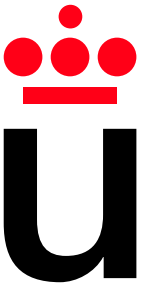


Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2023-2024

Trabajo Fin de Grado

MÉTODOS DE APRENDIZAJE AUTOMÁTICO PARA LA CLASIFICACIÓN DE COMPORTAMIENTOS ESTEREOTIPADOS

Autor: Gonzalo Ortega Carpintero
Tutor académico: Alberto Fernández Gil
Tutora CSIC: Paula Peixoto Moledo

Agradecimientos

(Por redactar)

Resumen

(Por redactar)

Palabras clave:

- Python
- Neurociencia
- Aprendizaje automático
- ...

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de códigos

1 Introducción

Este documento recoge el Trabajo de Fin de Grado (TFG) de Gonzalo Ortega Carpintero, incluido en el itinerario del grado en Ingeniería del *Software* de la Universidad Rey Juan Carlos.

En este primer capítulo se presenta la motivación que ha dado lugar a la idea y desarrollo de este TFG en la Sección 1.1. En la Sección 1.2 se introduce el objetivo principal del trabajo y se enumeran los objetivos parciales que se han propuesto para conseguirlo. En la Sección 1.3 se hace un repaso a la metodología seguida para realizar el trabajo y finalmente, en la Sección 1.4 se introducen las herramientas y bibliotecas que se han utilizado.

1.1. Motivación

La Neurociencia es la disciplina científica que estudia el sistema nervioso y todas sus componentes. Para el estudio del funcionamiento del cerebro pueden utilizarse multitud de técnicas diferentes, pero para evitar la intrusión en el sujeto de estudio de muchas de ellas, muchos experimentos e investigaciones se basan en el estudio del comportamiento. Es la etología la encargada de este tipo de estudio en animales y en la cual, históricamente, la categorización de comportamientos dependía fuertemente de la observación manual. Eso ha conllevado siempre una excesiva cantidad de tiempo y las desventajas de los posibles errores de percepción humanos, limitando la cantidad y la calidad de los análisis a realizar.

Gracias a los recientes avances en algoritmos de aprendizaje automático, hoy en día es posible computar multitud de datos de forma simultánea, agilizando y automatizando análisis como el ilustrado previamente.

Este Trabajo de Fin de Grado se ha realizado a lo largo de una estancia de prácticas académicas en el Jercog's Team, un equipo de investigación perteneciente al Instituto Cajal, instituto de Neurociencia del Consejo Superior de Investigaciones Científicas. El equipo está centrado en el estudio de la memoria mediante experimentos con ratones, y uno de sus proyectos abiertos consistía en poder elaborar una herramienta para clasificar automáticamente los comportamientos estereotipados de los ratones. Estos son comportamientos cortos, repetitivos y con cierta tendencia a generar patrones, tales como rascarse, caminar en círculos o levantarse a dos patas, ejecutados sin ninguna finalidad, y usualmente inducidos por estar en entornos cerrados y artificiales.

Para la ejecución de uno de sus últimos experimentos utilizan ratones tratados con anticuerpos bloqueadores de NMDA (*N-Metil-D-aspartato*), restringiendo la capacidad de los ratones para almacenar memoria a largo plazo. En las sesiones de experimentación los ratones tratados ejercen las mismas tareas que ratones control, tratados con un suero placebo, y en las grabaciones hay ratones de ambos tipos. A ojo de un experimentador, es complejo determinar a ciegas si un ratón está bajo los efectos de NMDA o es control, por lo que es interesante la posibililidad de contar con una herramienta que sea capaz de distinguir unos de otros a partir de los videos de las sesiones.

Durante la estancia se han valorado multitud de técnicas de análisis y procesado de datos, haciendo hincapié en métodos de aprendizaje automático para tratar de completar el proyecto gracias a los últimos avances en computación.

1.2. Objetivos

El objetivo de este trabajo es distinguir de forma automática, mediante la clasificación de los comportamientos o de forma directa, si un animal está bajo los efectos de un bloqueador de MDMA.

1.2.1. Hipótesis

Para contrastar nuestro objetivo hemos planteado la siguiente hipótesis:

Dados los datos de posición de una serie de puntos de un ratón en su caja a lo largo de una sesión de video, mediante técnicas de aprendizaje automático, es posible diferenciar diferentes comportamientos del ratón, e identificar si está bajo los efectos de un bloqueador de NMDA.

1.2.2. Objetivos parciales

Con el fin de desglosar el objetivo principal, y de poder corroborar o refutar nuestra hipótesis, hemos planteado varios objetivos parciales.

- Realizar una introducción a técnicas de aprendizaje automático aplicables al procesado de datos en el análisis de experimentos con animales.
- Preparar y preprocesar datos provenientes de archivos de video para su posterior análisis.
- Aplicar algoritmos de agrupación para separar fragmentos de video y clasificarlos en función del comportamiento que esté realizando el animal.
- Aplicar algoritmos no supervisados para distinguir animales control de animales tratados.
- Aplicar algoritmos supervisados para distinguir animales control de animales tratados.

1.3. Metodología

El trabajo se divide en dos capítulos principales, en el Capítulo 2 abordaremos el estudio teórico de ciertos algoritmos de agrupamiento, reducción de dimensionalidad y de clasificación supervisada. Todo ello desde un punto de vista teórico con ejemplos fáciles de visualizar para ayudar al lector a comprender en profundidad los métodos.

En el Capítulo 3, desarrollaremos paso a paso el análisis real ejercido durante la estancia en el Instituto Cajal. El principal objetivo durante estos meses ha sido buscar la mejor forma de, dados una serie de videos de ratones en su caja, detectar automáticamente diferentes tipos de comportamientos. El fin último del proyecto es estudiar la posibilidad de clasificar dos grupos de animales, control y medicados, basándose en los comportamientos estereotipados.

1.4. Bibliotecas y herramientas utilizadas

DeepLabCut

DeepLabCut [1] es una herramienta para la estimación 2D y 3D sin marcadores mediante el uso de redes neuronales. Es capaz de identificar de rastrear diferentes

partes del cuerpo de múltiples especies realizando todo tipo de comportamientos. Esta ha sido la base de todo nuestro trabajo, ya que todos los videos que hemos analizado han sido procesados en primer lugar por DeepLabCut para rastrear las posiciones de múltiples puntos de los animales a lo largo de los videos de las sesiones. Debido a su importancia, damos una explicación más en detalle de su funcionamiento y de como lo hemos utilizado en la sección 3.1.1.

Google Colab

Google Colab es una herramienta para realizar cuadernos de Jupyter en línea, y poder ejecutarlos en el *backend* de Google. Estos son documentos que intercalan fragmentos de texto con fragmentos de código ejecutable en Python, así como la salida de las distintas ejecuciones. Todo el codigo realizado para este trabajo ha sido realizado en cuadernos de Colab, y puede ser consultado en los siguientes enlaces:

Ejemplos teóricos	https://colab.research.google.com/drive/1qL-LQTCFLZcqExN7qyYyRAT1nm7P8NhuK?usp=sharing
DeepCutLab	https://colab.research.google.com/drive/1dFb-mUcfW9el50v0RBCkLIItTD6SF7A3x?usp=sharing
Análisis principal	https://colab.research.google.com/drive/1ak2VpDi-zTnV-uEDp-viEkpa8GFDGuBR?usp=sharing

Tabla 1.1: Enlaces a cuadernos de Google Colab.

