Universidad Autónoma De Yucatán

Facultad de Matemáticas

Reconocimiento de la Actividad Humana Basada en la Fusión de Datos con Python 2025

Maestro:

Antonio Armando Aguileta Güemez

Estudio Comparativo de Modelos de Fusión para la Clasificación de Actividades Humanas

Equipo 3:

Aguilar Pérez Johan Ricardo

Cabrera Alcocer Herberth Josueh

Canto Paredes Rodrigo Adrián

Dzul López Alex Enrique

Montero Salazar Luis Ángel

Fecha de entrega: 08/05/2025

1. Introducción:

El reconocimiento de actividades humanas (Human Activity Recognition, HAR) a partir de sensores portables ha cobrado relevancia en diversas aplicaciones como la salud, el deporte, la seguridad y la asistencia a personas mayores. Para mejorar la precisión y robustez de los sistemas de HAR, se han propuesto diversas estrategias de fusión de datos y modelos, que permiten aprovechar múltiples fuentes de información y técnicas de aprendizaje automático.

Este trabajo presenta un estudio comparativo de distintos modelos de fusión para la clasificación de actividades humanas, evaluando su desempeño en múltiples bases de datos públicas. Se hace énfasis en analizar qué enfoques ofrecen mayor estabilidad y generalización a través de métricas de desempeño como la exactitud, precisión, recall y F1-score, empleando validación cruzada.

1.1 Objetivos:

- Evaluar el desempeño de distintos modelos de fusión de clasificadores (Bosques Aleatorios agregados, Votación combinada y AdaBoost) para el reconocimiento de actividades humanas.
- Analizar el rendimiento de estos modelos sobre diferentes bases de datos públicas de HAR.
- Determinar qué modelo de fusión muestra mayor estabilidad y generalización en distintos contextos de datos.
- Proporcionar una base empírica para la elección de técnicas de fusión en futuras aplicaciones de HAR.

1.2 Metodología

I. Selección de Bases de Datos: Se utilizaron tres conjuntos de datos públicos con información de actividades humanas recolectada mediante sensores portátiles: Smartphone-Based Recognition of Human Activities and Postural Transitions Dataset, Daily Sports and Activities Dataset, y, MHEALTH Dataset.
II. Preprocesamiento: Incluyó limpieza de datos, manejo de valores faltantes, normalización, escalado y extracción de características con ventanas de tiempo de 3 segundos sin traslape, conforme a referencias del estado del arte.
III. Entrenamiento de Clasificadores Base: Se entrenaron modelos individuales (Regresión Logística, Bosques Aleatorios y Perceptrón) sobre cada conjunto de datos.
IV. Reducción de Dimensionalidad: Se aplicaron técnicas como PCA, KPCA y selección de características para optimizar el espacio de entrada.
V. Modelos de Fusión: Se implementaron tres estrategias de fusión:
    - Agregación con Bosques Aleatorios.
    - Votación de clasificadores (Regresión Logística, Random Forest y Naive Bayes).
    - Boosting mediante AdaBoost.

VI. Evaluación del Desempeño: Se aplicó validación cruzada para medir el desempeño de cada enfoque con métricas estándar (accuracy, precision, recall y F1-score).
VII. Análisis Comparativo: Se realizó una evaluación comparativa del desempeño de los modelos de fusión en las distintas bases, incluyendo análisis estadístico exploratorio para respaldar las observaciones.

2. Antecedentes y estado del arte

2.1 Reconocimiento de la actividad humana

El Reconocimiento de la Actividad Humana (HAR, por sus siglas en inglés) es un área de investigación centrada en identificar y clasificar patrones de movimiento humano a partir de datos sensoriales. Este campo ha tenido un crecimiento importante debido a su aplicabilidad en sectores como la salud, la rehabilitación, el deporte, la domótica y la seguridad personal.

HAR puede implementarse con técnicas de visión por computadora, pero los sistemas basados en sensores portables han ganado tracción por su bajo costo, menor intrusión y capacidad de monitoreo continuo. Estos sistemas recolectan datos crudos de movimiento corporal para inferir actividades como caminar, correr, sentarse, subir escaleras, entre otras [12].

2.2 Sensores en la actividad humana

1. Acelerómetro de tres ejes

El acelerómetro es un sensor que mide la aceleración lineal de un objeto en uno, dos o tres ejes. Un acelerómetro de tres ejes puede detectar variaciones de velocidad (positivas o negativas) a lo largo de los ejes X, Y y Z. Su funcionamiento se basa principalmente en principios mecánicos y eléctricos, con tecnología MEMS (Sistemas Microelectromecánicos) ampliamente adoptada [12].

En los sensores MEMS, una pequeña masa está suspendida por microresortes dentro de una estructura. Cuando el sensor se mueve, la inercia de la masa provoca una desviación que se traduce en un cambio de capacitancia eléctrica, el cual es procesado para determinar la aceleración. Esta medida incluye tanto aceleraciones dinámicas (movimiento) como la estática debida a la gravedad, lo que permite usar el acelerómetro también como inclinómetro.

Características clave:

Rango de medida típico: ±2 g a ±16 g.

Alta sensibilidad a movimientos rápidos.

Útil para detectar caídas, vibraciones, orientación e inclinación.

Puede presentar ruido en entornos con muchas vibraciones mecánicas.

Aplicaciones:

Detección de pasos o gestos en dispositivos móviles.

Medición de vibraciones en maquinaria industrial.

Estabilización en cámaras o drones.

Sistemas de airbag en automóviles.

2. Giroscopio de tres ejes

El giroscopio mide la velocidad angular, es decir, la rapidez con la que un objeto gira alrededor de un eje. En su versión de tres ejes, proporciona datos de rotación en los tres ejes cartesianos, lo que permite conocer cambios de orientación en tiempo real [20].

Los giroscopios MEMS funcionan mediante el efecto Coriolis. Dentro del sensor, una microestructura vibra en un plano. Cuando el sensor rota, la vibración se desvía debido a la fuerza de Coriolis, y esta desviación se traduce en señales eléctricas que permiten calcular la velocidad angular.

Características clave:

Rango típico de medida: ±250 °/s a ±2000 °/s.

Muy sensible a cambios rápidos de orientación.

Acumula error con el tiempo (deriva), especialmente sin calibración o sin fusión con otros sensores.

A diferencia del acelerómetro, no responde a la gravedad.

Aplicaciones:

Navegación inercial y sistemas de guiado.

Estabilización de plataformas móviles.

Seguimiento de movimiento en realidad virtual y videojuegos.

Control de vuelo en drones y aeronaves.

3. Magnetómetro de tres ejes

El magnetómetro detecta la intensidad y dirección del campo magnético en los tres ejes. Su uso más común es como brújula electrónica, al medir el campo magnético terrestre y determinar el rumbo del dispositivo con respecto al norte magnético [12].

Este sensor puede funcionar bajo diferentes principios físicos, siendo comunes los magnetorresistivos anisotrópicos (AMR), efecto Hall o magnetoinductivos. La lectura del campo magnético puede verse afectada por interferencias locales, como campos generados por motores o materiales ferromagnéticos.

Características clave:

Sensible a campos magnéticos muy débiles (en el rango de micro teslas).

Permite estimar la orientación absoluta (rumbo), pero requiere corrección de inclinación si se usa junto a acelerómetros.

Es afectado fácilmente por interferencias electromagnéticas y requiere calibración.

Aplicaciones:

Sistemas de navegación terrestre o marítima.

Orientación de smartphones y tablets.

Detección de metales.

Instrumentación científica y medición de campo magnético ambiental.

4. Electrocardiograma (ECG)

El electrocardiograma (ECG) es una técnica médica no invasiva utilizada para registrar la actividad eléctrica del corazón a lo largo del tiempo mediante electrodos colocados en la superficie del cuerpo. Esta señal eléctrica refleja la despolarización y repolarización del músculo cardíaco durante cada ciclo cardíaco, lo que permite evaluar el ritmo, la frecuencia y la conducción eléctrica del corazón [21].

El registro obtenido se denomina trazado electrocardiográfico y consiste en una serie de ondas características:

- Onda P, que representa la despolarización auricular.
- Complejo QRS, que refleja la despolarización ventricular.
- Onda T, asociada con la repolarización ventricular.
- En algunos casos, también se observa la onda U, cuya fisiología aún no está completamente definida.

El ECG puede realizarse con distintos números de derivaciones: el más común en la práctica clínica es el de 12 derivaciones, que proporciona una visión integral del corazón desde diferentes ángulos (planos frontal y horizontal). También existen versiones portátiles y de bajo consumo energético con 1 a 3 derivaciones, comúnmente empleadas en sistemas de monitoreo ambulatorio o wearable.

Principio de funcionamiento:

Cada vez que el corazón se activa eléctricamente, genera pequeñas variaciones de potencial eléctrico que se propagan por el cuerpo. Los electrodos detectan estas señales, que son amplificadas y procesadas por el dispositivo para producir un registro visual (en papel o digital). La señal del ECG es de bajo voltaje (del orden de milivoltios) y requiere circuitos de alta ganancia y filtrado para eliminar interferencias (ruido muscular, artefactos por movimiento, interferencia de la red eléctrica, etc.).

Parámetros clave que pueden analizarse:

Frecuencia cardíaca

Intervalos PR, QRS y QT

Eje eléctrico cardíaco

Presencia de arritmias, bloqueos o isquemia miocárdica

Aplicaciones:

Diagnóstico de enfermedades cardiovasculares como arritmias, infarto de miocardio, miocardiopatías y trastornos de conducción.

Integración en dispositivos portátiles para telemedicina o monitoreo remoto.

2.3 Técnicas de aprendizaje máquina

Técnicas de Aprendizaje Máquina

En el contexto del aprendizaje supervisado, diversas técnicas permiten modelar patrones complejos a partir de datos etiquetados. A continuación, se describen tres métodos ampliamente

utilizados por su eficacia en clasificación binaria y multiclase, su capacidad de generalización y su aplicabilidad a múltiples dominios.

1. Regresión Logística

La regresión logística es una técnica de clasificación estadística utilizada para modelar la probabilidad de que una observación pertenezca a una de dos clases posibles [20]. A diferencia de la regresión lineal, que predice un valor continuo, la regresión logística utiliza la función sigmoide (logística) para mapear cualquier valor real al intervalo [0, 1], lo que permite interpretar la salida como una probabilidad.

Función principal:

$$P(y=1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}}$$

Donde son los coeficientes del modelo que se aprenden mediante técnicas como la máxima verosimilitud.

Características:

Sencilla de implementar e interpretar.

Requiere que las variables independientes no estén altamente correlacionadas.

No asume una distribución normal de las variables independientes.

Puede extenderse a clasificación multiclase mediante esquemas como One-vs-Rest o Softmax Regression.

Aplicaciones:

Diagnóstico médico (presencia/ausencia de una condición).

Detección de fraudes.

Clasificación de correos como spam o no spam.

