es un conjunto de neuronas, las cuales cada una de ella proporciona un valor de salida. La Figura 3.23 representa un perceptrón, la matriz de pesos **W** es de dimensión SxR, donde S es el número de neuronas del perceptrón. También hay que tener en cuenta que ahora hay S valores de salida.

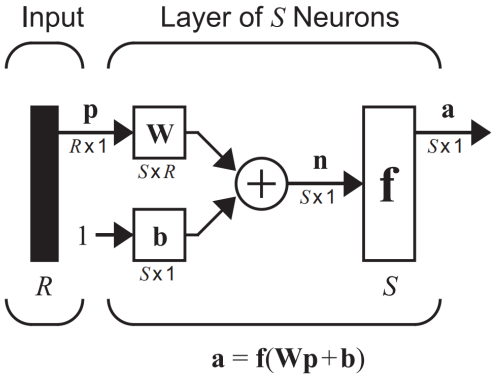


Figura 3.23: Representación de un perceptrón [32]

En un perceptrón hay que realizar la sintonización de los parámetros, **p** y **b**. *Perceptron Learning Rule* es un método que permite la sintonización, iniciando los pesos con cualquier valor. Luego se establece el error, que se calcula como $e = (t - a)$, es la resta entre el valor esperado y el de la predicción. Y para calcular los parámetros de **W** y **b** se utiliza las siguientes fórmulas 3.11 y 3.12, que itera permanentemente.

$$W^{new} = W^{old} + e * p \tag{3.11}$$

$$b^{new} = b^{old} + e * p \tag{3.12}$$

De esta manera se consigue clasificar cualquier problema lineal, pero no los no lineales. **MLP** es el algoritmo que se planteó con el fin de solucionar este problema.

MLP es un conjunto de perceptrones conectados uno detrás de otro, cada uno con un conjunto de neuronas y funciones diferentes, (ver Figura 3.24). Cada perceptrón es una capa, que el primero y el último son considerados la capa de entrada y de salida respectivamente, el resto son las capas ocultas. La salida es una concatenación de cada una de las funciones de cada perceptrón.

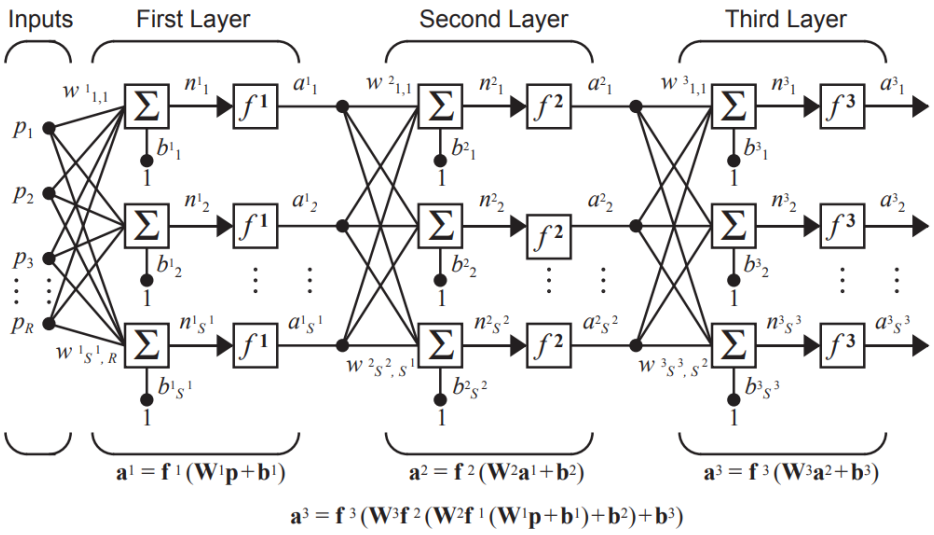


Figura 3.24: Representación de un MLP [32].

En este caso para sintonizar los parámetros no se puede utilizar *Perceptron Learning Rule*, hay que emplear el concepto de *Back Propragation*. Se calcula la sensibilidad del error de las capas de atrás a delante y cuando llegan de nuevo a la capa de entrada, se actualizan los pesos y se propagan por la red. La variación de estos pesos, tratan de buscar el error mínimo de la función y cuando lo encuentra se para de iterar. El problema se encuentra en que se puede atascar en un mínimo local y no seguir iterando. Por ello, existen parámetros que modifican el descenso del gradiente que trata de saltar estos mínimos locales. Por tanto, las variables a modificar del **MLP** son:

- ***hidden layer sizes*** o **capas ocultas**: indica el número de neuronas por cada capa.
- ***batch size***: indica cuantas iteraciones se utilizan para encontrar el valor óptimo del descenso del gradiente.
- ***learning rate init***: es el salto que se realiza en la función para encontrar un mínimo.
- ***learning rate***: puede tomar los siguientes valores:
  - *constant*: mantiene constante el valor de learning rate init.
  - *invscaling*: decrece el valor de learning rate init cada t saltos.
  - *adaptive*: mantiene el valor de learning rate init siempre que consiga mejorar el valor del entrenamiento. En el momento que lleve dos iteraciones sin mejorar, el valor de learning rate ini se divide entre 5.

Es muy difícil encontrar la combinación que proporcione buenos resultados. Por ello, se ha desarrollado un algoritmo, que indica el número máximo de capas y neuronas a probar. Se genera todas las posibles combinaciones. Este conjunto de combinaciones es entrenado para los diferentes valores de atributos mencionados arriba. Se ha utilizado el algoritmo para generar una combinación con un máximo de 5 capas, en la que cada capa podía tomar uno de los siguientes números de neuronas [1,3,5,7...,17,19].

Se ha realizado el entrenamiento para las combinaciones generadas tanto para ResNet como para MobileNet (Apéndice C.2.6). Para cada resultado se indica el número de capas y el número de neuronas por capa. Además, se especifica el valor de ***learning rate init***, ***learning rate*** y ***batch size*** en cada caso. Las combinaciones que presentan un mejor compromiso entre el porcentaje de aciertos en Train y Test para cada modo de entrenamiento de DLC se muestran en la siguiente Tabla 3.2:

DLC	Neuronas	batch size	learning rate init	learning rate
ResNet	[9, 13, 3, 13]	50	0.001	invscaling
MobileNet	[9, 15, 9, 15]	50	0.001	invscaling

Tabla 3.2: MLP óptimo para ResNet y MobileNet.