Scikit-learn

Scikit-learn [2] es una biblioteca de código abierto de Python que contiene numerosas implementaciones de algoritmos de aprendizaje automático. Hemos usado dichas implementaciones tanto como para la explicación teórica de los algoritmos, como para el análisis real de los datos de los animales.

PyTorch

[3]

pandas

`pandas` es una biblioteca de manejo de datos mediante tablas denominadas `DataFrame`. Todos los datos que hemos cargado para ser analizados los hemos

guardado como DataFrames para poder tener un fácil acceso a todos ellos y a sus variables pudiendo, además, visualizarlos de una forma sencilla.

NumPy

NumPy es una de las bibliotecas principales de cálculo científico en Python. Proporciona un objeto de `array` multidimensional y numerosas funciones estadísticas y algebraicas de mucha utilidad.

matplotlib

`matplotlib` es una biblioteca para crear figuras en Python. Todas las figuras de representación de datos que aparecen en este trabajo han sido creadas con `matplotlib`. El código concreto utilizado para generarlas puede consultarse en los cuadernos de Colab.

2

Fundamentos teóricos

En este Capítulo vamos a dar una introducción varios algoritmos de aprendizaje automático.

2.1. Métodos no supervisados

Los algoritmos de aprendizaje automático no supervisados se utilizan cuando no se conoce la salida esperada. Al algoritmo de aprendizaje se le otorgan los datos de entrada y se le pide extraer información de estos datos. Las principales aplicaciones de estos algoritmos, las cuales vamos a aprovechar, son la agrupación de datos y la reducción de dimensionalidad de las variables de los mismos. Esa última es usada principalmente para poder hacer representaciones de datos multidimensionales, los cuales serian complejos de visualizar de otra forma.

La principal pega que pueden tener estos algoritmos es que, si bien no siempre son capaces de identificar conocimiento dados los datos utilizados, cuando lo obtienen, no siempre es el conocimiento que esperábamos obtener. Póngase el ejemplo de un algoritmo que tratase de agrupar rostros de personas iguales. Al no darle a priori ningún tipo de salida de ejemplo, el algoritmo puede acabar clasificando si los rostros están de frente o de lado, no precisamente lo que esperábamos. Es por ello que estos algoritmos cuentan con diversidad de parámetros para ajustarlos a nuestras necesidades, tratando de realizar la agrupación deseada.

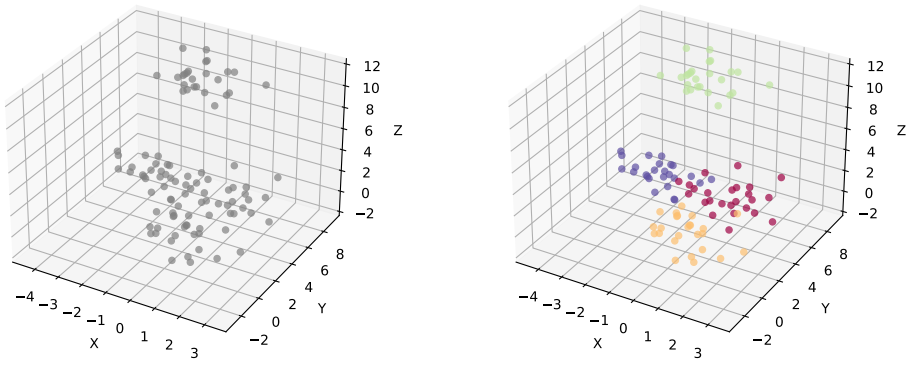
En esta sección vamos a estudiar a fondo tres tipos de algoritmos de agru-

pación: el agrupamiento por k-medias, la agrupación aglomerada, y la agrupación por afinidad. Además, estudiaremos también el principal algoritmo de reducción de dimensiones, el análisis de componentes principales, PCA, de sus siglas en inglés. Los principales ejemplos y explicaciones de los algoritmos han sido inspirados por los dados en [4].

Código 2.1: Generar datos artificiales de prueba.

```
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=100, centers=4,
                  n_features=3, random_state=0)
```



(a) (b)

Figura 2.1: Datos artificiales generados para probar algoritmos de agrupamiento. Consisten en 100 puntos tridimensionales agrupados equilibradamente al rededor de 4 centros aleatorios. 2.1a En gris los puntos generados sin asignar ninguna etiqueta. Estos serán los datos que recibirán los algoritmos para procesar y agrupar. 2.1b Datos etiquetados según el grupo al que pertenecen. Esto nos ayudará a verificar la eficacia de nuestros algoritmos.

En la tabla 1.1 está el enlace para poder ejecutar el código en el *backend* de Google.

2.1.1. Análisis de componentes principales

(EXPLICACIÓN PCA)

Código 2.2: Escalar y estandarizar los datos.

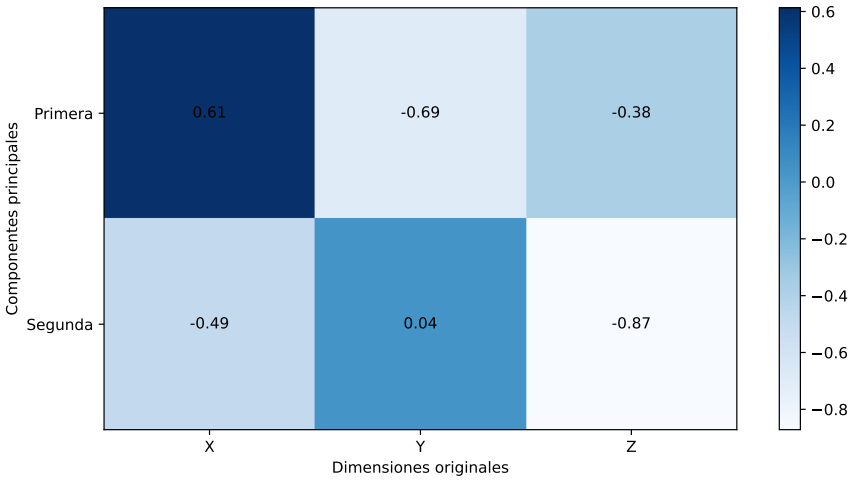
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X)
features_scaled = scaler.transform(X)
```

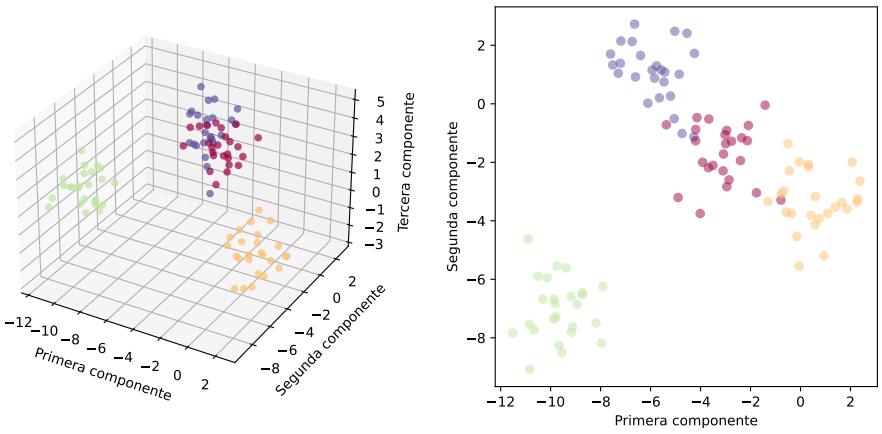
Código 2.3: Realizar un PCA.