2. Bosques Aleatorios (Random Forest)

El bosque aleatorio es un algoritmo de aprendizaje ensamblado basado en árboles de decisión. Consiste en una colección de árboles entrenados de manera independiente sobre subconjuntos aleatorios del conjunto de datos y características [20]. La decisión final se toma por votación mayoritaria (clasificación) o promedio (regresión).

Funcionamiento básico:

Cada árbol se entrena con una muestra aleatoria con reemplazo del conjunto de entrenamiento (bootstrap).

En cada nodo de decisión, se selecciona aleatoriamente un subconjunto de características para determinar la mejor división.

La diversidad entre los árboles reduce la varianza del modelo y mejora su capacidad de generalización.

Ventajas:

Robusto ante sobreajuste (overfitting).

Puede manejar datos faltantes y variables categóricas.

Importancia de características fácilmente extraíble.

Escalable a grandes volúmenes de datos.

Limitaciones:

Menor interpretabilidad que un único árbol.

Requiere más recursos computacionales.

Aplicaciones:

Detección de enfermedades en sistemas médicos.

Clasificación de texto o imágenes.

Predicción de riesgo financiero.

3. Perceptrón

El perceptrón es uno de los modelos más básicos de redes neuronales artificiales y constituye la base de redes más complejas como el perceptrón multicapa (MLP) [20][22]. Fue propuesto por Frank Rosenblatt en 1958 como un clasificador lineal binario.

Definición: Dado un vector de entrada, el perceptrón calcula una salida binaria mediante la función:

$$y = \begin{cases} 1 & \text{si } w \cdot x + b \geq \\ 0 & \text{si } w \cdot x + b < \end{cases}$$

Donde ***w*x*** es el vector de pesos y ***b*** el sesgo. El algoritmo de aprendizaje ajusta los pesos iterativamente en función del error de predicción, aplicando una regla de actualización simple:

$$w \leftarrow w + \eta(y_{\text{real}} - y_{\text{predicho}})x$$

Características:

Solo es capaz de resolver problemas linealmente separables.

Rápido y eficiente para tareas simples.

Constituye la base de redes neuronales más complejas.

Limitaciones:

No puede resolver problemas como XOR sin capas ocultas.

Requiere normalización de entrada para converger más fácilmente.

Aplicaciones:

Clasificación binaria simple.

Detección de patrones lineales.

Implementación en hardware debido a su bajo costo computacional.

3. Bases de datos

3.1 Base de Datos: Smartphone-Based Recognition of Human Activities and Postural Transitions Dataset

El conjunto de datos de reconocimiento de actividades y transiciones posturales [1] se creó a partir de grabaciones de 30 sujetos que realizaron actividades básicas y transiciones posturales mientras llevaban un teléfono inteligente montado en la cintura con sensores inerciales integrados. Los experimentos se realizaron con 30 voluntarios de entre 19 y 48 años. Siguiendo un protocolo de actividades, realizaron seis actividades básicas: tres posturas estáticas (de pie, sentado y acostado) y tres actividades dinámicas (caminar, bajar escaleras y subir escaleras). También se incluyeron transiciones posturales entre las posturas estáticas: de pie a sentado, de sentado a de pie, de sentado a acostado, de acostado a sentado, de pie a acostado y de acostado a de pie.

Cada participante llevaba un teléfono inteligente (Samsung Galaxy S II) en la cintura durante la ejecución del experimento. Se capturaron señales de aceleración lineal en tres ejes y velocidad angular en tres ejes a una frecuencia constante de 50 Hz, utilizando el acelerómetro y el giroscopio integrados del dispositivo.

Para cada participante, se lleva un registro de estos datos dentro de los 3 ejes principales (X, Y, Z), para cada uno de los dos sensores. En un archivo llamado “labels.txt”, se cuenta con varias filas de información con 5 datos cada fila, separados por espacios. Estos datos representan lo siguiente:

Columna 1: ID de experimento

Columna 2: ID de usuario

Columna 3: ID de actividad

Columna 4: Punto de inicio de la etiqueta (en número de muestras a 50hz)

Columna 5: Punto de fin de la etiqueta (en número de muestras a 50hz)

Se cuenta también con un archivo “activity_labels.txt”, el cual nos permite asociar cada ID de actividad con su nombre correspondiente. En pasos posteriores observaremos que trabajamos con esta información para construir un DataFrame más completo. Este archivo correspondiente a las asociaciones entre IDs de actividades y sus nombres contiene lo siguiente:

```
1 WALKING
2 WALKING_UPSTAIRS
3 WALKING_DOWNSTAIRS
4 SITTING
5 STANDING
6 LAYING
7 STAND_TO_SIT
8 SIT_TO_STAND
9 SIT_TO_LIE
10 LIE_TO_SIT
11 STAND_TO_LIE
12 LIE_TO_STAND
```

3.1.1 Dimensiones

En lo que respecta a dimensiones, podemos reconocer que contamos con un total de 815,614 lecturas en total, y 6 características (eje x, y, z del sensor acelerómetro, eje x, y, z del sensor giroscopio). Para efectos de una visualización más sencilla y amena, se agregó una columna extra "Activity", que detona la actividad correspondiente a cada muestra, asociación que se llegó a conseguir mediante el archivo "labels.txt". La estructura de la base de datos queda, entonces, de la siguiente manera;

	X axis_acc	Y axis_acc	Z axis_acc	X axis_gyro	Y axis_gyro	Z axis_gyro	Activity
0	1.020833	-0.125000	0.104167	-0.000916	0.001833	0.002749	STANDING
1	1.020833	-0.125000	0.105556	-0.002749	-0.004276	0.002749	STANDING
2	1.025000	-0.125000	0.101389	-0.000305	-0.002138	0.006109	STANDING
3	1.020833	-0.125000	0.104167	0.012217	0.000916	-0.007330	STANDING
4	1.016667	-0.125000	0.108333	0.011301	-0.001833	-0.006414	STANDING
...	...	...	...	...	...	...	...
815609	0.950000	-0.443056	-0.187500	1.184773	1.112386	-0.307876	WALKING_UPSTAIRS
815610	0.880556	-0.390278	-0.156944	1.163698	1.106277	-0.374155	WALKING_UPSTAIRS
815611	0.834722	-0.358333	-0.098611	1.177137	1.023810	-0.388816	WALKING_UPSTAIRS
815612	0.802778	-0.329167	-0.104167	1.213484	0.918130	-0.332311	WALKING_UPSTAIRS
815613	0.770833	-0.287500	-0.098611	1.326188	0.846659	-0.202502	WALKING_UPSTAIRS

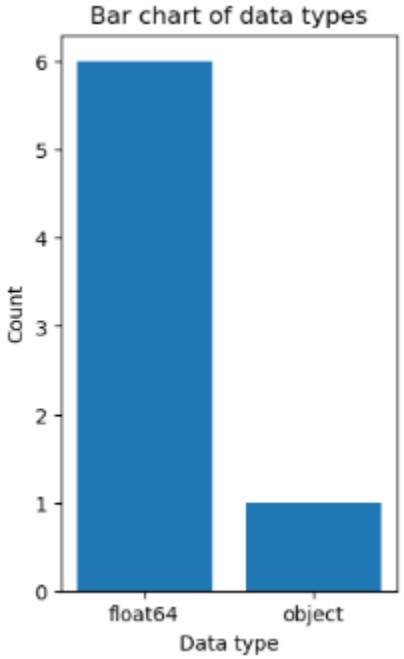
815614 rows × 7 columns

```
Dimensions: (815614, 7)
```

Donde se observa un tamaño de la base de datos correspondiente a 815,614 filas, propias de la cantidad de muestras con las que contamos a lo largo del tiempo, y 7 columnas, donde 6 de ellas son de las lecturas propias de cada eje para cada sensor, y nuestra última columna de actividad agregada, para asociar la muestra con su respectiva actividad real.

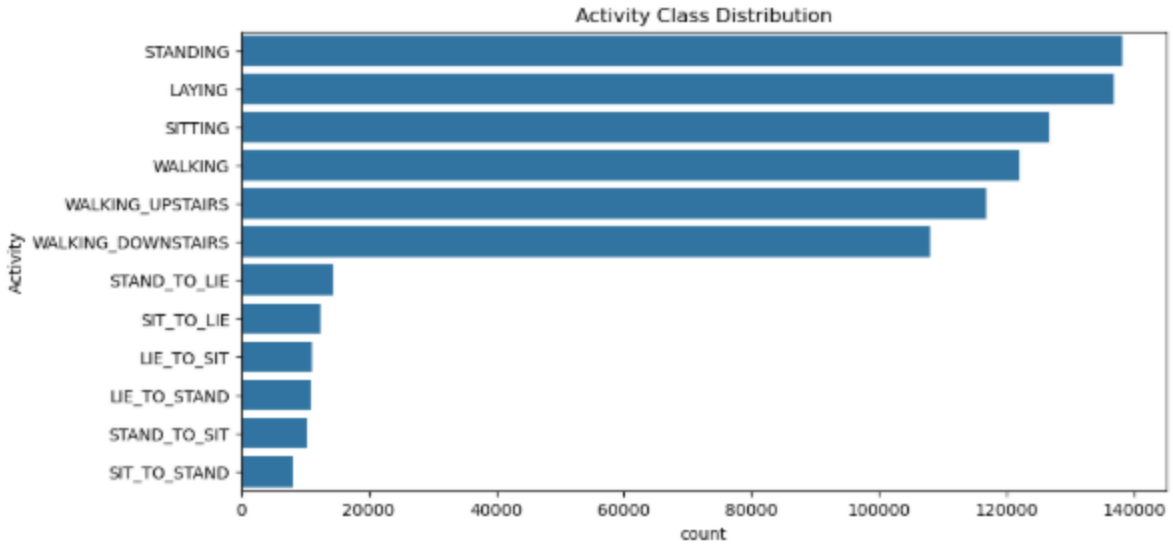
3.1.2 Atributos, distribución y resúmenes estadísticos

Antes de proceder con cualquier técnica de preprocesamiento de datos, debemos entender la estructura de estos [2], lo cual involucra examinar la distribución, escalas, valores atípicos y correlaciones entre características, entre otros. Es debido a lo anterior que procede la descripción de los atributos con los que contamos, así como sus resúmenes estadísticos. Es importante aclarar que, como bien mencionan los mismos autores, no poseemos con datos nulos, por lo que resulta innecesario el filtrarlos. Primeramente, observamos los tipos de datos:



Donde podemos ver que, excluyendo la columna categórica de la actividad, todas las demás columnas son de tipo flotante, por lo que contamos con información meramente cuantitativa, lo cual es importante para saber cómo proceder con las técnicas de preprocesamiento y clasificadores [3].

En tema de distribución, por clase, contamos con lo siguiente:



A simple vista se puede observar un claro desbalance en la distribución de clases, donde las primeras 6 de ellas tienen aproximadamente la misma densidad de aparición, y las siguientes 6 disminuyen drásticamente en cantidad, aproximadamente en un 80% respecto al grupo mayoritario. El manejo correcto de las clases y de este desbalance es fundamental para poder evitar el overfitting en el caso de las clases más representativas, o underfitting en el caso de las clases minoritarias. Más adelante se trabajará con esta información para tomar decisiones y poder observar la diferencia de desempeño de los clasificadores con un desbalance considerable respecto con aquellos clasificadores entrenados bajo un conjunto de datos con puntos sintéticos para las clases minoritarias con la ayuda del Borderline SMOTE.