Para ResNet tenemos 4 capas, la capa de entrada cuenta con 9 neuronas y la capa de salida tiene 13 neuronas, además existen dos capas ocultas, la primera con 9 neuronas y la segunda con 13 neuronas. En MobileNet también tenemos 4 capas, la capa de entrada cuenta con 9 neuronas y la capa de salida tiene 15 neuronas, y las dos capas ocultas, la primera con 9 neuronas y la segunda con 15 neuronas.

Capítulo 4

Resultados

En el Capítulo 3 se ha explicado los algoritmos de entrenamientos de DLC empleados para la predicción de posiciones cinemáticas, ResNet y MobileNet. También se ha contemplado el preprocesado de datos utilizado como paso previo a la explicación de los algoritmos de clasificación empleados en ML.

A continuación, se analiza los diferentes algoritmos de clasificación, tanto para los datos obtenidos por ResNet como de MobileNet. En el Apéndice D, se muestran en detalle las presas que se han clasificado bien y las que no, para ello se ha generado una matriz de confusión para cada algoritmo. Esta matriz de confusión facilita el estudio de las siguientes métricas, que ayudan a evaluar los clasificadores:

- ***Accuarcy*** o **Exactitud** mide los aciertos totales de todas las clases entre el número de muestras.

$$Accuarcy = \frac{\sum(Aciertos)}{Ntotal} \tag{4.1}$$

  Esta métrica presenta bajos resultados ante clases desbalanceadas. El *Accuarcy* se ha identificado como el % total de aciertos en Train o Test.

- ***Precision*** mide el porcentaje de aciertos de cada clase con respecto a la clasificación. Es decir, se calcula el porcentaje de los aciertos de esta clase entre la suma de los aciertos con los que se han identificado como pertenecientes a esta clase, conocidos como Falsos Positivos (FP).

$$Precision_{clase_i} = \frac{Aciertos_{clase_i}}{Aciertos_{clase_i} + FP} \tag{4.2}$$

  Si la precisión es alta, significa que se ha equivocado poco en la clasificación de dicha clase.

- ***Recall*** o **Sensibilidad** mide el ratio de clasificación de una clase respecto a la clase completa. Es decir, se calcula el porcentaje de aciertos de esta clase entre los aciertos más los identificados erróneamente de esta clase, conocidos como Falsos Negativos (FN).

$$Recall_{clase_i} = \frac{Aciertos_{clase_i}}{Aciertos_{clase_i} + FN} \tag{4.3}$$

  Un valor alto de *Recall* significa que se han dejado pocas muestras clasificadas como otra clase.

- ***F1-Score***. Los dos últimos parámetros anteriores son difíciles de ajustar de manera simultánea, por lo que se utiliza la métrica *F1-score* que es una media armónica entre ellas.

$$F1 - Score_{clase_i} = 2 * \frac{Precision_{clase_i} * Recall_{clase_i}}{Precision_{clase_i} + Recall_{clase_i}} \tag{4.4}$$

4.0.1. Logistic Regression

En *Logistic Regression* el único parámetro del algoritmo que se ha tenido que ajustar es *multi_class* para indicar que se trata de una BBDD de más de dos clases, ya que por defecto es un clasificador binario.

A continuación se muestran dos tablas Tabla 4.1 y Tabla 4.2, donde para el 30 % utilizados como Test, indica algunas de las métricas de clasificación para cada una de las presas. Por último, se presenta los valores obtenidos para el porcentaje de acierto de Train para el 70 % de los datos y el porcentaje de acierto de Test para el 30 % de los datos.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.333333	0.800000	0.526316	0.700000
Recall	0.250000	0.727273	0.833333	0.538462
F1-Score	0.285714	0.761905	0.645161	0.608696
Porcentaje de acierto Train = 0.761468				
Porcentaje de acierto Test = 0.583333				

Tabla 4.1: *Logistic Regresion* con ResNet-50.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.437500	0.833333	0.909091	0.555556
Recall	0.583333	0.909091	0.833333	0.384615
F1-Score	0.500000	0.869565	0.869565	0.454545
Porcentaje de acierto Train = 0.871560				
Porcentaje de acierto Test = 0.666667				

Tabla 4.2: *Logistic Regresion* con MobileNet.

Las presas que menor *Precision*, *Recall* y por *F1-Score* son la presa tetradigital y la presa pulpolateral, mientras que las otras posen un mayor porcentaje en las métricas. Por otro lado, MobileNet presenta mejores resultados que ResNet, aunque cabe destacar que su percentaje de acierto en Train tiene un 20 % más que en Test, el sistema podría presentar sobreespecialización.

4.0.2. K Nearest Neighbors

En KNN el parámetro a modificar es el número de vecinos con los cuales se realizan la clasificación. A continuación se muestra las tablas de los resultados observados tanto para ResNet con K = 10 y para MobileNet con K = 20, Tabla 4.3 y Tabla 4.4

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.400000	0.909091	0.666667	0.444444
Recall	0.333333	0.909091	0.500000	0.615385
F1-Score	0.363636	0.909091	0.571429	0.516129

Porcentaje de acierto Train = 0.669725
Porcentaje de acierto Test = 0.583333

Tabla 4.3: KNN con ResNet-50.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.400000	0.909091	1.000000	0.466667
Recall	0.500000	0.909091	0.583333	0.538462
F1-Score	0.444444	0.909091	0.736842	0.500000

Porcentaje de acierto Train = 0.678899
Porcentaje de acierto Test = 0.625000

Tabla 4.4: KNN con MobileNet.

Para **KNN** al igual que en ***Logistic Regresion*** las presas con un menor porcentaje son la presa tetradigital y la presa pulpolateral, aunque en el caso de MobileNet también baja el porcentaje de aciertos de la presa subterminal. En este algoritmo no se puede decir que ResNet o MobileNet sea superior al otro, ya que presentan porcentajes similares.

4.0.3. Árboles de Decisión

En **Árboles de Decisión** el parámetro modificado fue la profundidad del árbol y el criterio de evaluación de los nodos. Para ResNet se especificó la profundidad de 3 y para MobileNet de 4, y en ambos se seleccionó el criterio de Entropía. A continuación se muestran dos tablas que muestran algunas de las métricas de clasificación cada uno de los dos algoritmos (Tabla 4.5 y Tabla 4.6).

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.384615	0.583333	0.555556	0.500000
Recall	0.416667	0.636364	0.416667	0.538462
F1-Score	0.400000	0.608696	0.476190	0.518519

Porcentaje de acierto Train = 0.688073

Porcentaje de acierto Test = 0.500000

Tabla 4.5: Árboles de Decisión con ResNet-50.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.600000	0.888889	0.583333	0.454545
Recall	0.250000	0.727273	0.583333	0.769231
F1-Score	0.352941	0.800000	0.583333	0.571429

Porcentaje de acierto Train = 0.770642

Porcentaje de acierto Test = 0.583333

Tabla 4.6: Árboles de Decisión con MobileNet.