```
from sklearn.preprocessing import StandardScaler

pca = PCA(n_components=3)
pca.fit(features_scaled)
X_reduced = pca.transform(X)
```



(a)



(b) (c)

Figura 2.2: Prueba del algoritmo de PCA. 2.2a Mapa de calor con las ponderaciones asignadas por el PCA a cada uno de las dimensiones originales. 2.2b Visualización de los datos previamente generados tras escalarlos y aplicarles la PCA. Los colores de los grupos se mantienen, pero los ejes ahora representan las componentes principales, en vez de las dimensiones originales. 2.2c Visualización de las dos primeras componentes principales sobre el plano. Este será el formato en el que visualizaremos más adelante las agrupaciones ejercidas sobre datos reales de dimensionalidad alta.

2.1.2. Agrupamiento por K-medias

El algoritmo de k-medias clasifica los datos separando los datos en n grupos de con la misma varianza.

El algoritmo comienza inicializando aleatoriamente n centroides, siendo n el número de agrupaciones que se le han dicho que realice. Estos centroides serán los centros de las agrupaciones que va a realizar, no tienen por qué pertenecer a los datos, pero sí que están contenidos en su misma dimensión. En cada iteración, el algoritmo asigna a cada punto de los datos el centroide más próximo y luego asigna a cada agrupación un nuevo centroide calculado como la media de los datos que se han asignado a dicha agrupación.

Formalmente, el algoritmo divide un conjunto de n puntos x en k agrupaciones disjuntas C, cada una descrita por la media μ_j de los puntos en la agrupación. Para ello, el algoritmo trata de encontrar los centroides que minimicen

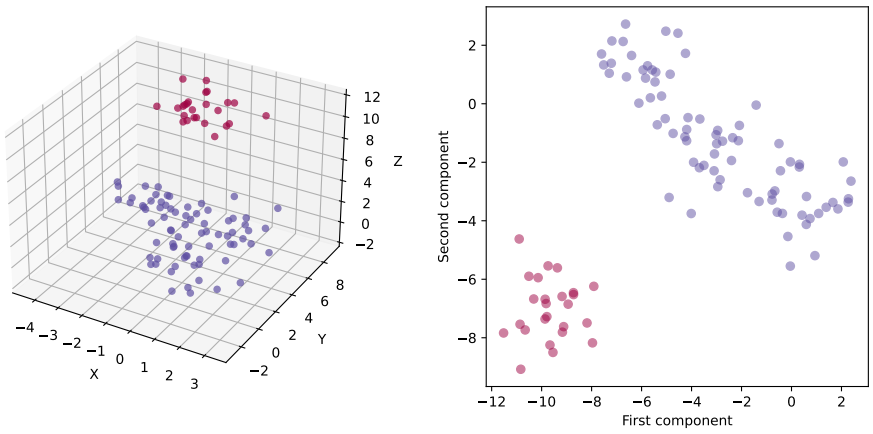
$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{2.1}$$

El algoritmo finaliza cuando una iteración no realice ninguna modificación de las agrupaciones.

Código 2.4: k-medias.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, n_init="auto"
kmeans_assigment = kmeans.fit_predict(X)
```



(a) (b)

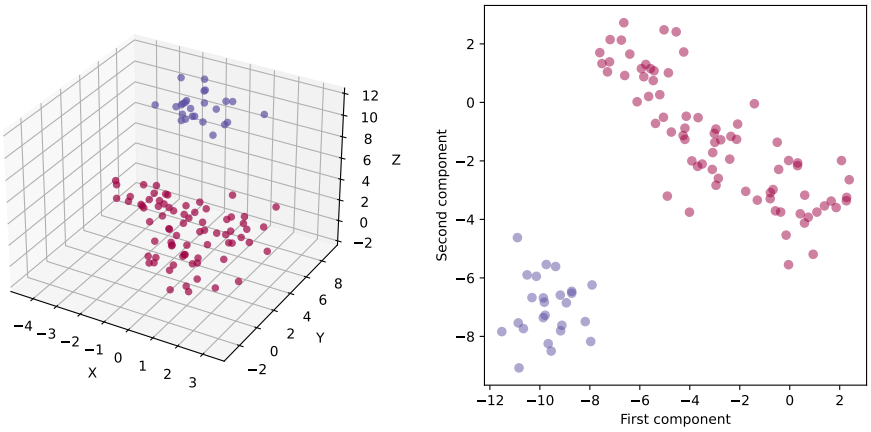
Figura 2.3: Prueba del algoritmo de k-medias. 2.3a Visualización sobre los datos sin modificar de los grupos que ha realizado el algoritmo de k-medias. 2.3b Visualización de los grupos realizados sobre los datos procesados por el PCA.

2.1.3. Agrupación aglomerada

(EXPLICACIÓN ALGORITMO)

Código 2.5: Agrupación aglomerada.

```
1  from sklearn.cluster import AgglomerativeClustering
2
3  agg = AgglomerativeClustering()
4  agg_assigment = agg.fit_predict(X)
```



(a) (b)

Figura 2.4: (EXPLICACIÓN FIGURA)

2.1.4. Agrupación por afinidad

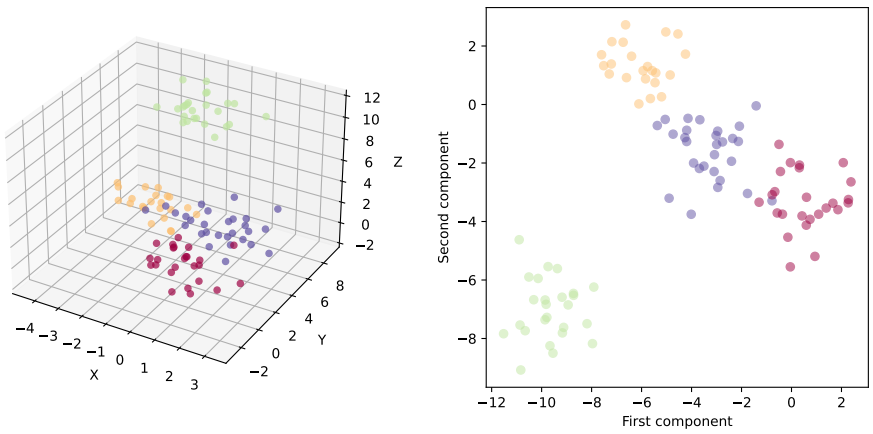
El algoritmo de propagación de afinidad crea agrupaciones mandando mensajes entre pares de puntos hasta que converge. La principal cualidad de este algoritmo es que, a diferencia con la mayoría de algoritmos de agrupamiento, no necesita saber el número de agrupaciones a realizar de antemano, sino que las genera dinámicamente.

(TERMINAR EXPLICACIÓN ALGORITMO)

Código 2.6: Propagación de afinidad.

```
from sklearn.cluster import AffinityPropagation

aff = AffinityPropagation()
aff_assigment = aff.fit_predict(X)
```



(a) (b)

Figura 2.5: (EXPLICACIÓN FIGURA)

2.2. Métodos supervisados

(POR HACER)

[5]

2.2.1. Red neuronal

Red secuencial

Red convolucional

3

Análisis y experimentación

Una vez adquiridos los conocimientos teóricos necesarios, y comprendido el funcionamiento de los diferentes algoritmos que vamos a utilizar, estamos en disposición de aplicarlos a los datos de experimentación y comprobar la eficacia de los algoritmos en un caso real. Este capítulo aborda el trabajo realizado durante tres meses en el Instituto Cajal, aplicando diferentes técnicas para tratar de agrupar los comportamientos de los animales, y de diferenciar animales bajo efectos de tratamiento o de placebo.