Respecto a los resúmenes estadísticos, debido a la carga computacional de técnicas como la transformación Fourier y otras [4], se optó por obtener los siguientes estadísticos, que resultan ser lo suficientemente básicos como para agilizar el proceso de obtención de estos, pero a la vez lo suficientemente descriptivos para brindarnos información valiosa sobre los datos:

• Media: El valor promedio de la aceleración o velocidad angular para cada ventana.

• Desviación estándar: La medida de la dispersión de los valores de aceleración o velocidad angular alrededor de la media.

• Varianza: El grado de dispersión que indica cómo la aceleración o velocidad angular difiere de la media. También complementa la desviación estándar.

• Mediana: El valor central de la aceleración o velocidad angular ordenada en cada ventana.

• Mínimo: El valor más pequeño registrado de aceleración o velocidad angular.

• Máximo: El valor más grande registrado de aceleración o velocidad angular.

• Asimetría (Skewness): La medida de la asimetría en la distribución de los datos del acelerómetro o giroscopio.

• Curtosis: La medida del grado de apuntamiento o achatamiento en la distribución de los datos del acelerómetro o giroscopio.

• Magnitud: La magnitud total de la aceleración o velocidad angular, calculada como la raíz cuadrada de la suma de los componentes al cuadrado (tanto para los datos del acelerómetro como del giroscopio).

Obteniendo consigo los valores siguientes:

	X axis_acc	Y axis_acc	Z axis_acc	X axis_gyro	Y axis_gyro	Z axis_gyro
count	815614.000000	815614.000000	815614.000000	815614.000000	815614.000000	815614.000000
mean	0.806468	0.032947	0.087695	0.009292	-0.003563	-0.005986
std	0.409161	0.405570	0.345823	0.440613	0.411000	0.298523
min	-0.647222	-1.698611	-1.837500	-5.249165	-6.323371	-3.124270
25%	0.675000	-0.223611	-0.119444	-0.048258	-0.073915	-0.038485
50%	0.950000	-0.072222	0.045833	0.007025	0.000611	-0.000305
75%	1.019445	0.213889	0.238889	0.077885	0.047953	0.044593
max	2.004167	1.716667	1.502778	5.964488	6.240294	3.408628

```
Skewness:

X axis_acc    -0.707468
Y axis_acc     0.731837
Z axis_acc     0.390440
X axis_gyro   -0.204585
Y axis_gyro    0.393263
Z axis_gyro   -0.202577
dtype: float64

Kurtosis:

X axis_acc     0.256857
Y axis_acc     0.036421
Z axis_acc     0.542543
X axis_gyro    8.599942
Y axis_gyro    8.991414
Z axis_gyro    6.135584
dtype: float64
```

Primeramente, podemos observar que contamos con la misma cantidad de lecturas en todos los ejes de ambos sensores. Esto resulta beneficioso para nosotros porque así nos aseguramos de que ciertos datos no estén representados en mayor o menor medida respecto a otros, y lleguemos a conclusiones bajo contextos distintos.

Enseguida tenemos información en cuanto a la media, donde notamos principalmente la tendencia de dirección de ejes, donde el eje X y Z de acelerómetro tienen una tendencia positiva, pero es el eje X el que tiene una tendencia altamente positiva con una media muy superior a 0 (0.83). Esto nos da a entender que las lecturas favorecen esta dirección en particular de este eje (X) sobre este sensor (acelerómetro).

Después tenemos la desviación estándar, donde en la mitad de los casos tenemos resultados similares, (cercanos a 0.38, lo tres casos para las lecturas del acelerómetro, exceptuando el eje Z que es igual a 0.30, indicando una baja variabilidad), y en la otra mitad tenemos una variación más considerable entre las mismas variaciones (para las lecturas del giroscopio, donde contamos con un valor tan grande como 0.56 y uno tan pequeño como 0.30). De aquí discernimos no solamente cierta dependencia entre los ejes del giroscopio, sino también una variabilidad rotacional muy grande en lo que respecta a su eje Y (0.56), y en cierta medida también a su eje X (0.44).

Respeto a los mínimos y máximos, observamos, en el eje X del acelerómetro, un rango de (-0.64, 1.95), pudiendo asumir que se presenta un sesgo hacia valores positivos. En cuanto al giroscopio, en el eje Y tenemos un rango de (-2.40, 3.12), indicando un movimiento rotacional muy grande en ambos ejes (debido a los valores absolutos máximos en ambos signos)

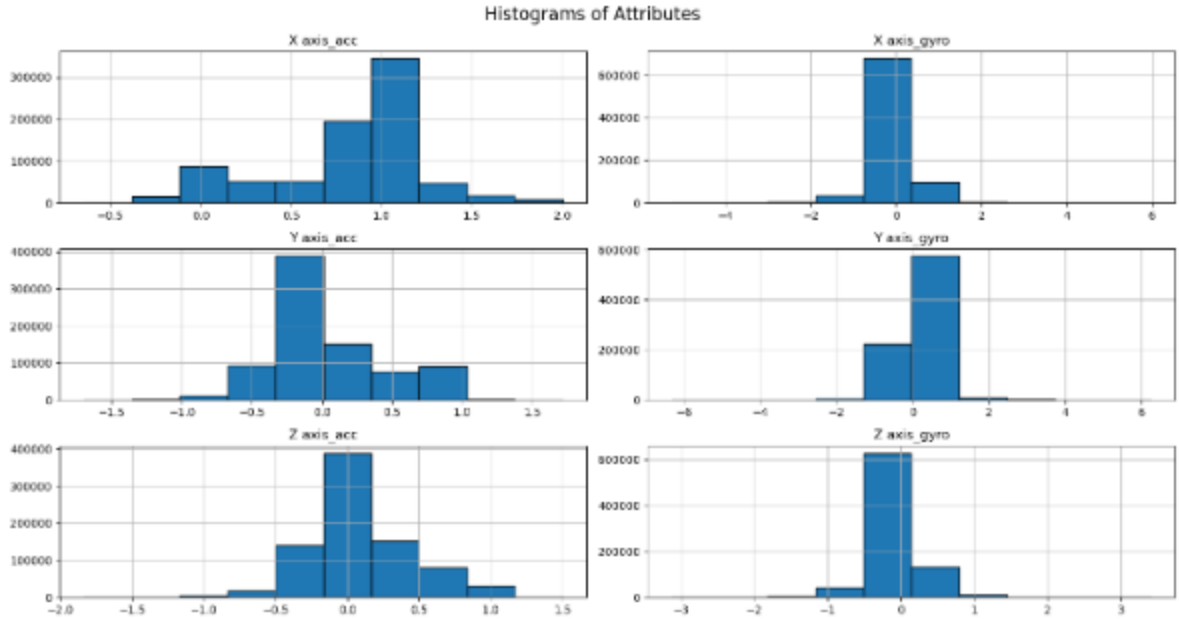
Observando la información cuartal, observamos que la primera mitad de los datos respecto a las lecturas del acelerómetro son muy cercanas a su media, por lo que esto indica una baja variabilidad y una distribución aproximadamente normal. Por el lado del giroscopio, debido a que las medianas son muy cercanas a 0, esto nos da a entender que la mayoría de los eventos rotacionales son pequeños, con algunas excepciones en valores extremos como según indican los valores mínimos y máximos.

Para una mejor visualización de los estadísticos, se proponen los siguientes gráficos, los cuales ayudarán a identificar con más facilidad patrones en los datos y la correlación entre los mismos, lo cual contribuirá a la decisión entre permanencia o desalojo de ciertas características para evitar la redundancia y disminuir dimensionalidad [5]. [6]

3.1.3 Visualizaciones de los datos e interpretaciones

Histograma de los atributos

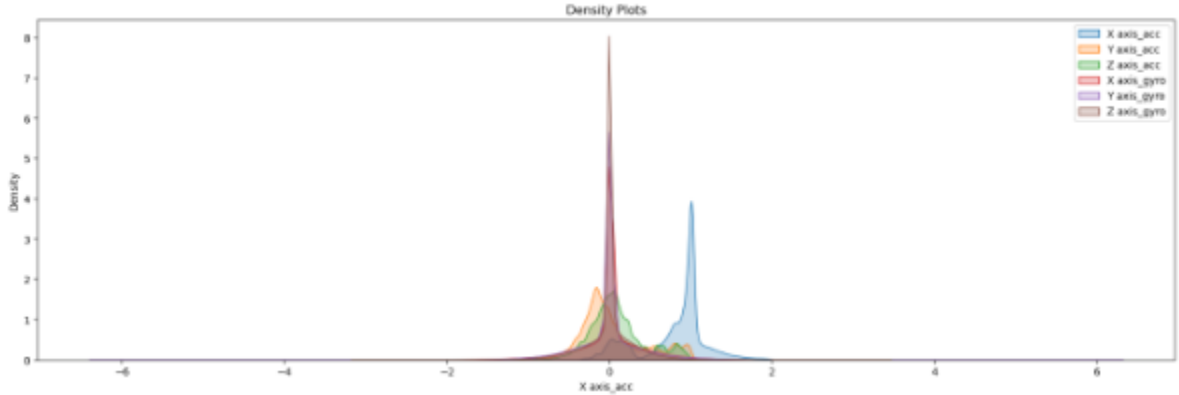
Podemos observar la distribución de los datos a manera de histogramas para cada característica (exceptuando la columna de actividad, puesto que esta información la tratamos anteriormente). Optamos, para una mejor comprensión, agrupar nuestras gráficas en dos columnas: La primera conteniendo los histogramas para las características del sensor acelerómetro, y la segunda las del sensor giroscopio:



Podemos observar una clara tendencia hacia el cero incluso sin que nuestro conjunto de datos se encuentre escalado de momento. Se observa también una dispersión mucho más significativa en las lecturas de los datos del acelerómetro a lo largo del tiempo, donde la tendencia a valores negativos es mucho más favorable que haciendo contraparte con los ejemplos del sensor giroscopio.