En **Árboles de Decisión** la presa que peor se ha clasificado vuelve a ser la tetradigital. Por último, MobileNet presenta ligeramente mejores resultados de Test que ResNet. Pero hay que tener especial cuidado en ambos clasificadores ya que, el sistema se podría haber sobreespecializado, ya que *Árboles de Decisión* es sensible a los datos de entrenamiento, como se vio en el Capítulo 3, en la sección de *Árboles de Decisión*.

4.0.4. Random Forest

Se calculó el número óptimo de árboles que tiene que haber en el bosque de ResNet y de MobileNet, (3.18), siendo 11 y 14 árboles respectivamente. A continuación se muestran dos tablas que muestra el resumen de las decisiones tomadas en este algoritmo, ***Random Forest***, (Tabla 4.7 y Tabla 4.8).

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.571429	0.666667	0.500000	0.750000
Recall	0.666667	0.727273	0.583333	0.461538
F1-Score	0.615385	0.695652	0.538462	0.571429

Porcentaje de acierto Train = 0.816514
Porcentaje de acierto Test = 0.604167

Tabla 4.7: Random Forest con ResNet-50.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.555556	0.611111	1.0	0.533333
Recall	0.416667	1.000000	0.5	0.615385
F1-Score	0.476190	0.758621	0.666667	0.571429

Porcentaje de acierto Train = 0.844037
Porcentaje de acierto Test = 0.625000

Tabla 4.8: Random Forest con MobileNet.

En MobileNet se observa que a pesar de haber tenido una precisión del 100 % para la presa subterminal, las métricas de *Recall* indica que por mucho que se haya clasificado esta clase como correcta, se está cometiendo errores en otras clases clasificándolas como si perteneciesen a esta. En la comparación con *Árboles de Decisión* se observa que se mejora en los resultados de ResNet, pero MobileNet no. Por último, no se podría decir que ninguno de los dos clasificadores estudiados sea mejor que el otro, ya que tiene bandas de probabilidades muy similares.

4.0.5. Support Vector Machine

SVM es el último algoritmo de ML que se ha estudiado. Tras haber ajustado los hiperparámetros de SVM donde se ha seleccionado un Kernel lineal para ResNet y MobileNet, pero se ha escogido un valor de $C = 1$ y $C = 0{,}1$ respectivamente, y se han conseguido los siguientes resultados, (Tabla 4.9 y Tabla 4.10):

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.277778	0.888889	0.900000	0.454545
Recall	0.416667	0.727273	0.750000	0.384615
F1-Score	0.333333	0.800000	0.818182	0.416667
Porcentaje de acierto Train = 0.779817				
Porcentaje de acierto Test = 0.562500				

Tabla 4.9: SVM con ResNet-50.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.562500	0.916667	1.000000	0.666667
Recall	0.750000	1.000000	0.916667	0.461538
F1-Score	0.642857	0.956522	0.956522	0.545455
Porcentaje de acierto Train = 0.834862				
Porcentaje de acierto Test = 0.770833				

Tabla 4.10: SVM con MobileNet.

Los valores obtenidos para ResNet con el algoritmo **SVM** son bastante buenos para la clasificación de casi todas las presas, pero en ResNet la presa tetradigital presenta un valor bajo en las métricas, mientras que en MobibleNet la presa pulpolateral es la que presenta una métrica menor. Al igual que en *Árboles de Decisión*, MobileNet clasifica bien la presa subterminal, pero hay otras clases que son asignadas como esta clase cuando no lo son. Se puede afirmar que ante estos dos clasificadores, el que presenta un mejor resultado es **MobileNet**, ya que tiene porcentaje mayor de aciertos de Test.

4.0.6. Multi-Layer Perceptron

Para las mejores combinaciones encontradas, se muestran las métricas que califican como es clasificador en la Tabla 4.11 y Tabla 4.12. Donde para ResNet y MobileNet se han empleado cuatro capas ocultas, cada una con sus respectivas neuronas, un de *batch size* de 50, un *learning rate init* de 0.001 y se ha seleccionado *invscaling* en *learning rate*.

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.727273	0.833333	0.785714	0.818182
Recall	0.666667	0.909091	0.916667	0.692308
F1-Score	0.695652	0.869565	0.846154	0.750000

Porcentaje de acierto Train = 0.834862
Porcentaje de acierto Test = 0.791667

Tabla 4.11: MLP con ResNet-50

Presa / %	tetradigital	palmar	subterminal	pulpolateral
Precision	0.785714	0.846154	1.000000	0.909091
Recall	0.916667	1.000000	0.833333	0.769231
F1-Score	0.846154	0.916667	0.909091	0.833333

Porcentaje de acierto Train = 0.926606
Porcentaje de acierto Test = 0.875000

Tabla 4.12: MLP con MobileNet.

Se ha mejorado los resultados de clasificación en comparación con el resto de los algoritmos de clasificación. En ResNet es donde más se observa la mejora del uso de este algoritmo, llegando a tener un valor de porcentaje de aciertos en Test cercano al 80 % y con buenas métricas, como se observa en el Apéndice D. Por otro lado, con el algoritmo de MobileNet contempla mejoras en comparación con *SVM* llegando a obtener un porcentaje de aceirto en Test del 87.5 % superior.

4.0.7. Resumen de resultados

A continuación se resume en la Tabla 4.13 los valores calculados de porcentajes de aciertos de Train y Test para cada algoritmo de clasificación, para ResNet y MobileNet.

DLC	ML	Train	Test
ResNet	Logistic Regresion	0.761468	0.583333
MobileNet	Logistic Regresion	0.871560	0.666667
ResNet	KNN	0.669725	0.583333
MobileNet	KNN	0.678899	0.625000
ResNet	Decision Tree	0.688073	0.500000
MobileNet	Decision Tree	0.770642	0.583333
ResNet	Random Forest	0.816514	0.604167
MobileNet	Random Forest	0.844037	0.625000
ResNet	SVM	0.779817	0.562500
MobileNet	SVM	0.834862	0.770833
ResNet	MLP	0.834862	0.791667
MobileNet	MLP	0.926606	0.875000

Tabla 4.13: Resumen de resultados.