En la Sección 3.1 tratamos como hemos preprocesado los videos para homogeneizar todas las sesiones de experimentación, obtener los datos de posición de los animales en ellas y filtrar e interpolar los datos poco verosímiles. En la Sección 3.2 desarrollamos el proceso de análisis de los datos preprocesados aplicando los diferentes algoritmos vistos anteriormente y analizando sus resultados.

3.1. Preprocesado de datos

Los datos de los que hemos partido para realizar el análisis han sido 88 pares de videos en blanco y negro, de unos cinco minutos cada uno, de ratones en su caja sin estar realizando ninguna tarea concreta. Cada par de videos consistía en un video de la vista cenital de la caja y otro de la vista lateral, como se puede observar en la Figura 3.1. Además, de cada par de videos teníamos el código del animal que estaba siendo grabado, la fecha de la sesión y si el animal estaba bajo los efectos del tratamiento de NDMA en dicha fecha o no. Mostramos un resumen

de estos datos en la Tabla 3.1.

Video	Código del animal	Fecha de la sesión	Tratamiento
1	4128	2020-12-02	CONTROL
2	4128	2020-11-21	CONTROL
3	4128	2020-11-23	CONTROL
...	...	...	...
80	4108	2020-09-25	NMDA
81	4089	2020-09-18	NMDA
82	4089	2020-09-12	CONTROL
83	4089	2020-09-24	NMDA

Tabla 3.1: Fragmento del `DataFrame` que almacenaba los datos de las sesiones, incluyendo el número del video según se almacenaban en memoria, el código del animal, la fecha de la sesión y el estado del tratamiento.

Para tratar de averiguar de forma automática el tipo de tratamiento de cada sesión, podríamos entrenar una red neuronal utilizando directamente los videos como datos de entrenamiento. Sin embargo, no contamos con una cantidad muy elevada de videos para entrenar, y además esto no nos daría ninguna pista sobre como comportamientos concretos se relacionan con el tratamiento. Al no contar con una base de datos de fragmentos de videos clasificados según el comportamiento del animal, tampoco podemos entrenar una red para distinguir estos comportamientos sin tener que pasar antes por un seguramente tedioso periodo de etiquetado manual de fragmentos. Por ello, hemos hecho uso de la herramienta DeepLabCut para rastrear la posición del ratón en cada fotograma y así poder hacer cálculos sobre ella, y trabajar con otros algoritmos de agrupación no supervisada para tratar de clasificar diferentes movimientos del ratón a lo largo del video.

3.1.1. DeepLabCut

(EXPLICACIÓN DEEPLABCUT)

3.1.2. Filtrado e interpolación

Una vez hemos pasado todos los videos por DeepLabCut, tenemos dos ficheros de datos para cada uno de las sesiones, uno para las coordenadas de los puntos de la vista cenital y otro para las coordenadas de los puntos de la vista lateral. Al cargar los ficheros a memoria como `DataFrames` de `pandas`, vemos que no todos ellos tienen la misma duración, por lo que los homogeneizamos todos eliminando los datos de los últimos fotogramas, consiguiendo que cada `DataFrame` tenga un

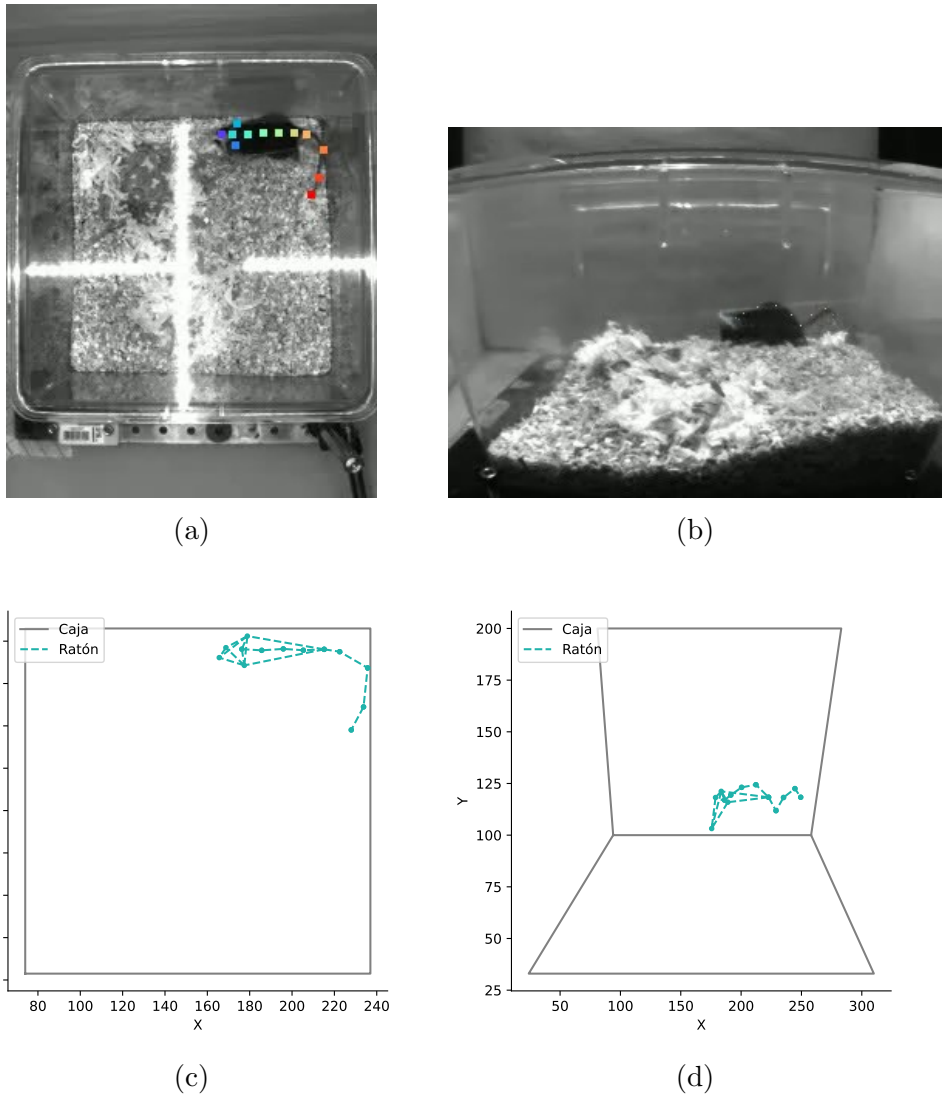


Figura 3.1: Salida de DeepLabCut del animal 4128 el 02-12-2020, 1:00:37. 3.1a Video de la vista cenital de la caja. Los puntos sobre el animal son los dibujados por DeepLabCut para rastrear las partes del animal. 3.1b Video de la vista lateral de la caja. 3.1c En azul, la triangulación formada por los puntos obtenidos por DeepLabCut de la vista cenital. Cinco puntos para la cabeza, cuatro para la espalda y otros cuatro para la cola. En gris, las dimensiones de la base de la caja, obtenidas midiendo las distancias en píxeles sobre un fotograma de video. 3.1d Análogo a la vista cenital, desde la vista lateral. En gris, la altura mínima del suelo de la caja y su pared trasera.