Gráfica de densidad para la distribución de los atributos

De manera análoga, podemos obtener una gráfica de densidad para las características, esta vez generalizando en una sola figura, puesto que lo que se pretende observar es precisamente cómo se comparan en densidad unos atributos respecto a otros:



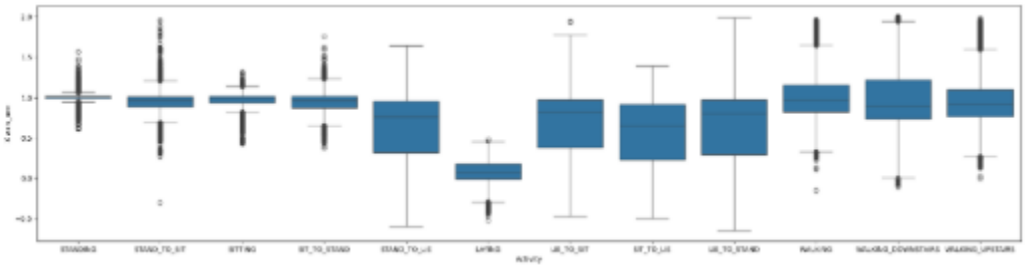
Proporcionándonos información similar al histograma de atributos anterior, podemos corroborar y confirmar que nuestro eje X del sensor acelerómetro representa una distribución de valores mucho más dispersa que los demás ejes. En cuanto a densidad se refiere, se observa que son

considerablemente variables, exceptuando únicamente al par formado por el eje Y y Z del acelerómetro, donde coinciden en gran medida.

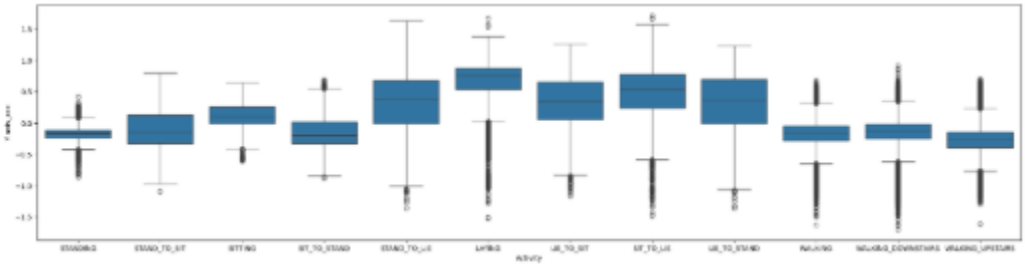
Gráfica de cajas y bigotes para la distribución de los atributos

Con ayuda de la ligera modificación que realizamos al añadir la columna de actividad, podemos apoyarnos de las gráficas de cajas y bigotes para obtener la distribución aproximada de los datos por actividad y eje de cada sensor, y así obtener información valiosa respecto a qué patrones de comportamiento similares observamos:

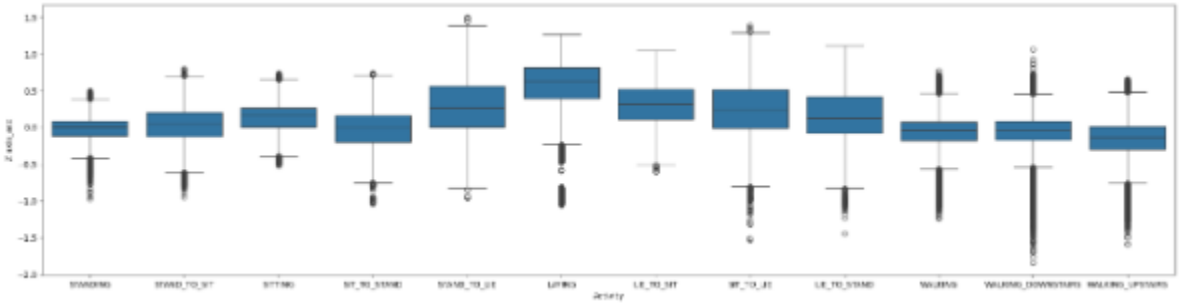
Eje X, acelerómetro:



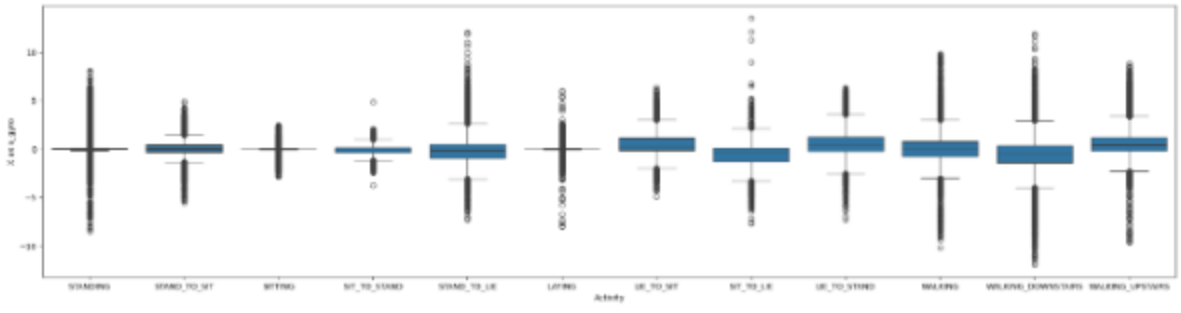
Eje Y, acelerómetro:



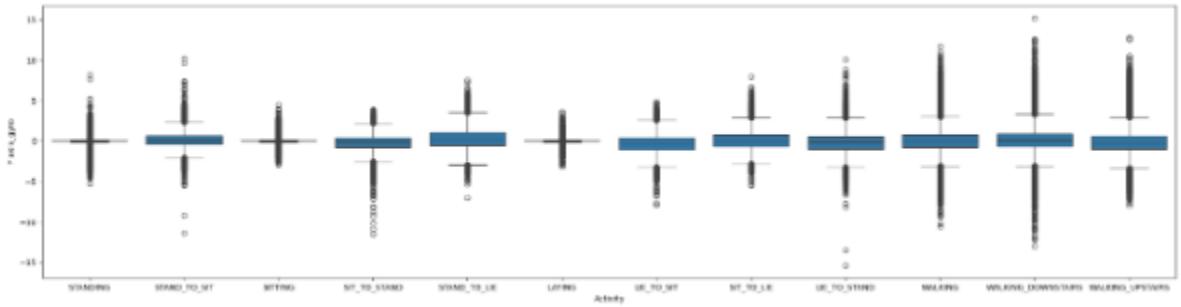
Eje Z, acelerómetro:



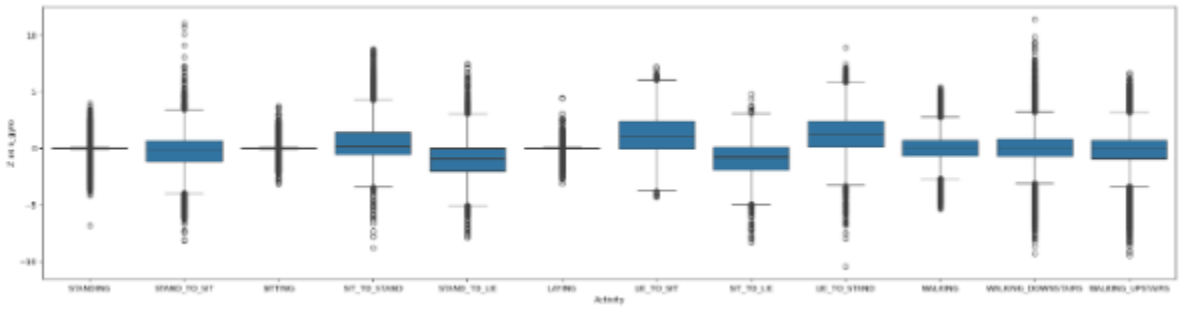
Eje X, giroscopio:



Eje Y, giroscopio:



Eje Z, giroscopio:



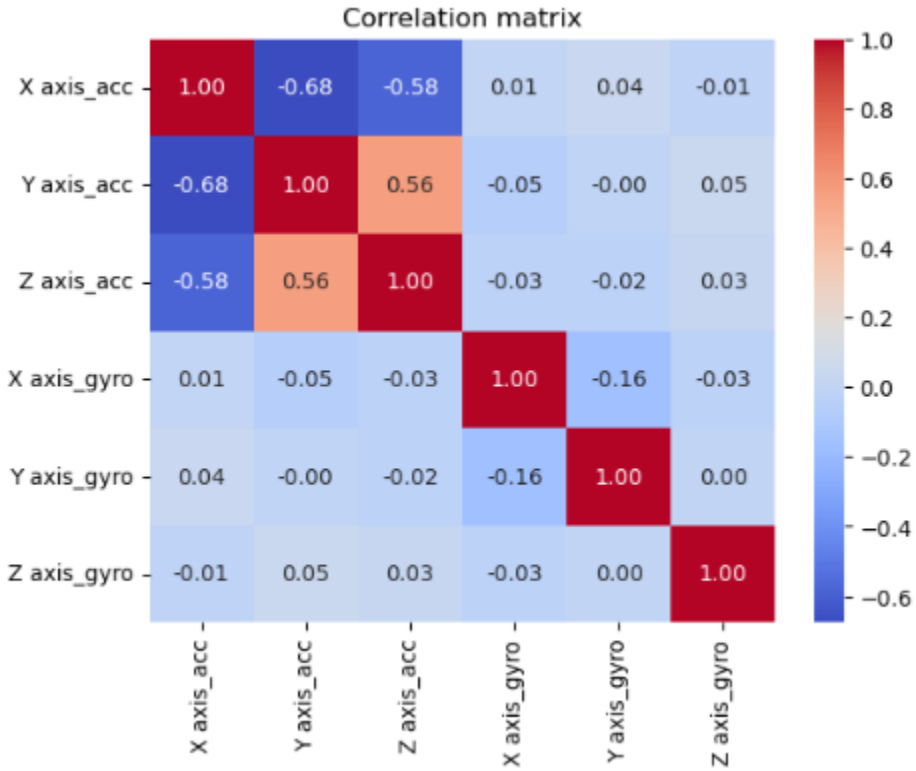
Como podemos observar en el caso de las lecturas del sensor acelerómetro, aquellas actividades con el rango inter-cuartil más largo corresponden a actividades que involucran movimientos bruscos, repentinos o repetitivos. Respecto a outliers, el eje Y y Z predominan en su presencia, particularmente en las actividades dinámicas, que involucran caminar. Lo anterior es de esperarse, puesto que, recordando la posición del sensor (Samsung Galaxy S II a nivel de la cintura), las diferentes estructuras óseas y estilos de caminata pueden hacer que algunas de estas lecturas varíen considerablemente en cuanto a desplazamiento en ambas direcciones. En general, el grado de agresividad con el que se realizan las actividades varía mucho de individuo a individuo si bien tendiendo a ciertas cantidades, por lo que los rangos inter-cuartiles de todas las actividades son considerablemente más grandes que aquellos presentes en el sensor giroscopio.

Pasando al sensor giroscopio, como es de esperarse, aquellas actividades que son estáticas poseen valores prácticamente nulos para los tres ejes (no se presenta ningún tipo de rotación). En realidad,

los valores para la gran mayoría de actividades tienden a valores nulos debido a la poca presencia de rotación, inclusive para las actividades dinámicas y cambios posturales, aunque con rangos inter-cuartiles ligeramente más grandes por las mismas razones que las antes descritas en el caso del acelerómetro.