Según los datos obtenidos, el método de procesado de vídeos que ha proporcionado mejores resultados ha sido MobileNet para cualquier algoritmo de clasificación. Como se vio en el Capítulo 3 en la sección de DLC, MobileNet es un algoritmo que hace una buena predicción de las posiciones cinemáticas cuando hay pocos datos y estos son grabados con un móvil. MLP es el algoritmo de clasificación que mejor resultados proporciona. Por tanto, la mejor combinación encontrada es cuando se emplea MobileNet con MLP, llegando a obtener resultados cercanos al 90 %. Seguramente si se aumenta el número de muestras se observaría una mejora en todos los algoritmos.

Capítulo 5

Conclusiones

5.0.1. Conclusiones

Tras haber estudiado la clasificación de las presas y las pinzas, se exponen las siguientes conclusiones que se ciñen a los objetivos planteados:

- El estudio de la clasificación de las presas y pinzas se puede automatizar mediante el diseño de un sistema basado en IA. La clasificación se ha podido realizar con Machine Learning y Deep Learning.
- El diseño de la plataforma de grabación ha aportado diversos beneficios para el desarrollo del Trabajo Fin de Grado. Ha facilitado la grabación de los vídeos aportando una gran estabilidad a estos. El uso de la plataforma facilita el estudio de los vídeos se consigue eliminar el ruido del fondo de la grabación. Además ha permitido realizar otros estudios, debido a que está preparado para realizar la toma de vídeos para cualquier análisis de manos.
- Ante la escasez de la BBDD generada, de aproximadamente 150 muestras, la herramienta DLC ha aportado grandes resultados. Se ha conseguido obtener las posiciones cinemáticas de los vídeos para cualquier instante.
- La posterior caracterización de las posiciones cinemáticas obtenidas ha permitido aumentar la información extraída de los vídeos. El gran aumento de la información tuvo que ser preprocesada eliminando aquellas variables que eran redundantes. Además, para conseguir reducir la dimensión de la BBDD se optó por aplicar técnica de PCA.
- Con el algoritmo de entrenamiento MobileNet utilizado en DLC y el algoritmo de clasificaión MLP que ha proporcionado el mejor resultado con un 90 % presas bien identificadas.
- Se sabe que ante un aumento de la BBDD, DLC habría proporcionado mejores resultados sin presentar tantas alteraciones indeseadas de las posiciones cinemáticas. Este aumento habría ampliado el número de instancias generando clasificadores más óptimos.

5.0.2. Líneas futuras

Este Trabajo Fin de Grado forma parte de un estudio más amplio que se centran en los movimientos de las manos. Por ello se ha comenzado a realizar un sistema capaz de clasificar los diferentes movimientos dando pie a desarrollar sistemas más robustos que profundicen en dicho estudio. En líneas futuras se plantean las siguientes mejoras del estudio:

- Conseguir aumentar el número de vídeos obtenidos e incluso realizar la grabación de estos independientemente del fondo. De esta manera, se desarrollaría un sistema mucho más complejo, que permita detectar con una mayor facilidad la mano sin influir el ruido del vídeo.

- En el actual estudio se ha hecho la clasificación de cuatro presas y pinzas, pero el fin es conseguir llegar a clasificar muchos más tipos. Para ello habría que volver a analizar que clasificador de Machine Learning o Deep Learning es más favorable.

- Otro de los posibles objetivos a largo plazo es realizar es el estudio de clasificación de personas con apraxia, ya que se ha visto que las apraxias ideatorias tienen una gran correlación con el asir de los objetos. Para ello habría que contar con numerosos pacientes que padezcan de la o las apraxias que se deseen clasificar.

Referencias

References

[1] Apraxia. Medlineplus. https://medlineplus.gov/spanish/ency/article/007472.htm. [Accesed online] 10/06/2022.

[2] Wanda Politis, Daniel Gustavo; Rubinstein. Revisión de los patrones de alteración práxicos encontrados en diferentes tipos de demencia, sobre la base de un modelo cognitivo. *Revista española de neuropsicología*, 3(4), 2011.

[3] Juan Luis; Muliz Marrón Elena; Zulaica Cardoso Amaia Álvarez Guerra, Margarita; Arezana Blázquez. Neuropsicilogía de las praxias. http://cv.uoc.edu/annotation/35526708256c132e35ae82da1b199fa2/645604/PID_00241579/PID_00241579.html. [Accesed online] 12/06/2022.

[4] Rehametrics. Praxias. https://rehametrics.com/praxias-rehabilitacion-caracteristicas/. [Accesed online] 10/06/2022.

[5] Wanda Yanina Gómez, Pablo Guillermo; Rubinstein and Daniel Gustavo Politis. Déficit en producción motora y severidad de la demencia en alzheimer y demencia frontotemporal. 6(1):1–12, 2014.

[6] Yahaira; Alva-Diaz Carlos; Montesinos Rosa Custodio, Nilton; Becerra-Becerra. Validación y precisión de la escala de deterioro global (gds) para establecer severidad de demencia en una población de lima. *CES Medicina*, 1(31), 2017.

[7] Daniel Gustav artiGómez, Pablo Guillermo; Politis. Severidad de la demencia y apraxia en demencia frontotemporal variante fronta. *Neurología Argentina*, (2), 2005.

[8] Georg; Cogollor Delgado-Jose Maria Bienkiewicz, Marta; Goldenberg. Use of biological motion based cues and ecological sounds in the neurorehabilitation of apraxia. *Automática, Ingeniería Eléctrica y Electrónica e Informática Industrial*, 2013.

[9] Marta Pérez de Heredia Piédrola, Rosa María Martínez; Torres and Cristina Gómez Calero. *Control neuromotor del agarre y la manipulación, Terapia de la mano*. Síntesis, 2015.

[10] Cristina Gómez Calero Rosa María Martínez Piédrola, Marta Pérez de Heredia Torres. *Terapia de la mano*. Síntesis, 2015.

[11] Yolanda Carretero Serrano. Las diferentes configuraciones de la mano para el agarre y la manipulación. parte i. *RhbNeuromad*, 2020.

[12] *Can machines think? Inteligencia Artificial y Derechos de Daños*. Revista e-Mercatoria, 2011.

[13] What is machine learning? `https://www.ibm.com/cloud/learn/machine-learning`. [Accesed online] 11/06/2022.

[14] Aaron Courville Yoshua Bengio and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35:1798–1828, 2013.

[15] Ai vs. machine learning vs. deep learning vs. neural networks: What's the difference? [Accesed online] 17/06/2022.

[16] Motedis. `https://www.motedis.es/shop/index.php`. [Accesed online] 12/06/2022.