	Nosex	Nosey	Noselikelihood	Headx	Heady	...
0	136.165344	177.722496	0.000084	129.790253	174.772552	...
1	162.032005	201.444756	0.942181	168.152061	202.420639	...
2	156.297043	200.326378	0.000073	162.436028	203.156837	...
3	155.370415	199.043297	0.000277	159.507599	200.199928	...
4	149.272644	197.677170	0.000045	155.493912	198.814835	...
...	...	...	...	...	...	...

Tabla 3.2: Extracto del `DataFrame` de `pandas` de los datos sin procesar de DeepLabCut. Para cada punto rastreado, y para cada fotograma del video, se computa la predicción de la coordenada x y de la coordenada y y se da la verosimilitud de dicha predicción.

tamaño de 6000 filas, lo que equivale a 6000 fotogramas o 5 minutos de video. Además, eliminamos los datos de 4 videos que no llegan a esa duración, ya que son videos con problemas que entorpecerían el análisis. Finalmente, eliminamos los datos de 2 videos más, ya que tras el preprocesado seguían teniendo valores nulos debido a que DeepLabCut no ha podido identificar varios de los puntos a rastrear en ninguno de los fotogramas de los videos.

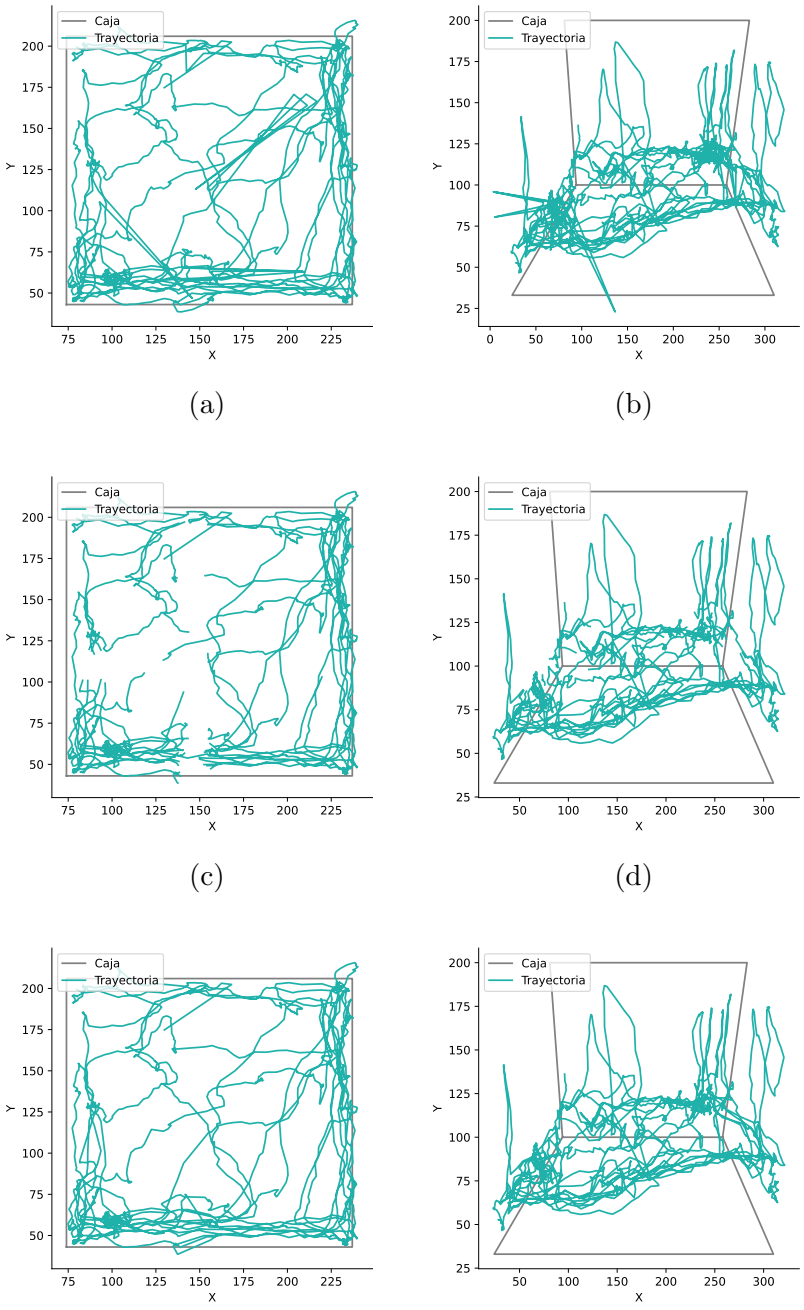


Figura 3.2: Vistas cenital y lateral de trayectorias de un punto de la cabeza del animal 4128. Datos correspondientes a los dos primeros minutos de la sesión del 2020-12-02. 3.2a, 3.2b: Vista cenital de los datos antes del preprocesado. En gris se ha dibujado las dimensiones de la caja y en color la trayectoria del punto `Head`. 3.2e, 3.2f: Vistas cenital y lateral de los datos una vez filtrados los valores de baja berosimilitud. 3.2e, 3.2f: Vistas cenital y lateral de los datos filtrados con los valores nulos interpolados.

3.2. Análisis de datos

3.2.1. Clasificación manual de comportamientos.

Gracias a los datos de DeepLabCut podemos tratar de identificar ciertos comportamientos de forma manual, especificando ciertos umbrales para extraer los comportamientos deseados. A modo de ejemplo, hemos calculado cuando el punto de la cabeza en la vista cenital, `head`, se salía durante un intervalo de cinco fotogramas fuera de los límites establecidos de la caja. Así, como se observa en la Figura 3.3, hemos identificado los intervalos de tiempo de cada sesión en los cuales el animal se alza sobre sus extremidades traseras apoyándose en la pared de la caja.