Matriz de correlación de los atributos

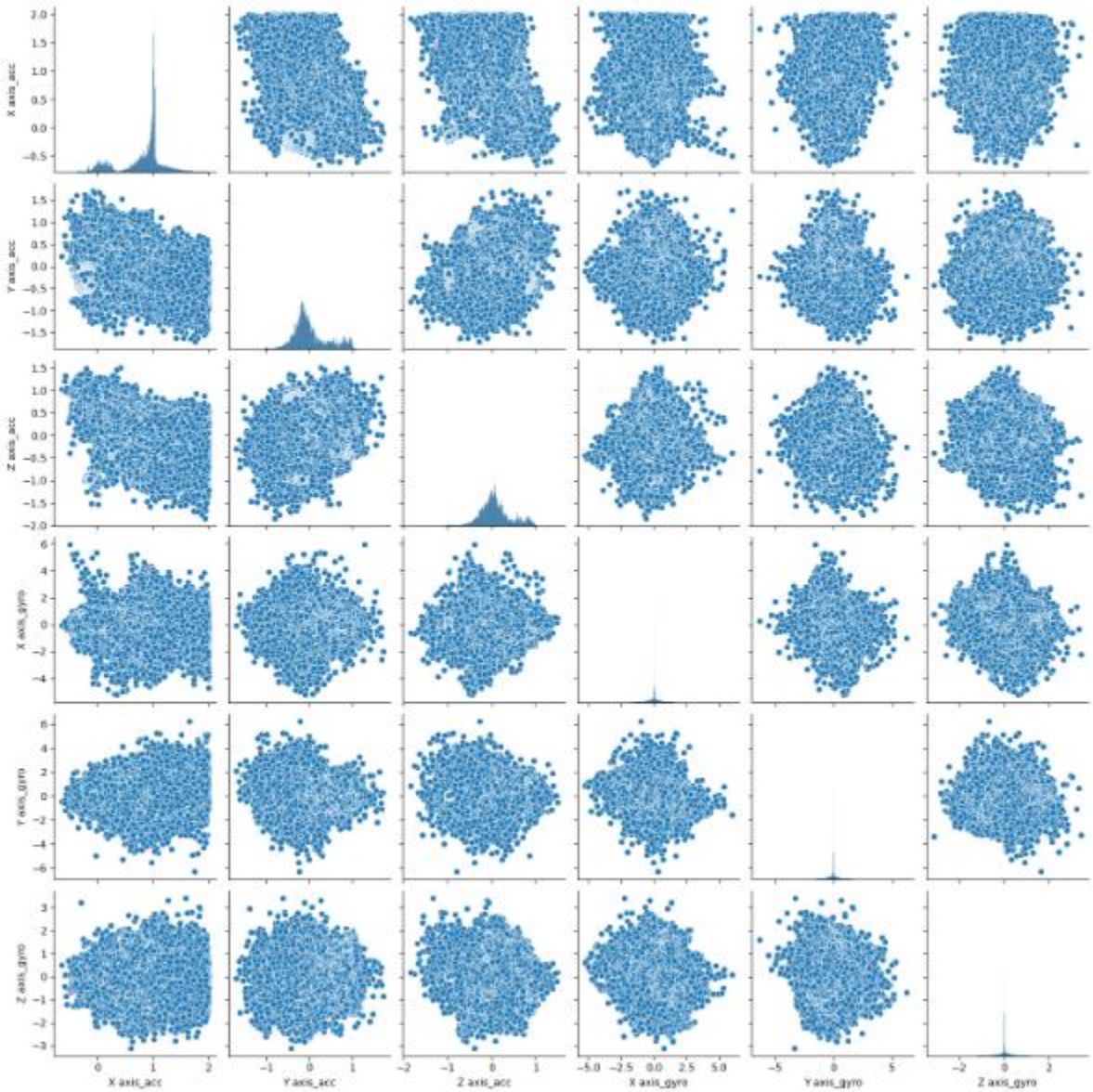
Apoyándonos de la información obtenida de la correlación entre columnas, podemos graficar nuestra información resultante a manera de mapa de calor para contar con una representación mucho más clara:



Con valor cercanos a –1, parece haber una correlación altamente negativa entre los ejes X e Y (-0.63), y X y Z (-0.76) del acelerómetro, donde podemos inferir que a medida que un valor aumenta, el otro disminuye y viceversa. Se observa también una correlación positiva considerable entre los ejes Y y Z del acelerómetro (0.72), donde a medida que uno aumenta, el otro aumenta también. En lo que respecta a información entre sensores, al comparar los pares de datos disponibles y observar coeficientes de correlación muy cercanos a 0, podemos concluir con cierto grado de confianza que no existe una correlación entre ambos sensores: esto es, la información del acelerómetro es independiente de la del giroscopio y no presentan valores predictibles entre ellos. Estos datos cercanos a 0 serán, ultimadamente, aquellos más relevantes para entrenar al clasificador.

Matriz de dispersión de los atributos

Para cada atributo respecto a todos los demás atributos, podemos obtener una matriz de dispersión que nos guía a través de la posible correlación entre estos atributos de una manera más visual. Al contar con 6 características distintas, tendremos un total de 36 matrices de dispersión para cada atributo, independientemente de las actividades:



A grandes rasgos podemos observar una dispersión considerable y aparentemente irregular respecto a todos los pares de datos, con un par de excepciones entre los ejes del acelerómetro X e Y (fila 2, columna 1), ejes X y Z (fila 3, columna 1) y los ejes Y y Z (fila 3, columna 2). En los primeros dos casos parece haber de manera clara una correlación negativa, donde a medida que aumenta la aceleración sobre el eje X, disminuye sobre el eje Y y Z, y en el tercer caso una correlación aparentemente positiva donde, a medida que aumenta la aceleración sobre el eje Y,

aumenta sobre el eje Z. Lo anterior coincide con la información provista por la matriz de correlación obtenida.

3.1.4 Escalamiento

Debido a la importancia del proceso de escalado de los datos para garantizar una contribución balanceada de las características hacia nuestro clasificador y otros algoritmos [7], ya habiendo observado y analizado el comportamiento de nuestros datos y los rangos de valores que abarcan, podemos identificar el tipo de escalado que más se adapte a nuestro caso. Debido a que contamos con datos de sensores distintos, sin valores restrictivos en cuanto a mínimos y máximos, y sin "outliers" considerables discernibles, podemos utilizar la estandarización para escalar los datos respecto a la media y la desviación estándar. Con esto conseguimos trabajar con los datos desde un eje común, siempre manteniendo la agrupación respecto a actividades. Este escalamiento será aplicado según sea necesario mediante pipelines propias para cada clasificador o algoritmo, con el fin de garantizar la modularidad de estos procesos.

3.1.5 Segmentación en ventanas y extracción de características

Debido a que ha sido demostrado que una ventana de 3s sin superposición es tiempo suficiente para el reconocimiento de la actividad humana de acuerdo con estudios previos [8], optamos por seccionar nuestro conjunto de datos para dar paso a la extracción de características. Dado que la frecuencia de muestreo para este dataset es de 50hz, tenemos que 50 ejemplos son capturados cada segundo, para cada uno de los 3 ejes presentes en cada uno de los sensores. Con esta información podemos obtener el número de muestras en cada ventana (50 x 3 = 150 muestras por ventana de tiempo).

Así mismo, como resultado de experimentos previos realizados donde se pone a prueba la efectividad de distintas características estadísticas para efectos de descripción de los datos [9], se decidió por extraer 16 características propias para cada ventana de tiempo de cada sensor, siendo estas las siguientes:

- Valor medio para cada uno de los 3 ejes
- Desviación estándar para cada uno de los 3 ejes
- Valor máximo para cada uno de los 3 ejes
- Correlación entre cada par de ejes (x – y, x – z y y – z, tanto para el acelerómetro como el giroscopio)
- Valor medio de la magnitud
- Desviación estándar de la magnitud
- Magnitud del área bajo la curva (AUC, fig. 1)
- Diferencia de medias de magnitud entre lecturas consecutivas (fig 2)

, donde la magnitud de la señal representa la contribución en general de la aceleración de los 3 ejes (fig 3)

$$AUC = \sum_{t=1}^{T} magnitude(t)$$

(fig 1)

$$meandif = \frac{1}{T-1} \sum_{t=2}^{T} magnitude(t) - magnitude(t-1)$$

(fig 2)

$$Magnitude(x, y, z, t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2},$$

(fig 3)

```
[5 rows x 33 columns]
Index(['X axis_acc_mean', 'Y axis_acc_mean', 'Z axis_acc_mean',
       'X axis_gyro_mean', 'Y axis_gyro_mean', 'Z axis_gyro_mean',
       'X axis_acc_std', 'Y axis_acc_std', 'Z axis_acc_std', 'X axis_gyro_std',
       'Y axis_gyro_std', 'Z axis_gyro_std', 'X axis_acc_max',
       'Y axis_acc_max', 'Z axis_acc_max', 'X axis_gyro_max',
       'Y axis_gyro_max', 'Z axis_gyro_max', 'acc_xy_corr', 'acc_xz_corr',
       'acc_yz_corr', 'gyro_xy_corr', 'gyro_xz_corr', 'gyro_yz_corr',
       'acc_magnitude_mean', 'gyro_magnitude_mean', 'acc_magnitude_std',
       'gyro_magnitude_std', 'acc_magnitude_auc', 'gyro_magnitude_auc',
       'acc_magnitude_meandif', 'gyro_magnitude_meandif', 'Activity'],
      dtype='object')
```

Aquí se aprecia la totalidad de las 33 columnas, donde 16 de estas son propias de las características del sensor acelerómetro, 16 propias de las características del sensor giroscopio, y una columna más de Actividad que representa la actividad real (columna objetivo -> etiqueta y), según la información proporcionada en la literatura, que corresponde a cada lectura (true labels).

Consiguiendo entonces una estructura similar a la siguiente:

	X axis_acc_mean	Y axis_acc_mean	Z axis_acc_mean	X axis_gyro_mean	Y axis_gyro_mean	Z axis_gyro_mean	X axis_acc_std	Y axis_acc_std	Z axis_acc_std	X axis_gyro_std	...
0	1.019278	-0.124065	0.098463	0.011126	-0.003462	0.002171	0.002599	0.004167	0.006087	0.014279	...
1	1.020130	-0.127602	0.090815	0.001517	-0.005042	0.005180	0.003144	0.007112	0.007858	0.014730	...
2	1.020306	-0.126500	0.085787	0.001004	0.000379	-0.000387	0.002739	0.004956	0.005328	0.014036	...
3	1.020574	-0.135778	0.078583	0.003728	-0.003065	0.005626	0.002894	0.007167	0.007856	0.011213	...
4	1.021120	-0.133620	0.074981	0.007927	-0.000324	0.001303	0.003670	0.007297	0.006793	0.014978	...
...	...	...	...	...	...	...	...	...	...	...	...
5432	1.020444	-0.111389	0.073991	0.135337	0.063542	0.044491	0.347427	0.193255	0.177095	0.660813	...
5433	1.009528	-0.219583	0.024889	0.025178	-0.049000	0.033763	0.260438	0.152240	0.187390	0.315084	...
5434	0.982389	-0.195648	-0.046481	0.574843	-0.164661	-0.092935	0.262729	0.176079	0.187472	0.678318	...
5435	1.001694	-0.212491	-0.058093	0.441704	-0.132680	-0.025557	0.261502	0.172279	0.203355	0.532342	...
5436	0.971824	-0.222407	-0.020787	0.024382	0.124089	0.046526	0.257563	0.146994	0.215006	0.402951	...