[17] Mathis Laboratory. Deeplabcut. `http://www.mackenziemathislab.org/deeplabcut`. [Accesed online] 11/06/2022.

[18] Alexander; Chen An Chi; Patel Amir; Bethge Matthias Nath, Tanmay; Mathis and Mackenzie Weygandt Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. 14(7):2152–2176, 2019.

[19] Mathis Laboratory. Deeplabcut. `https://deeplabcut.github.io/DeepLabCut/docs/UseOverviewGuide.html`. [Accesed online] 12/06/2022.

[20] Mathis Laboratory. Deeplabcut user guide (for single animal projects). `https://deeplabcut.github.io/DeepLabCut/docs/standardDeepLabCut_UserGuide.html`. [Accesed online] 12/06/2022.

[21] Aurélien Géron. Hands-on machine learning with scikit-learn, keras tensorflow. pages 483–485, 2019.

[22] Technical (hardware considerations). `https://deeplabcut.github.io/DeepLabCut/docs/recipes/TechHardware.html`. [Accesed online] 12/06/2022.

[23] Feature scaling. `https://en.wikipedia.org/w/index.php?title=Feature_scaling&oldid=1084677480`. [Accesed online] 14/06/2022.

[24] Aurélien Géron. Hands-on machine learning with scikit-learn, keras tensorflow. pages 215–231, 2019.

[25] Abhishek Jhunjhunwala. Is logistic regression a good multi-class classifier ? `https://medium.com/@jjw92abhi/is-logistic-regression-a-good-multi-class-classifier-ad20fecf1309#:`

~:text=Multinomial%20logistic%20regression%20is%20a,the%20cross%20entropy%20loss%20function. [Accesed online] 16/06/2022.

[26] K-nearest neighbor(knn) algorithm for machine learning - javatpoint. https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. [Accesed online] 11/06/2022.

[27] Aurélien Géron. Hands-on machine learning with scikit-learn, keras tensorflow. pages 177–190, 2019.

[28] Onesmus Mbaabu. Introduction to random forest in machine learning. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/. [Accesed online] 16/06/2022.

[29] Khanna Rahul Awad, Mariette. *Support Vector Machines for Classification.* 2015.

[30] upport vector machine algorithm. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm. [Accesed online] 16/06/2022.

[31] Aurélien Géron. Hands-on machine learning with scikit-learn, keras tensorflow. pages 155–176, 2019.

[32] Martin T. Hagan; Howard B. Demuth; Mark H. Beale. Neural network design. pages 44–50, 2014.

[33] Mathis Laboratory. How to install deeplabcut. https://deeplabcut.github.io/DeepLabCut/docs/installation.html. [Accesed online] 12/06/2022.

[34] sklearn.linear_model.logisticregression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. [Accesed online] 16/06/2022.

[35] sklearn.neighbors.kneighborsclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html. [Accesed online] 16/06/2022.

[36] Decision trees. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. [Accesed online] 16/06/2022.

[37] sklearn.neural_network.mlpclassifier. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. [Accesed online] 16/06/2022.

Apéndice A

Aspectos éticos, económicos, sociales y ambientales

Para el desarrollo del actual Trabajo Fin de Grado se han considerado diferentes aspectos éticos, económicos, sociales y ambientales. Estos aspectos serán mencionados a lo largo de esta sección.

En el Capítulo 1, se comentó la correlación que existe entre algunas enfermedades de origen neurológico y las apraxias. Por tanto, el estudio del Trabajo Fin de Grado pretende ayudar al estudio con el desarrollo de un sistema de clasificación de las pinzas y las presas. Esto deja una línea futura abierta, para mejorar el sistema y dar apoyo al sistema sanitario.

En este Trabajo Fin de Grado se han aplicado diversas técnicas relacionadas con la Inteligencia Artificial para aplicarlo en el mundo de la Ingeniería Biomecánica. Se ha contado con el apoyo del departamento de Tecnología de Fotónica y Bioingeniería de la Escuela Técnica Superior de Ingeniería de Telecomunicaciones en la Universidad Politécnica de Madrid, además, el estudio se ha respaldado por el Departamento de Fisioterapia, Terapia ocupacional, Rehabilitación Y Medicina Física de la facultad de Ciencias de la Salud de la Universidad Rey Juan Carlos.

A.0.0.1. Impacto social

Se pretende dar apoyo al estudio de la salud de la sociedad. Las personas pueden sufrir accidentes o contraer enfermedades, se puede determinar su gravedad en función del grado de las apraxias. De esta manera se pretende reducir el tiempo del diagnóstico precoz y aumentar la calidad de vida.

A.0.0.2. Impacto medioambiental

Los materiales empleados son la estructura de grabación y un dispositivo móvil para la grabación de los vídeos. El móvil es de uso cotidiano y la mayor parte de la población cuenta con uno, por ello se decidió utilizarlo como herramienta de grabación para evitar la compra de cámaras especializadas en la captación de movimiento. Por otro lado, la estructura ha permitido realizar otros estudios por el departamento de

Tecnología de Fotónica y Bioingeniería, y al contar con el uso de perfiles de aluminio para su montaje facilita su desmontaje para poder buscar otras aplicaciones.

A.0.0.3. Impacto económico

Para el desarrollo del sistema automático solo se necesita hacer uso de la plataforma de grabación, donde los costes vienen indicados en el Apéndice B. El uso del sistema de captura de vídeos permite la grabación de los vídeos de manera rápida y su posterior clasificación permite la posterior inspección visual en un menor tiempo. Por ello, a pesar de realizar una inversión mínima en la plataforma, su uso reduce el tiempo invertido y por tanto, personal y dinero. Además, en líneas futuras se plantea mejorar el sistema y tratar de evitar hacer uso de la maqueta, por tanto, solo habría que hacer uso de cualquier móvil para grabar los movimientos deseados.

A.0.0.4. Responsabilidad ética y profesional

Los vídeos fueron grabados en la Escuela Técnica Superior, en los que no ha hecho tener constancia de las características de los sujetos, por tanto, se ha conseguido respetar la privacidad de estos. Se ha mantenido un trato cordial con todos los usuarios para incentivar a la participación de la toma de vídeos.

Apéndice B

Presupuesto económico

A continuación se muestran los gatos del actual Trabajo Fin de Grado, donde se reúnen los gastos de los recursos empleados para su desarrollo. Hay que tener en cuenta los gastos de las personas implicadas y el material empleado.

- **Costes de personal**. Considerando que necesita un Ingeniero de Telecomunicaciones para conseguir la toma de muestras, el estudio de datos y su posterior clasificación se han estimado los siguientes costes (Tabla B.1):

	Costes/hora	Horas	Total
Ingniero	12	400	4800

Tabla B.1: Costes del personal.