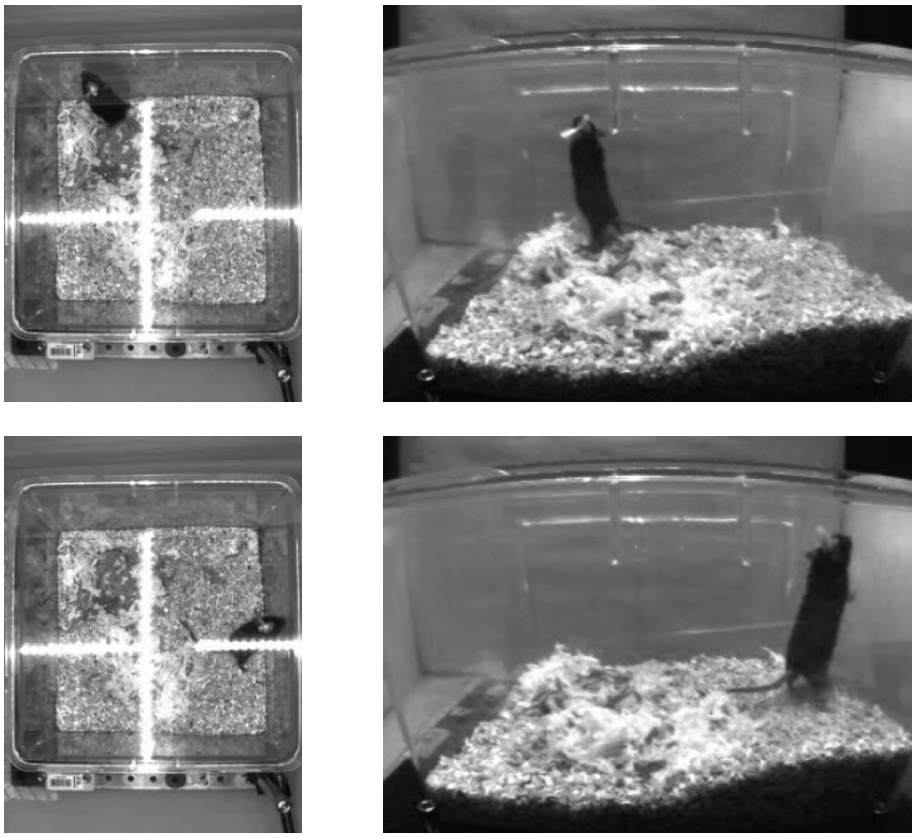


Figura 3.3: Fotogramas de los dos primeros intervalos en los que el animal 4128 se alza en la sesión del 2020-12-02. Los fotogramas superiores corresponden a la vista cenital y lateral del segundo 3,05. Los fotogramas inferiores corresponden a las mismas vistas del segundo 19,43.

De la misma forma, podemos calcular en que intervalos la media de los puntos de la espalda tienen una velocidad mayor de tres píxeles por segundo para identificar cuando el animal se está moviendo como se puede ver en la Figura 3.4, o

calcular el ángulo formado entre un vector de los puntos de la cabeza del animal y otro de los puntos de la espalda, para determinar si el animal está girado hacia algún lado, visualizado en la Figura 3.5.

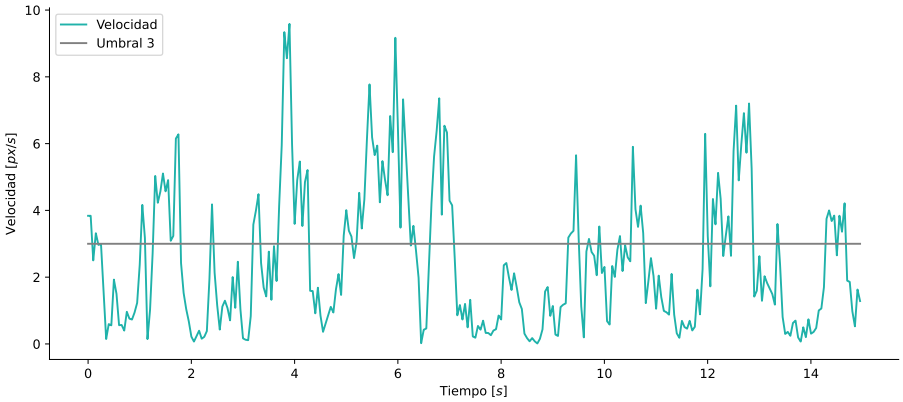


Figura 3.4: Gráfico de la velocidad del animal 4128, durante los primeros 15 segundos de la sesión del 2020-12-02. En color, los datos obtenidos derivando la trayectoria de la media de los puntos de la espalda del animal: `Neck`, `Back_1`, `Back_2`, `Back_3` y `Back_4`. En gris, el umbral arbitrario de 3 píxeles por segundo, para diferenciar cuando el animal está desplazándose y cuando no.

Estos métodos de extracción de comportamientos funcionan para este caso particular, pero al depender de funciones hechas específicamente para estos datos, sería complejo exportarlo a otros escenarios. DeepLabCut ayuda en ese aspecto, ya que no calculamos propiedades sobre los videos en sí, sino sobre un conjunto de coordenadas de posiciones de partes del animal. De esta forma, si se tratase de analizar videos de animales grabados en otras cajas o realizando alguna tarea, podríamos computar de forma sencilla estas propiedades ajustando los parámetros necesarios.

Sin embargo, para el cómputo manual de estas propiedades se han puesto umbrales arbitrarios para definir el estado del animal. Un umbral de 3 píxeles por segundo sobre la media de los puntos de la espalda para determinar cuando el animal se está desplazando, un umbral de $\pm 30°$ para determinar si el animal está girando o no, o un umbral de 5 fotogramas fuera de las dimensiones de la caja para determinar si el animal se está alzando sobre una pared. Los umbrales son necesarios si consideramos necesario discretizar estas variables continuas en comportamientos cualitativos, pero están a su vez sesgados por el juicio del experimentador, ya que una varianza en la magnitud de los umbrales o en el conjunto de puntos utilizados para calcularlos puede resultar en una discrepancia de la clasificación obtenida.

La clasificación manual hace que sea complejo determinar también cuando el animal realiza algún comportamiento más complejo que los vistos hasta ahora,

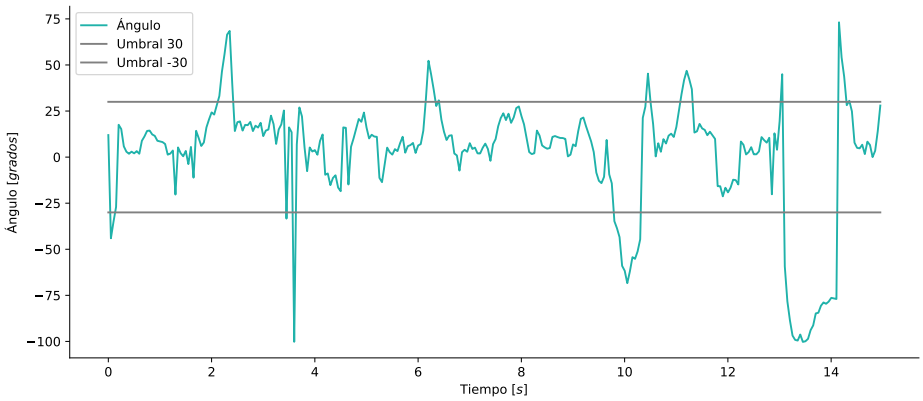


Figura 3.5: Gráfico de los ángulos del animal 4128, durante los primeros 15 segundos de la sesión del 2020-12-02. En color, el ángulo formado por el vector de los puntos de la cabeza con el vector de los puntos de la espalda. En gris, los umbrales de 30° y -30° para determinar si el animal está recto, o tiene el cuerpo girado hacia la izquierda o derecha respectivamente,

ya que si no sabemos de antemano la relación entre las coordenadas de las partes del animal y el comportamiento, no podremos identificarlo. De la misma forma, al no conocer de antemano la relación entre los diferentes comportamientos y el tipo de tratamiento dado a los animales, con estas clasificaciones no se podrán diferenciar animales tratados con anticuerpos de NDMA de animales control.