5437 rows × 33 columns

Se aprecian las características definidas y el total de ventanas obtenidas (número de lecturas // tamaño de ventana = 815,614 // 150 = 5437 ventanas)

3.1.6 Conjuntos de entrenamiento y de prueba

Según la literatura del libro "Introduction to Machine Learning with Python", la segmentación por defecto del 70% para el conjunto de entrenamiento y 30% para el conjunto de pruebas resulta ideal para un conjunto de datos de tamaño razonable. Debido a que contamos con más de 800,000 lecturas inicialmente y un total de unas 5,000 ventanas al realizar las transformaciones correspondientes, seguir con este modus operandi podría parecer una idea razonable. Sin embargo, debido al significativo desbalance en algunas clases, resulta esencial el representar a la mayoría de los datos posible dentro de nuestro conjunto de entrenamiento para no limitarnos aún más. Un buen punto de partida, para nuestro conjunto de datos, es de 80% de los datos dirigidos al conjunto de entrenamiento y 20% de los datos dirigidos al conjunto de pruebas.

Contamos entonces con dos conjuntos. Primeramente, el conjunto de entrenamiento, con un total de 4349 ventanas de muestras (80% del dataset), y nuestro conjunto de pruebas, con un total de 1088 ventanas de muestras (20% del dataset).

Recordamos, sin embargo, el desbalance que hubo inicialmente en las clases. Una manera de solucionarlo y no tener falsos positivos que podrían afectar a nuestra precisión es mediante el SMOTE [13], correspondiente a una técnica de oversampling para crear puntos sintéticos de las clases minoritarias. Dentro de las variantes disponibles del SMOTE, el Borderline-SMOTE resulta ser de las que traen mejores resultados en conjuntos de datos altamente desbalanceados de acuerdo con estudios previos [14]. Al aplicar esta técnica, obtendremos consigo un conjunto de datos de 8,820 ventanas, sobre las cuales conseguiríamos, siguiendo la misma distribución, 7,056 para el conjunto de entrenamiento y 1,764 para el conjunto de pruebas. Es importante recalcar que el resampling NO se aplicará inmediatamente posterior a la división de los conjuntos, si no que será agregado a los pipelines propios de cada clasificador para mantener reproducibilidad e independencia de los procesos. Dicho lo anterior, se menciona la estructura que tendría nuestro conjunto de datos para efectos explicativos.

3.1.7 Entrenamiento de clasificadores

Procedemos ahora a entrenar a nuestros modelos y calcular la precisión para el conjunto de entrenamiento y el de pruebas. En el presente documento, se estará trabajando y comparando el desempeño de tres clasificadores: regresión logística, bosques aleatorios y perceptrón. El proceso análogo a cada uno es descrito a continuación:

1. Crear pipeline para cada clasificador asegurando reproductibilidad y modularidad [15].
2. Agregar al pipeline un tipo de escalado. Debido a la sensibilidad de los clasificadores y algoritmos a los diferentes rangos, se desea obtener unidad de varianza. Procedemos a utilizar la estandarización. Esto contribuye directamente a la fiabilidad del resampling por

parte de SMOTE y a aquellos clasificadores que necesitan el escalado (caso que no aplica para los bosques aleatorios)

3. Agregar al pipeline la técnica de oversampling preferida. Para los tres clasificadores, se optó por Borderline-SMOTE
4. Agregar al pipeline el clasificador a entrenar.
5. Entrenar los pipelines con los conjuntos de entrenamiento
6. Realizar las predicciones de cada pipeline con los conjuntos de pruebas
7. Evaluar resultados obteniendo métricas de desempeño

3.1.8 Desempeño de clasificadores

Para poder observar un desempeño más generalizado de los clasificadores, pasamos más allá de utilizar un simple split de conjuntos de datos de entrenamiento y de prueba y procedemos a obtener métricas de desempeño de nuestros modelos con el uso de la validación cruzada, la que permite seccionar esos conjuntos de datos dinámicamente, reduciendo así el sesgo y esclareciendo la viabilidad del modelo como un clasificador bien generalizado [16] Se emplea entonces la validación cruzada "k-fold", la cual resulta ser la más común y tiene como fin que el experimentador modifique el valor "k", representando la cantidad de "folds" o partes en las que dividiremos nuestro dataset, y algunas de ellas siendo elegidas sistemáticamente durante varias iteraciones como conjuntos de entrenamiento y de pruebas. La literatura nos menciona que una cantidad k de folds común es de 5 o 10 [2]. Se optará pues por elegir unos valores de k de 5 y 10. A medida que nuestro valor de k aumenta, los conjuntos de datos son cada vez más pequeños y se presenta mayor variabilidad, por lo que se espera tener resultados más generalizados, si bien consumiendo más recursos computacionales.

Primeramente, observaremos las precisiones obtenidas por cada clasificador sin utilizar alguna técnica de oversampling, para comparar cómo esto nos afecta. Tenemos entonces:

```
Logistic Regression

Training set score: 0.9048057024603358
Test set score: 0.8823529411764706
Cross validation score with 5 folds: [0.88694853 0.81341912 0.83072677 0.88040478 0.89328427]
Cross validation score with 10 folds: [0.91360294 0.86580882 0.75       0.86213235 0.77757353 0.90441176
 0.85110294 0.90791897 0.9281768  0.86003683]
Cross val score 5 folds mean: 0.8609566940851778
Cross val score 10 folds mean: 0.862076494962626

Random Forests
Training set score: 1.0
Test set score: 0.9108455882352942
Cross validation score with 5 folds: [0.88786765 0.86121324 0.84636615 0.84728611 0.88776449]
Cross validation score with 10 folds: [0.91727941 0.88602941 0.82904412 0.89154412 0.81433824 0.89154412
 0.85477941 0.87108656 0.92081031 0.86372007]
Cross val score 5 folds mean: 0.8660995251366416
Cross val score 10 folds mean: 0.8740175766439172

Perceptron
Training set score: 0.848700850770292
Test set score: 0.8373161764705882
Cross validation score with 5 folds: [0.83180147 0.796875   0.80772769 0.82704692 0.87212511]
Cross validation score with 10 folds: [0.875      0.84558824 0.75551471 0.79227941 0.81433824 0.83455882
 0.79779412 0.86740331 0.87108656 0.78084715]
Cross val score 5 folds mean: 0.827115238919855
Cross val score 10 folds mean: 0.827115238919855
```

Y realizando la comparación con el entrenamiento de los mismos clasificadores, pero esta vez utilizando la técnica de oversampling de "BorderlineSMOTE" en cada pipeline, tenemos:

Para regresión logística:

```
Cross validation score with 5 folds: [0.88051471 0.8125     0.83164673 0.86384545 0.87948482]
Cross validation score with 10 folds: [0.90625    0.84007353 0.73529412 0.85845588 0.77941176 0.89705882
 0.86029412 0.86556169 0.91160221 0.85635359]
Cross val score 5 folds mean: 0.8535983413604631
Cross val score 10 folds mean: 0.8510355730690067
```

Para bosques aleatorios:

```
Cross validation score with 5 folds: [0.87867647 0.84007353 0.84820607 0.84636615 0.89052438]
Cross validation score with 10 folds: [0.91360294 0.87132353 0.82169118 0.88419118 0.80147059 0.91544118
 0.86213235 0.86556169 0.92633517 0.87845304]
Cross val score 5 folds mean: 0.860769319227231
Cross val score 10 folds mean: 0.874020284909544
```

Para perceptrón:

```
Cross validation score with 5 folds: [0.83180147 0.796875   0.80772769 0.82704692 0.87212511]
Cross validation score with 10 folds: [0.875      0.84558824 0.75551471 0.79227941 0.81433824 0.83455882
 0.79779412 0.86740331 0.87108656 0.78084715]
Cross val score 5 folds mean: 0.827115238919855
Cross val score 10 folds mean: 0.827115238919855
```

Observando los resultados, y tomando en cuenta principalmente el puntaje obtenido con la validación cruzada de 10 folds para una mejor representación de nuestro conjunto de datos, podemos apreciar un cambio muy ligero entre los clasificadores de bosques aleatorios y perceptrón, pero una diferencia considerable en el caso de la regresión logística, disminuyendo en más de un 1% la precisión. Esto no quiere decir que nuestros clasificadores sean peores, sino que representan con mayor exactitud al conjunto de datos provisto, disminuyendo falsos positivos. El si el clasificador resulta tener un buen desempeño o no en general, está provisto por las métricas de desempeño que veremos a continuación.

Métricas de desempeño

Se emplearán 4 métricas de desempeño para evaluar los resultados obtenidos: exactitud (accuracy), precisión (precision), puntaje F-1 (F-1 Score) y sensibilidad (Recall). Todas las anteriores son propiedades que nos sirven para poder resumir la matriz de confusión de los datos que nos proporciona sklearn para observar el comportamiento de nuestras predicciones respecto a los valores reales.

Accuracy

La exactitud es la forma más sencilla de resumir los resultados y consiste en una razón entre el número de predicciones correctas (dadas por TP y TN) dividida entre el número total de muestras (que representa a la sumatoria de todas las entradas de nuestra matriz de confusión). Tenemos entonces:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Donde:

TP (True Positive): Representa la cantidad de casos positivos correctamente clasificados,

TN (True Negative): Representa la cantidad de casos negativos correctamente clasificados,

FP (False Positive): Representa la cantidad de casos negativos clasificados como positivos (erróneamente clasificados: falsos positivos)

FN (False Negative): Representa la cantidad de casos positivos clasificados como negativos (erróneamente clasificados: falsos negativos)

Precision

La métrica de precisión representa una razón entre la cantidad de casos positivos correctamente clasificados entre la cantidad de casos clasificados como positivos, ya sea correcta o incorrectamente; esto es, la precisión nos brinda información acerca de cuántas muestras predichas como positivas en realidad lo son:

$$\text{Precision} = \frac{TP}{TP+FP}$$

Se sigue el mismo glosario de términos que en el caso de la exactitud.

Esta métrica es especialmente útil cuando los falsos positivos cuentan con mucho peso y deseamos limitarlos lo más posible. Es también conocida como PPV (Positive Predictive Value: Valor predictor positivo)

Recall

Similar a la precisión, el recall/sensibilidad/tasa de aciertos/TPR (True Positive Rate: Razón de verdaderos positivos) mide cuántos de los son capturadas por predicciones positivas, a manera de razón entre la cantidad de muestras positivas clasificadas correctamente entre la misma cantidad de muestras positivas clasificadas correctamente y la cantidad de muestras negativas clasificadas

incorrectamente (siendo estas, en realidad, positivas). Es comúnmente empleada cuando se desea eliminar los falsos negativos

$$\text{Recall} = \frac{TP}{TP+FN}$$

F-1 Score

Debido a la contraposición entre recall y precision, donde optimizar una podría conllevar a la reducción de la otra, una manera de resumir los resultados de ambas métricas, las cuales resultan ser las dos más utilizadas en la comunidad de aprendizaje máquina, es el F-1 Score, o también F-1 Measure.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

La cual tiene como fórmula derivada la media armónica de ambas métricas, recall y precision.