- **Costes del material empleado**. Se han calculado los precios del material empleado, ya que se ha empleado el sistema de grabación mencionado, un móvil y un portátil para poder realizar el estudio (B.2).

Material	Unidades	Precio	Total
Tabla de madera (50x50cm)	1	15€	15€
Perfiles de aluminio (50cm)	9	7.23€/m	32.535€
Escuadra	14	0.85€	11.9€
Tornillos	28	0.45€	12.6€
Tuercas	28	0.45€	4.2€
Móvil	1	180€	180€
Portátil	1	800€	800€
GeForce RTX 2080 Ti	1	1000€	1000€
		Total	**2056.24€**

Tabla B.2: Costes del material.

B.0.1. Resumen de Costes

Aquí se resumen los costes totales para la realización de este Proyecto (Tabla B.3):

Coste de personal	4800€
Coste de Material	2056.24€
Total	**6.856.24**

Tabla B.3: Resumen de costes.

Apéndice C

Implementación en Python

C.1. Implemntación de DeepLabCut en Python

En esta sección se explica como se realiza la instalación de DLC y como realizar la elaboración del proyecto completo en Python.

C.1.1. Guía de instalación de DeepLabCut

DLC es una herramienta que funciona tanto en sistemas que cuentan tanto, con CPU como GPU. Preferiblemente se emplea la GPU ya que en algoritmos de entrenamiento y evaluación como ResNet el sistema es capaz de ir hasta 10 veces más rápido. Esta herramienta se puede utilizar en diferentes plataformas, puede ser ejecutada en remoto, en un servidor o plataformas online como GoogleColab.

La instalación de DLC se realiza desde la máquina de comandos ejecutando la siguiente linea [33]:

```
[1] pip install deeplabcut
```

En el caso de querer instalarlo en una GPU:

```
[2] pip install deeplabcut–gpu
```

Se utilizan librerías como tensorflow, por ello, hay que comprobar la compatibilidad de versiones. Una vez realizada la instalación en CPU o GPU, se decide si emplear la interfaz gráfica de usuario (GUI, *Graphical User Interface*) o utilizar DLC escribiendo en código.

En el caso de querer emplear la GUI habrá que aplicar la siguiente línea de comando:

```
[3] python deeplabcut –m
```

Posteriormente se abrirá la siguiente interfaz gráfica de inicio (ver Figura C.1).

En este Trabajo se ha utilizado el servidor que se encuentra alojado en el departamento del Robolabo en la Escuela Técnica Superior de Ingeniería de Telecomunicaciones en la Universidad Politécnica de Madrid. Se accedió mediante ssh al servidor. DLC utiliza interfaz gráfica, entonces para poder utilizarla desde el servidor hay que tener habilitado el X11.

Figura C.1: Interfaz de inicio de DeepLabCut.

C.1.2. Creación del proyecto

Antes de crear el proyecto en Python, hay que importar la librería de deeplabcut.

```
import deeplabcut
```

Para crear un proyecto se hace de la siguiente manera:

```
path_config_file = deeplabcut.create_new_project(Name, Autor,
[videos], videotype= '.mp4',copy_videos=True)
```

- **Name**: es el nombre del proyecto.
- **Autor**: autor del proyecto.
- **videos**: puede se un vídeo o una lista de vídeos. En el caso de ser una lista de vídeos se puede pasar una lista de rutas relativas o absolutas de los vídeos, pero también se puede especificar la ruta a una carpeta en este caso sería necesario especificar el tipo de vídeo.
- **videotype**: permite verificar el formato de los vídeos utilizados para crear el proyecto.
- **copy_videos**: sirve para copiar o no los vídeos importados en la carpeta vídeo.

C.1.3. Etiquetado de Frames

La extracción de Frames significa dividir el vídeo en fotos en determinados instantes. Para extraer los frames:

```
deeplabcut.extract_frames(path_config_file, mode='automatic',
algo = 'kmeans', userfeedback = False )
```

- **mode**: automatic o manual. Se escoge automatico para no estar especificando que frames se desean.
- **algo**: kmeans o uniform. La diferencia se encuentra en que **uniform** escoge los frames distribuidos en el tiempo, mientras que **kmeans** pretende que estos frames sean lo más diferente entre ellos.
- **userfeedback**: False o True, para identificar si se quiere verificar cada frame extraído.

Posteriormente se procede al etiquetado de los frames, las etiquetas son los puntos especificados en el fichero de configuración *config.yaml* en el parámetro ***bodyparts***.

```
deeplabcut.label_frames(path_config_file)
```

C.1.4. Entrenamiento

Hay que crear una BBDD antes de comenzar con el entrenamiento de los datos.

```
deeplabcut.create_training_dataset(path_config_file,
net_type='resnet_50'/'mobilenet_v2_1.0',
windows2linux=True)
```

Aquí cabe destacar el atributo **windows2linux**, que se debe poner a **True** si se cambia de Sistema Operativo, para que no haya confusiones y problemas con las rutas que se encuentran dentro de los ficheros.

Los parámetros de entrenamiento se pueden modificar de manera manual en el fichero de ***pose_config.yaml*** o cuando se introduzca la línea de ejecución de entrenamiento del sistema.

```
deeplabcut.train_network(path_config_file,
gputouse=1, max_snapshots_to_keep=10,
displayiters=1000, saveiters=20000, maxiters=500000)
```

- **gputouse**: este parámetro indica que en el caso de contar con una o más GPUs, indicar cual quieres.
- **max_snapshots_to_keep**: en cada iteración se decide cuantas muestras de la iteración anterior deseas mantener.
- **displayiters**: permite ver al usuario la ejecución de las iteraciones que son mostradas por la terminal de ejecución.
- **saveiters**: determina cada cuanto se guardan los pesos de las iteraciones en la carpeta de **train** que se encuentra en **dlcmodels** y la iteración correspondiente.
- **maxiters**: indica el número máximo de iteraciones a ejecutar.

En el caso de haber terminado el entrenamiento y querer entrenar un mayor número de iteraciones, bastaría con modificar dentro del fichero ***pose_config.yaml*** se puede modificar la variable **init_weights** y poner la ruta absoluta de la última iteración realizada. Los datos de entrenamiento se encuentran alojados dentro de la carpeta de **training-dataset**.