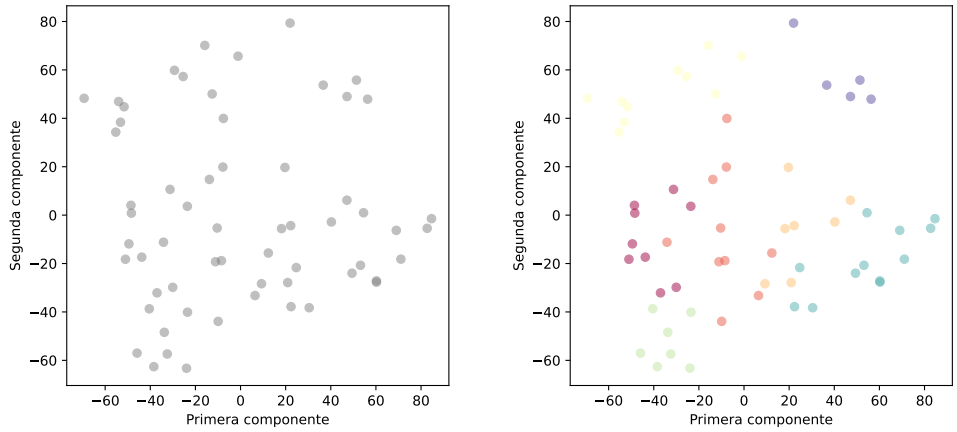
Esto motiva las siguientes secciones en las cuales hemos utilizado diversos métodos de aprendizaje automático para tratar de conseguir clasificaciones sin depender de sesgos preestablecidos.

3.2.2. Agrupaciones no supervisadas.

Para tratar de automatizar el proceso de clasificación, sin depender de cualidades computadas *ad hoc* para este conjunto de datos, hemos realizado los algoritmos vistos en la sección 2.1 sobre los datos de las trayectorias preprocesados según hemos visto en la sección 3.1.

Para clasificar los comportamientos de un animal en cada sesión no contamos con datos previamente clasificados por lo que no podemos hacer uso de métodos de aprendizaje automático supervisados. Además, si no queremos introducir un sesgo en la clasificación, tampoco sabemos de antemano en cuantos grupos de comportamiento similares podremos agrupar los diferentes fragmentos de cada sesión de video. Por ello hemos utilizado un algoritmo de propagación de afinidad, basándonos en el trabajo realizado en [6], con el cual podemos realizar una agrupación de intervalos de tiempo en función de su afinidad. Pese a todo

ello, hay que introducir un pequeño sesgo en el proceso, determinando arbitrariamente la longitud de los intervalos de tiempo que queremos clasificar. Hemos dividido los videos de las sesiones en intervalos de 5 segundos e introducido los datos de las coordenadas de todos los puntos de las vistas cenital y lateral para todos los puntos de cada fotograma de los intervalos. En la Figura 3.6 se muestra la agrupación obtenida para la sesión de uno de los animales, consiguiendo agrupar los distintos intervalos en 7 grupos de comportamiento.



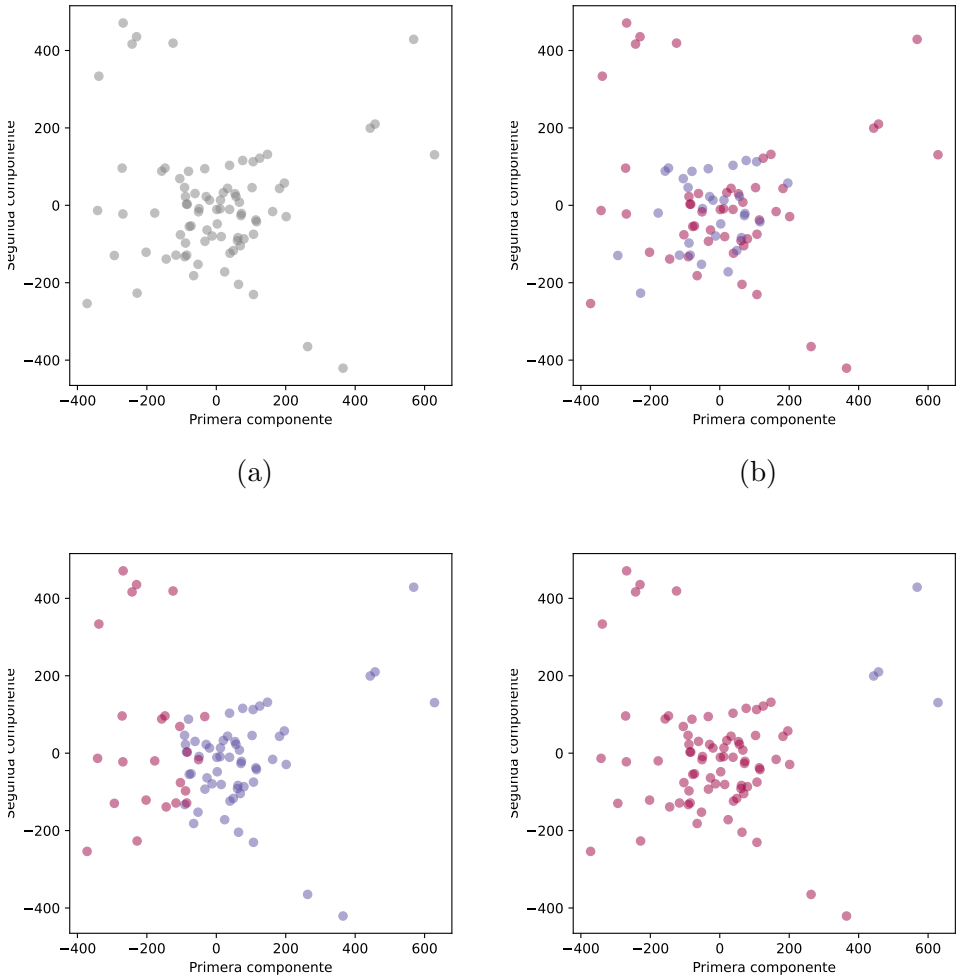
(a) (b)

Figura 3.6: Clasificación de comportamiento mediante el algoritmo de propagación de afinidad sobre intervalos de la sesión del animal 4128 el 2020-12-02. 3.6a: Distribución de los puntos de un PCA sobre una subdivisión de la sesión en 60 intervalos. Cada punto representa un intervalo de 5 segundos, 100 fotogramas, de la sesión de video. Cada punto original cuenta con 56 dimensiones, correspondientes a las coordenadas x e y de los 14 puntos extraídos por DeepLabCut tanto para la vista cenital como la vista lateral. Todos los puntos se han normalizado para tener media 0 y desviación 1 antes de realizar el PCA. Tras el PCA, y únicamente para su representación gráfica, se conservan las dos primeras componentes principales de cada punto. 3.6b: Misma representación de los 60 intervalos que en 3.6a, pero los puntos han sido coloreados según el número de grupo asignado por el algoritmo de propagación de afinidad. El algoritmo se ha realizado sobre los puntos de los intervalos normalizados, manteniendo las 56 dimensiones originales y utilizando la norma euclídea como medida de afinidad.

Para diferenciar el tipo de tratamiento de cada una de las sesiones utilizando los datos de DeepLabCut mediante algoritmos no supervisados, no podemos usar el algoritmo de propagación de afinidad, ya que no es posible garantizar que vaya a generar dos únicas clases. Por ello, hemos utilizado los otros dos algoritmos explicados en la sección 2.1: la agrupación por k-medias y la agrupación aglomerativa, ambos utilizados en estudios similares [7]. Para ambos casos hemos tomado los datos de todas las sesiones y tras aplicarles el algoritmo hemos mostrado los

resultados en la Figura 3.7. Con ninguno de los dos algoritmos hemos conseguido resultados significativamente mejores que los que obtendríamos haciendo asignaciones aleatorias de las clases, con lo que podemos asumir que no hay ninguna relación directa entre la posición de los animales y el tipo de tratamiento que se les ha suministrado.

Para conseguir realizar esta clasificación habría que computar otras características manualmente con las que dar más información a los algoritmos, cambiar los puntos rastreados por DeepCutLab para conseguir información sobre otras partes del animal, o tratar de predecir una función desconocida que fuera capaz de agrupar las sesiones correctamente. Como tenemos los datos de todas las sesiones podemos entrenar métodos supervisados, y estamos en las condiciones idóneas para tratar de predecir, en caso de que exista, esa función que relacione las coordenadas de los puntos que hemos rastreado con el tipo de tratamiento de cada sesión.



(a) (b)

(c) (d)

Figura 3.7: Clasificaciones no supervisadas del tipo de tratamiento. 3.7a: Distribución de los puntos de un PCA realizado sobre los datos de las 81 sesiones preprocesados y normalizados. Cada punto tiene una dimensión original de 6000 fotogramas por 52 coordenadas por fotograma, que se han linealizado para tener 312000 dimensiones por punto antes de realizar el PCA. 3.7b: Asignación real del estado de cada tratamiento para cada sesión. En azul se han marcado las sesiones bajo el efecto de anticuerpos de NMDA y en rosa se han marcado las sesiones en las que no hay efecto, o se ha suministrado un tratamiento placebo. 3.7c: Resultados de una agrupación mediante un algoritmo de k-medias sobre los puntos de las 81 sesiones con todas las dimensiones, mostrado sobre los puntos obtenidos mediante el PCA. Clases agrupadas con una precisión del $51,85\,\%$. 3.7d Resultados de una agrupación mediante un algoritmo de agrupación aglomerada sobre los puntos de las 81 sesiones con todas las dimensiones, mostrado sobre los puntos obtenidos mediante el PCA. Clases agrupadas con una precisión del $58,02\,\%$.

3.2.3. Agrupaciones supervisadas.

Para aproximar una función que relaccióne

Código 3.1: Red secuencial

```
model = nn.Sequential(
          nn.Linear(52*1200, 1024),
          nn.Tanh(),
          nn.Linear(1024, 512),
          nn.Tanh(),
          nn.Linear(512, 128),
          nn.Tanh(),
          nn.Linear(128, 2)).to(device=device)
```

Código 3.2: Red convolucional

```
class TimeSeriesNet(nn.Module):
  def __init__(self):
      super().__init__()
      self.conv1 = nn.Conv1d(in_channels=52, out_channels=104,
         kernel_size=3, padding=1)
      self.conv2 = nn.Conv1d(in_channels=104, out_channels=8,
         kernel_size=3, padding=1)
      self.fc1 = nn.Linear(2400, 32)
      self.fc2 = nn.Linear(32, 2)

  def forward(self, x):
      # Input shape: (batch_size, channels, sequence_length)
      # Max pooling with kernel size 2
      out = F.max_pool1d(torch.tanh(self.conv1(x)), 2)
      # Max pooling with kernel size 2
      out = F.max_pool1d(torch.tanh(self.conv2(out)), 2)
      # Reshape for fully connected layer
      out = out.view(-1, 2400)
      out = F.tanh(self.fc1(out))
      out = self.fc2(out)
      return out

model = TimeSeriesNet()
```

4

Resultados

(Por redactar)

5

Conclusiones

(Por redactar)

Bibliografía

[1] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, 2018. [Online]. Available: https://doi.org/10.1038/s41593-018-0209-y

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[4] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python*. O'Reilly, 2017.

[5] E. Stevens, L. Antiga, and T. Viehmann, *Deep Learning with PyTorch*. Manning, 2020.

[6] A. Klaus, G. J. Martins, V. B. Paixao, P. Zhou, L. Paninski, and R. M. Costa, "The spatiotemporal organization of the striatum encodes action sapce," *Neuron*, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.neuron.2017.08.015

[7] T. Menaker, J. Monteny, L. O. de Beeck, and A. Zamansky, "Clustering for automated exploratory pattern discovery in animal behavioral data," *Frontiers in Veterinary Science*, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fvets.2022.884437

Apéndices

A

Apéndice 1

A.1. Salida de la validación cruzada de la red convolucional.

```
Fold 1
-------
Epoch 1, Training loss 0.6996589693529852
Epoch 10, Training loss 0.3914181091662111
Epoch 20, Training loss 0.028187466969970484
Epoch 30, Training loss 0.006742463795061426
Epoch 40, Training loss 0.003754144479271731
Epoch 50, Training loss 0.002467109620895293
Epoch 60, Training loss 0.0018386000028265531
Epoch 70, Training loss 0.0014798810073537833
Epoch 80, Training loss 0.001202181870766431
Epoch 90, Training loss 0.0010188377396927494
Epoch 100, Training loss 0.0009259502981124036
Accuracy train: 100.0%
Accuracy val: 53.84615384615385%

Fold 2
-------
Epoch 1, Training loss 0.6590047591719134
Epoch 10, Training loss 0.3379397751099762
Epoch 20, Training loss 0.02254703572025287
Epoch 30, Training loss 0.006650687512826581
Epoch 40, Training loss 0.003737638848659786
Epoch 50, Training loss 0.0025370304577116824
Epoch 60, Training loss 0.0019184314137680627
```

```
Epoch 70, Training loss 0.00152091032581164
Epoch 80, Training loss 0.0012658066058567532
Epoch 90, Training loss 0.0010788602454737685
Epoch 100, Training loss 0.00092578063789727
Accuracy train: 100.0%
Accuracy val: 60.0%

Fold 3
-------
Epoch 1, Training loss 0.667039497145291
Epoch 10, Training loss 0.4190175721700164
Epoch 20, Training loss 0.04101331377851552
Epoch 30, Training loss 0.010446132586910337
Epoch 40, Training loss 0.005377942471411721
Epoch 50, Training loss 0.003532323196139615
Epoch 60, Training loss 0.0026109476099406294
Epoch 70, Training loss 0.002050086105544784
Epoch 80, Training loss 0.0016736525467840485
Epoch 90, Training loss 0.0014004294222187995
Epoch 100, Training loss 0.0011978454296758141
Accuracy train: 100.0%
Accuracy val: 64.61538461538461%

Fold 4
-------
Epoch 1, Training loss 0.6882521503273098
Epoch 10, Training loss 0.31157968178305817
Epoch 20, Training loss 0.017680644784617804
Epoch 30, Training loss 0.005089276510554409
Epoch 40, Training loss 0.0028648498048604435
Epoch 50, Training loss 0.001965091866612516
Epoch 60, Training loss 0.001495680903320752
Epoch 70, Training loss 0.00118541678366641
Epoch 80, Training loss 0.0010830991498018004
Epoch 90, Training loss 0.0008304081024232738
Epoch 100, Training loss 0.0007286544534466905
Accuracy train: 100.0%
Accuracy val: 60.0%

Fold 5
-------
Epoch 1, Training loss 0.6766076351719341
Epoch 10, Training loss 0.2829631914909201
Epoch 20, Training loss 0.04570655433605585
Epoch 30, Training loss 0.005027402004015621
Epoch 40, Training loss 0.0028682147691644535
Epoch 50, Training loss 0.001964785525442807 5
Epoch 60, Training loss 0.001485317545344493 8
Epoch 70, Training loss 0.001183477345886325 7
Epoch 80, Training loss 0.000980541127852308 7
Epoch 90, Training loss 0.000835897151334612 6
Epoch 100, Training loss 0.0007206922087663833
Accuracy train: 100.0%
```

```
Accuracy val: 57.8125%

All folds completed!
Mean accuracy: 59.25480769230769%
```