C.1.5. Análisis de vídeo

El método para realizar el análisis de vídeo es:

```
deeplabcut.analyze_videos(path_config_file, [videos],
videotype = '.mp4', gputouse = 1, save_as_csv = True)
```

Donde el parámetro de **videos** al igual que al crear proyecto, se puede pasar una lista de vídeos o la ruta a una carpeta.

Y **save_as_csv** genera un fichero '.csv' por cada vídeo analizado, devolviendo las posiciones cinemáticas de cada marcador en cada instante. En el caso de seleccionar una carpeta para que sea analizada, los ficheros '.csv' son generados en esa carpeta. Por otro lado, se pueden generar vídeos nuevos que introducen los marcadores de los vídeos y el skeleton. Esto facilita la inspección visual de las predicciones creadas por el sistema.

C.2. Implementación de los algorítmos de clasificación en Python

Antes de realizar la clasificación de los datos hay que realizar la división de los datos entre Train y Test:

```
X1, y1 = df.iloc[:, :-1].values, df.tipo_pinza.values
X1_train, X1_test, y1_train, y1_test =
    train_test_split(X1, y1, test_size=0.3, random_state=11)
```

X1_train son los datos de entrenamientos e ***y1_train*** es la clase correspondiente. Mientras que ***X1_test*** y ***y1_test*** son los datos que se utilizarán para realizar el Test.

C.2.1. Logistic Regression

La creación y entrenamiento de este algoritmo se realiza de la siguiente manera [34]:

```
from sklearn.linear_model import LogisticRegression
...
logreg1 = LogisticRegression(multi_class='multinomial')
logreg1.fit(X_train, y_train)
```

El parámetro *multi_class* por defecto está a null, esto significa que se realizará una clasificación binaria. En caso de querer realizar una clasificación con más de una clase, como la planteada hay que modificarlo tal y como viene indicado

C.2.2. K Nearest Neighbors

Para implementar KNN este algoritmo en Python y entrenar el clasificador [35]:

```
from sklearn.neighbors KNeighborsClassifier
...
knn_clf = KNeighborsClassifier(n_neighbors=K)
knn_clf.fit(X1_train, y1_train)
```

El único parámetro a ajustar en este algoritmo es ***n_neighbors*** que determina el número de vecinos deseados para realizar la clasificación.

C.2.3. Árboles de Decisión

Decision Tree se implementa indicando la profundidad del árbol, *depth* y el criterio que se desea seguir, *criterion*. Por tanto, para crear el árbol y realizar su posterior entrenamiento [36]:

```
from sklearn.tree import DecisionTreeClassifier
...
tree_clf = DecisionTreeClassifier(max_depth=depth,
        criterion = 'gini'/'entropy')
tree_clf.fit(X1_train, y1_train)
y1_pred = tree_clf.predict(X1_train)
```

C.2.4. Random Forest

Con algunas excepciones *Random Forest* utiliza los mismos parámetros que *Decision Tree*. La integración del modelo:

```
from sklearn.ensemble import RandomForestClassifier
...
random_forest = RandomForestClassifier(n_estimators=M,
    max_depth=p, max_leaf_nodes,
    criterion = 'gini'/'entropy').fit(X1_train, y1_train)
```

Donde *n_estimators* indica el número máximo de árboles en el bosque, *max_depth* indica la máxima profundidad del árbol y *max_leaf_nodes* es el número máximo de hojas. Donde es necesario indicar el número máximo de árboles en el bosque, la máxima profundidad del árbol y el número máximo de hojas.

C.2.5. Support Vector Machine

Para crear este modelo de clasificación hay que especificar los parámetros necesarios que permitan encontrar el mejor hiperplano que separe los datos. Su implementación es:

```
from sklearn.svm import SVC
...
svc = SVC(C=c, kernel='linear'/'rbf'
/'poly'/'sigmoid', gamma = g)
svc.fit(X1_train, y1_train)
```

Donde C es un parámetro que se regula para modificar los *penalties*, Kernel es utilizado para identificar la técnica a utilizar en este algoritmo. Por último, el parámetro gamma que controla cómo de cerca deben estar las muestras de los vectores de soporte para que sean tenidos en cuenta. A gamma se le suele dar los siguientes valores:

- **'scale'** que indica que el valor de γ es: $\gamma = \frac{1}{n_{features} * var(X)}$, valor por defecto
- **'auto'** que indica que el valor de γ es: $\gamma = \frac{1}{n_{features}}$

C.2.6. Multi-Layer Perceptron

Para la implementación de **MLP** hay que ajustar varios parámetros, a continuación se explica los parámetros más relevantes de este algoritmo [37]:

```
from sklearn.neural_network import MLPClassifier
...
perceptron = MLPClassifier(hidden_layer_sizes=[s1,s2,s3],
    batch_size=100, learning_rate_init=0.1,
    learning_rate='constant',
    shuffle=True, random_state=51).fit(X_train, y_train)
```

- ***hidden_layer_sizes***: indica el número de neuronas por cada capa. Si hidden_layer_sizes = [20,10,5], significa que la capa de entrada tiene 20 neuronas, que hay una capa oculta y que es de 10 neuronas y la capa de salida tiene 5.
- ***batch_size***: indica cuantas iteraciones se utilizan para encontrar el valor óptimo del descenso del gradiente.
- learning_rate_init: es el salto que se realiza en la función para encontrar un mínimo.
- ***learning_rate***: puede tomar los siguientes valores:
  - *constant*: mantiene constante el valor de learning_rate_init.
  - *invscaling*: decrece el valor de learning_rate_init cada t saltos.
  - *adaptive*: mantiene el valor de learning_rate_init siempre que consiga mejorar el valor del entrenamiento. En el momento que lleve dos iteraciones sin mejorar, el valor de learning_rate_ini se divide entre 5.

Apéndice D

Extensión de Resultados

Se desarrolla más en detalle el resultado de cada algoritmo. Se muestran tablas tanto para ResNet como para MobileNet para cada algoritmo de clasificación empleado.

Las tablas son matrices de confusión que muestran las predicciones realizadas para cada una de las presas. Además se indica las métricas de *Precision*, *Recall* y *F1-Score*.

D.0.1. Logistic Regression

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	3	0	1	5	9
palmar	1	8	1	0	10
subterminal	5	3	10	1	19
pulpolateral	3	0	0	7	10
Total	12	11	12	13	48
Precision	0.333333	0.800000	0.526316	0.700000	
Recall	0.250000	0.727273	0.833333	0.538462	
F1-Score	0.285714	0.761905	0.645161	0.608696	

Porcentaje de acierto Train = 0.761468

Porcentaje de acierto Test = 0.583333

Tabla D.1: *Logistic Regresion* con ResNet-50.

Para comprender un poco mejor la tabla se utiliza como ejemplo la Tabla D.1. Para el caso de presa tetradigital que en Test hay un total de 12 muestras, se han clasificado 3 como presa tetradigital y se ha equivocado clasificando 1 como presa palmar, 5 como presa subterminal y 3 como presa pulpolateral, y en Recall se indica la probabilidad de que sabiendo que tiene que ser presa tetradigital hayas acertado.

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	7	1	1	7	16
palmar	0	10	1	1	12
subterminal	1	0	10	0	11
pulpolateral	4	0	0	5	9
Total	12	11	12	13	48
Precision	0.437500	0.833333	0.909091	0.555556	
Recall	0.583333	0.909091	0.833333	0.384615	
F1-Score	0.500000	0.869565	0.869565	0.454545	

Porcentaje de acierto Train = 0.871560

Porcentaje de acierto Test = 0.666667

Tabla D.2: *Logistic Regresion* con MobileNet.

D.0.2. K Nearest Neighbors

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	4	1	1	4	10
palmar	0	10	1	0	11
subterminal	2	0	6	1	9
pulpolateral	6	0	4	8	18
Total	12	11	12	13	48
Precision	0.400000	0.909091	0.666667	0.444444	
Recall	0.333333	0.909091	0.500000	0.615385	
F1-Score	0.363636	0.909091	0.571429	0.516129	

Porcentaje de acierto Train = 0.669725

Porcentaje de acierto Test = 0.583333

a) ResNet

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	6	1	2	6	15
palmar	0	10	1	0	11
subterminal	0	0	7	0	7
pulpolateral	6	0	2	7	15
Total	12	11	12	13	48
Precision	0.400000	0.909091	1.000000	0.466667	
Recall	0.500000	0.909091	0.583333	0.538462	
F1-Score	0.444444	0.909091	0.736842	0.500000	

Porcentaje de acierto Train = 0.678899

Porcentaje de acierto Test = 0.625000

b) MobileNet

Tabla D.3: KNN extension de resultados a) ResNet b) MobileNet.

D.0.3. Árboles de Decisión

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	5	1	2	5	13
palmar	0	7	5	0	12
subterminal	1	2	5	1	9
pulpolateral	6	1	0	7	14
Total	12	11	12	13	48
Precision	0.384615	0.583333	0.555556	0.500000	
Recall	0.416667	0.636364	0.416667	0.538462	
Recall	0.400000	0.608696	0.476190	0.518519	

Porcentaje de acierto Train = 0.688073

Porcentaje de acierto Test = 0.500000

a) ResNet

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	3	0	2	0	5
palmar	0	8	0	1	9
subterminal	2	1	7	2	12
puplpolateral	7	2	3	10	22
Total	12	11	12	13	48
Precision	0.600000	0.888889	0.583333	0.454545	
Recall	0.250000	0.727273	0.583333	0.769231	
F1-Score	0.352941	0.800000	0.583333	0.571429	

Porcentaje de acierto Train = 0.770642

Porcentaje de acierto Test = 0.583333

b) MobileNet

Tabla D.4: Decision Tree extension de resultados a) ResNet b) MobileNet.

D.0.4. Random Forest

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	8	2	0	4	14
palmar	0	8	4	0	12
subterminal	3	1	7	3	14
pulpolateral	1	0	1	6	8
Total	12	11	12	13	48
Precision	0.571429	0.666667	0.500000	0.750000	
Recall	0.666667	0.727273	0.583333	0.461538	
F1-Score	0.615385	0.695652	0.538462	0.571429	

Porcentaje de acierto Train = 0.816514
Porcentaje de acierto Test = 0.604167

a) ResNet.

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	5	0	1	3	9
palmar	2	11	3	2	18
subterminal	0	0	6	0	6
pulpolateral	5	0	2	8	15
Total	12	11	12	13	48
Precision	0.555556	0.611111	1.0	0.533333	
Recall	0.416667	1.000000	0.5	0.615385	
F1-Score	0.476190	0.758621	0.666667	0.571429	

Porcentaje de acierto Train = 0.844037
Porcentaje de acierto Test = 0.625000

b) MobileNet.

Tabla D.5: Random Forest extension de resultados a) ResNet b) MobileNet.

D.0.5. Support Vector Machine

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	5	3	2	8	18
palmar	0	8	1	0	9
subterminal	1	0	9	0	10
pulpolateral	6	0	0	5	11
Total	12	11	12	13	48
Precision	0.277778	0.888889	0.900000	0.454545	
Recall	0.416667	0.727273	0.750000	0.384615	
F1-Score	0.333333	0.800000	0.818182	0.416667	

Porcentaje de acierto Train = 0.779817

Porcentaje de acierto Test = 0.562500

a) ResNet.

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	9	0	1	6	16
palmar	0	11	0	1	12
subterminal	0	0	11	0	11
pulpolateral	3	0	0	6	9
Total	12	11	12	13	48
Precision	0.562500	0.916667	1.000000	0.666667	
Recall	0.750000	1.000000	0.916667	0.461538	
F1-Score	0.642857	0.956522	0.956522	0.545455	

Porcentaje de acierto Train = 0.834862

Porcentaje de acierto Train = 0.834862

Porcentaje de acierto Test = 0.770833

b) MobileNet.

Tabla D.6: SVM extension de resultados a) ResNet b) MobileNet.

D.0.6. Multi-Layer Perceptron

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	8	0	0	3	11
palmar	0	10	1	1	12
subterminal	2	1	11	0	14
pulpolateral	2	0	0	9	11
Total	12	11	12	13	48
Precision	0.727273	0.833333	0.785714	0.818182	
Recall	0.666667	0.909091	0.916667	0.692308	
F1-Score	0.695652	0.869565	0.846154	0.750000	

Porcentaje de acierto Train = 0.834862

Porcentaje de acierto Test = 0.791667

a) ResNet.

Presa / Predicción	tetradigital	palmar	subterminal	pulpolateral	Totales
tetradigital	11	0	1	2	14
palmar	0	11	1	1	13
subterminal	0	0	10	0	10
pulpolateral	1	0	0	10	11
Total	12	11	12	13	48
Precision	0.785714	0.846154	1.000000	0.909091	
Recall	0.916667	1.000000	0.833333	0.769231	
F1-Score	0.846154	0.916667	0.909091	0.833333	

Porcentaje de acierto Train = 0.875000

Porcentaje de acierto Test = 0.875000

b) MobileNet.

Tabla D.7: MLP extension de resultados a) ResNet b) MobileNet.