Se obtienen, entonces, sin más modificaciones, los siguientes resultados:

Para aquellos pipelines sin SMOTE:

```
Logistic Regression Metrics
Accuracy:  0.8786764705882353
Precision: 0.8780412608416714
Recall:    0.8786764705882353
F1 Score:  0.8779361235916237

Random Forest Metrics
Accuracy:  0.9071691176470589
Precision: 0.9079487262014536
Recall:    0.9071691176470589
F1 Score:  0.9061236676476062

Perceptron Metrics
Accuracy:  0.8235294117647058
Precision: 0.8406218095265997
Recall:    0.8235294117647058
F1 Score:  0.8169166797096838
```

Para aquellos pipelines con SMOTE:

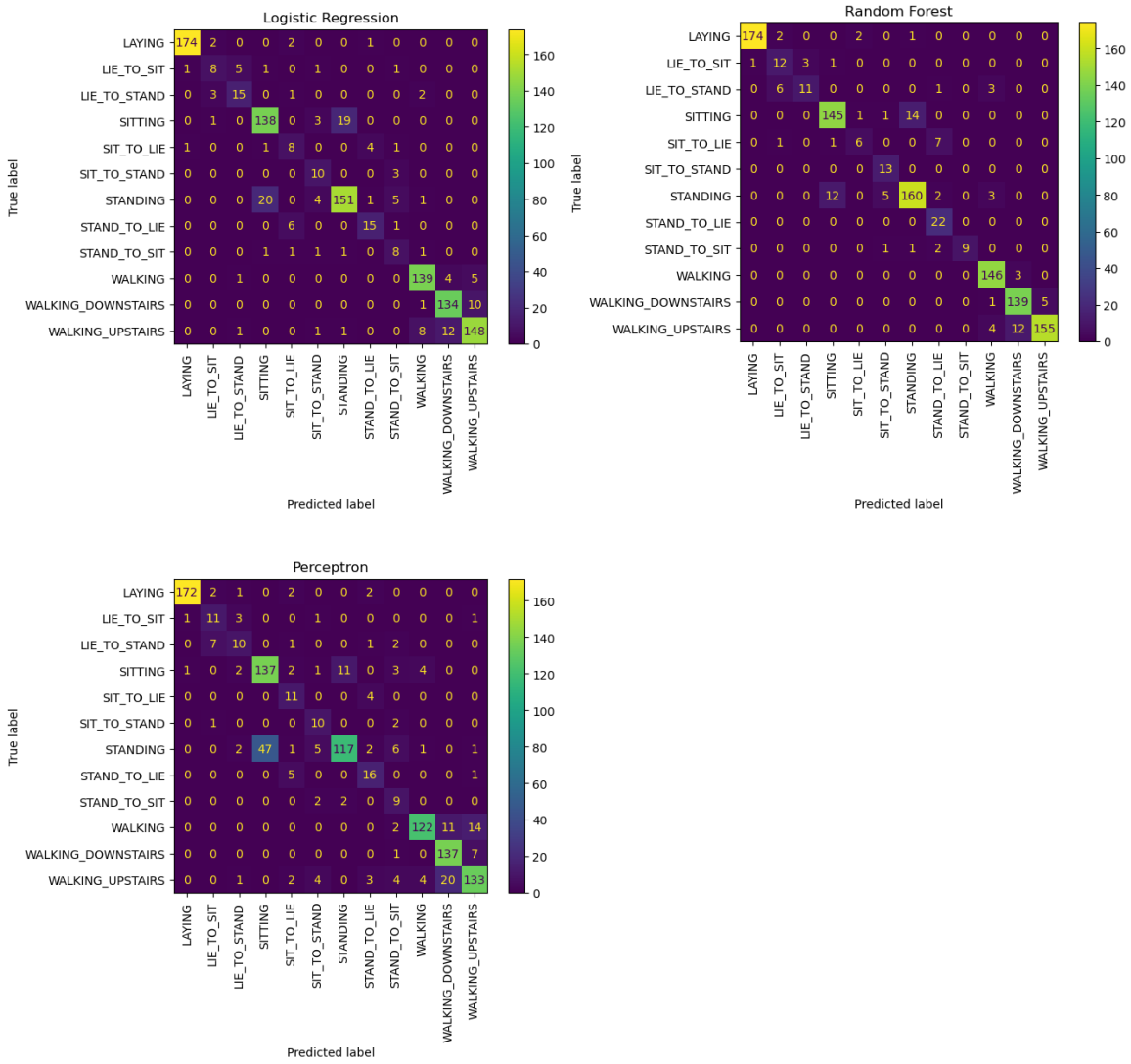
```
Logistic Regression Metrics
Accuracy:  0.8713235294117647
Precision: 0.877009006382335
Recall:    0.8713235294117647
F1 Score:  0.873207729446176

Random Forest Metrics
Accuracy:  0.9117647058823529
Precision: 0.9165887412182478
Recall:    0.9117647058823529
F1 Score:  0.9111262662315779

Perceptron Metrics
Accuracy:  0.8134191176470589
Precision: 0.8378449324187249
Recall:    0.8134191176470589
F1 Score:  0.8179529554849596
```

En el caso de la regresión logística, podemos discernir un leve descenso de puntajes en las cuatro métricas, con lo que, para este clasificador, se interpreta que la adición de datos sintéticos no ayuda tanto como los datos originales.

En cuanto a los bosques aleatorios, se obtuvo una mejora en todas las métricas, leve en todas excepto en la de precisión, donde se presentó una mejora significativa. Dada la mayor complejidad de la base de datos, junto con los datos sintéticos añadidos, se llega a la conclusión de que nuestro clasificador de bosques aleatorios aprendió mejor de los datos balanceados, particularmente en términos de la precisión anteriormente mencionada y el score F-1.

Por último, el clasificador Perceptrón presentó disminuciones leves en todas las métricas exceptuando el score F-1, donde se observa una leve mejora. Se llega a la conclusión de que este clasificador no se benefició claramente de la técnica SMOTE, posiblemente por la sensibilidad del clasificador al ruido.

Y para observar mejor el cómo estos modelos se comportan a nivel de clasificación (qué tan acertados están), podemos optar por una visualización a manera de matrices de confusión para cada modelo, resultándonos lo siguiente:



Donde se observa con claridad que el mejor desempeño dentro de los 3 clasificadores es aquel provisto por el modelo de bosques aleatorios. Esto dentro del contexto en el cual NO hemos realizado ningún cambio en los parámetros de los modelos, más allá de aumentar el número máximo de iteraciones de la regresión logística con el fin de mejorar la convergencia.

Sin embargo, ya habiendo observado lo anterior, podemos ahora realizar la comparación de los modelos al nutrirlos a un algortimo GridSearchCV, que se encargará de realizar todas las combinaciones posibles de los parámetros que nosotros le indiquemos con el fin de obtener los parámetros que mejor desempeño nos provean en promedio, a la par que hará uso de la validación cruzada para obtener resultados más fiables [17] Contamos con los reportes siguientes por cada clasificador, una vez habiendo realizado la búsqueda de los mejores parámetros con SMOTE, obtenemos:

Regresión Logística (C: 10, Max_iter: 1000, Penalty: L1, Solver: Saga):

```
Best cross-validated score: 0.8802

Classification Report for Logistic Regression:

                    precision    recall  f1-score   support

           LAYING       0.99      0.97      0.98       179
       LIE_TO_SIT       0.56      0.53      0.55        17
     LIE_TO_STAND       0.67      0.67      0.67        21
          SITTING       0.84      0.85      0.85       161
       SIT_TO_LIE       0.42      0.53      0.47        15
     SIT_TO_STAND       0.50      0.69      0.58        13
         STANDING       0.87      0.83      0.85       182
     STAND_TO_LIE       0.74      0.64      0.68        22
     STAND_TO_SIT       0.40      0.46      0.43        13
          WALKING       0.89      0.93      0.91       149
WALKING_DOWNSTAIRS       0.87      0.91      0.89       145
  WALKING_UPSTAIRS       0.90      0.85      0.88       171

         accuracy                           0.86      1088
        macro avg       0.72      0.74      0.73      1088
     weighted avg       0.87      0.86      0.86      1088
```

Bosques aleatorios (N_Estimators: 350):

```
Best cross-validated score: 0.9094

Classification Report for Random Forest:

                    precision    recall  f1-score   support

           LAYING       0.99      0.97      0.98       179
       LIE_TO_SIT       0.58      0.65      0.61        17
     LIE_TO_STAND       0.86      0.57      0.69        21
          SITTING       0.91      0.90      0.90       161
       SIT_TO_LIE       0.67      0.40      0.50        15
     SIT_TO_STAND       0.62      1.00      0.76        13
         STANDING       0.91      0.87      0.89       182
     STAND_TO_LIE       0.65      1.00      0.79        22
     STAND_TO_SIT       1.00      0.85      0.92        13
          WALKING       0.94      0.98      0.96       149
WALKING_DOWNSTAIRS       0.91      0.95      0.93       145
  WALKING_UPSTAIRS       0.96      0.92      0.94       171

         accuracy                           0.91      1088
        macro avg       0.83      0.84      0.82      1088
     weighted avg       0.92      0.91      0.91      1088
```

Perceptrón (Alpha: 1x10^-3, Max_iter: 500, Penalty: L1)

```
Best cross-validated score: 0.8420

Classification Report for Perceptron:

                    precision    recall  f1-score   support

           LAYING       0.98      0.97      0.97       179
       LIE_TO_SIT       0.61      0.65      0.63        17
     LIE_TO_STAND       0.47      0.67      0.55        21
          SITTING       0.89      0.76      0.82       161
       SIT_TO_LIE       0.39      0.60      0.47        15
     SIT_TO_STAND       0.29      0.77      0.43        13
         STANDING       0.84      0.81      0.82       182
     STAND_TO_LIE       0.61      0.64      0.62        22
     STAND_TO_SIT       0.14      0.08      0.10        13
          WALKING       0.95      0.67      0.79       149
WALKING_DOWNSTAIRS       0.80      0.97      0.87       145
  WALKING_UPSTAIRS       0.77      0.82      0.80       171

         accuracy                           0.81      1088
        macro avg       0.65      0.70      0.66      1088
     weighted avg       0.83      0.81      0.81      1088
```

Obteniendo así, después de construir pipelines con escalado, SMOTE, modificación de parámetros de los clasificadores y validación cruzada con 10 folds, obtenemos los mejores puntajes de validación cruzada con 10 folds de:

Regresión Logística: 0.8802

Bosques Aleatorios: 0.9094

Perceptrón: 0.8420

3.1.9 Reducción de características

Dado que nuestra base de datos estaba limitada en gran medida por el desbalance de clases, donde tuvimos que optar por puntos sintéticos de datos para las clases minoritarias, y la cantidad de características iniciales no es muy alta en primer lugar, resulta una buena idea el mantener el conjunto de datos de manera intacta y preservar la mayor cantidad de información posible [18]. Sin embargo, para otras bases de datos, contamos con técnicas de reducción de características propias para reducir nuestra dimensionalidad de manera significativa, manteniendo la mayoría de la varianza entre los datos. Dentro de las técnicas disponibles de reducción de dimensionalidad mediante reducción/selección de características, el análisis de componente principal (PCA) resulta ser la técnica más empleada y probada [19]. Se recomienda, de igual manera mantener un nivel de varianza del 95% para no perder demasiada información valiosa. Para cuestiones de comparación y fines educativos, aplicaremos técnicas de reducción de dimensionalidad de PCA y PCA de Kernel, junto con algunas técnicas de selección de características de acuerdo con el modelo (RFECV para regresión logística y perceptrón, demostrando ser El RFE (Recursive Feature Elimination - Eliminación recursiva de características), particularmente eficaz con este tipo de modelos lineales, incluso con variaciones de tamaño dentro del conjunto de datos [23], y SelectFromModel para bosques aleatorio, habiéndose demostrado con anterioridad su eficiencia y

eficacia al seleccionar las variables más relevantes de acuedo con sus coeficientes en un modelo no lineal como los bosques aleatorios [24]). Se opta entonces por crear pipelines para cada clasificador, compuestos por:

1. Estandarización
2. SMOTE
3. Técnica de reducción de dimensionalidad/selección de características
4. Clasificador

Para observar cambios respecto a los resultados obtenidos previamente sin reducción de dimensionalidad. Obtenemos consigo los resultados siguientes, utilizando validación cruzada con 5 y 10 folds:

Regresión logística + PCA: 5-fold mean: 0.8085, 10-fold mean: 0.8121

Regresión logística + PCA de Kernel: 5-fold mean: 0.8085, 10-fold mean: 0.8121

Regresión logístia + RFECV: 5-fold mean: 0.8553, 10-fold mean: 0.8516

Bosques aleatorios + PCA: 5-fold mean: 0.8264, 10-fold mean: 0.8334

Bosques aleatorios + PCA de Kernel: 5-fold mean: 0.8190, 10-fold mean: 0.8339

Bosques aleatorios + SelectFromModel: 5-fold mean: 0.8558, 10-fold mean: 0.8573

Perceptrón + PCA: 5-fold mean: 0.7195, 10-fold mean: 0.6866

Perceptrón + PCA de Kernel: 5-fold mean: 0.7195, 10-fold mean: 0.6866

Perceptrón + RFECV: 5-fold mean: 0.8227, 10-fold mean: 0.7981

Con los resultados anteriores, podemos ver que los mejores resultados son los obtenidos mediante las técnicas de selección de características propuestas, como era de suponerse de acuerdo con investigaciones previas.

3.1.10 Método estadístico para la comparación de clasificadores

Para realizar una evaluación objetiva del rendimiento de los clasificadores empleados en este trabajo, es esencial aplicar métodos estadísticos que permitan determinar las diferencias observadas en las métricas de desempeño son estadísticamente significativas.

Elección del método estadístico

Debido a que en este trabajo se comparan más de dos clasificadores sobre un mismo conjunto de datos utilizando validación cruzada, el método más adecuado es el **Test de Friedman**, seguido de un análisis **Post-hoc de Nemenyi**. Esta elección se fundamenta por las siguientes razones:

El test de Friedman es un método no paramétrico utilizado para detectar diferencias entre varios algoritmos que han sido evaluados sobre múltiples particiones de un dataset. A diferencia de otros análisis, este test no asume ninguna normalidad ni homogeneidad de varianzas, lo cual es útil cuando se trabaja con métricas de clasificación que no siguen una distribución normal o casi siempre.

De ser así, cuando el test de friedman indica la existencia de diferencias estadísticas significativas entre los clasificadores, se aplica un análisis Nemenyi. Para compara todos los clasificadores entre sí por pares e identificar cuales algoritmos presentan diferencias significativas.

Este enfoque ha sido utilizado y recomendado por Demšar (2006) sobre el uso del test de friedman en conjunto con el análisis post-hoc de Nemenyi. Y en particular, se recomienda el uso de pruebas paramétricas para comparar el rendimiento entre múltiples clasificadores de datos de HAR dado que las distribuciones de métricas como la exactitud de o la F1 no son siempre normales ni homogéneas.

Aplicado a nuestro conjunto de datos de actividades y transiciones posturales, enfocándonos en la exactitud, obtenemos:

```
Friedman statistic: 19.5385, p-value: 0.0001
```

Debido a que nuestro valor P es mucho menor que 0.05 (valor de alpha), podemos concluri que se debe rechazar la hipótesis nula a favor de la alternativa. La hipótesis nula, dentro de nuestro contexto, indica que no hay diferencias significativas entre los modelos, y por tanto, nuestra hipótesis alternativa indica que, para al menos uno de los clasificadores, el rendimiento es significativamente diferente.

Debido a lo anterior, resulta imprescindible realizar una prueba post-hoc de Nemenyi para identificar entre qué pares de modelos existen estas diferencias:

```
📌 Tabla de p-valores del test de Nemenyi:
                    Perceptron  Random Forest  Logistic Regression
Perceptron            1.000000       0.000039             0.049475
Random Forest         0.000039       1.000000             0.109180
Logistic Regression   0.049475       0.109180             1.000000
```

Lo que debemos observar aquí son aquellos valores P menores a 0.05, donde observamos una leve diferencia entre el par perceptrón y regresión logística, y una diferencia contundente y abrumadora para el perceptrón y los bosques aleatorios. Podemos llegar a la conclusión, por ende, de que los bosques aleatorios, junto con las modificaciones correspondientes en hiperparámetros, aplicando escalado y técnicas de oversampling, presenta el mejor desempeño.

3.1.11 Mecanismos de fusión de datos y desempeño

En lo que respecta a clasificadores, contamos con muchas opciones. Las opciones que hemos revisado hasta ahora han sido clasificadores propios, esto es, algoritmos con comportamientos particulares representativos de métodos específicos para obtener la clasificación deseada. Hasta ahora hemos estado trabajando con 3 de estos: regresión logística, bosques aleatorios, y perceptrón. A la par, hemos estado empleando distintas técnicas de preprocesamiento de datos para mejorar la generalización de nuestros clasificadores, y hemos visto los efectos que esto tiene en las métricas de desempeño.

Pasamos ahora a mencionar un nuevo concepto: fusión de datos. Según Vlaicu & Matei (2025): "La fusión de datos se define como el proceso de integración de múltiples fuentes de información con el fin de generar datos más coherentes, precisos y útiles que los obtenidos por separado" [25]. Tomando en cuenta lo anterior, aplicado a nuestro campo, podemos aprovechar la fusión de datos para combinar la estructura de distintos estimadores con el fin de obtener un estimador final compuesto que se desempeñe de una mejor manera que los estimadores componentes individuales. Consigo tenemos distintos métodos de fusión que nos ayudan a conseguir nuestro objetivo. Para este escrito se consideran 3 de estos métodos: Agregación con Bosques aleatorios, Votación con distintos clasificadores (regresión logística, bosques aleatorios y bayes ingenuo) y diferentes conjuntos de características y AdaBoost, evaluando el desempeño de estos mediante las 4 métricas declaradas con anterioridad (Exactitud, precisión, score F-1 y Recall) y validaciones cruzadas estratificadas con 10 folds. Se aplica también escalado de los datos y BorderlineSMOTE en cada pipeline.

Obtenemos, entonces, los siguientes resultados:

Agregación mediante bosques aleatorios:

```
Accuracy: 0.9136
Precision: 0.9175
Recall: 0.9136
F1: 0.9144
```

Votación personalizada con regresión logística, bosques aleatorios y bayes ingenuo:

```
Accuracy:  0.8930
Precision: 0.8979
Recall:    0.8930
F1 Score:  0.8940
```

Adaboost:

```
Accuracy: 0.5443
Precision: 0.6526
Recall: 0.5443
F1: 0.4821
```

Se observan diferencias significativas en el desempeño de los 3 métodos de fusión utilizados, particularmente respecto al par de agregación con bosques aleatorios y la votación por mayoría, y el Adaboost. Al utilizar la agregación mediante bosques aleatorios subconjuntos aleatorios de características y muestras bootstrap, se mejora la generalización disminuyendo el sobreajuste, particularmente útil después de realizar las modificaciones de escalado y oversampling a nuestro conjunto de datos.

En el caso de la votación por mayoría combinando los clasificadores de regresión logística, bosques aleatorios y bayes ingenuo, obtenemos un desempeño similar a la agregación única mediante bosques aleatorios, si bien son una leve disminución del mismo, se espera que esto sea debido a la inclusión de un clasificador más sencillo y menos preciso como el bayes ingenuo, y al nivel de optimización de la votación.

Por último, el muy bajo desempeño del Adaboost tiene consigo una fundamentación debido a la sensibilidad al ruido y datos mal etiquetados, junto con el escalado y BorderlineSMOTE aplicado, modificando la estructura inherente de los datos y haciendo la adaptación secuencial de Adaboost menos óptima.

3.2 Base de Datos: Daily Sports and Activities Dataset

El presente dataset recopila información obtenida a partir de sensores de movimiento durante la realización de 19 actividades diarias y deportivas. Los datos fueron registrados con cinco unidades Xsens MTx colocadas estratégicamente en el torso, brazos y piernas de ocho sujetos (cuatro hombres y cuatro mujeres) con edades comprendidas entre los 20 y 30 años.

Cada actividad fue realizada por los sujetos durante 5 minutos, sin restricciones sobre la manera de ejecución, lo que introduce variaciones en las velocidades y amplitudes de movimiento entre individuos. Las actividades fueron llevadas a cabo en distintos entornos, incluyendo el gimnasio de la Universidad de Bilkent, el edificio de Ingeniería Eléctrica y Electrónica, y una zona exterior plana dentro del campus.

Los sensores fueron calibrados para registrar datos con una frecuencia de muestreo de 25 Hz. Para facilitar el análisis, las señales de 5 minutos fueron divididas en segmentos de 5 segundos, lo que resulta en 60 segmentos de señal por actividad.

Estructura del dataset

El conjunto de datos está organizado de la siguiente manera:

• 19 actividades registradas (A1 - A19), incluyendo estar sentado, caminata, trote, uso de máquinas de ejercicio y deportes como el baloncesto.

• 8 sujetos participantes (4 hombres y 4 mujeres).

• 60 segmentos de datos por actividad y sujeto (divididos en lapsos de 5 segundos).

• 5 sensores ubicados en diferentes partes del cuerpo: torso (T), brazo derecho (RA), brazo izquierdo (LA), pierna derecha (RL) y pierna izquierda (LL).

• 9 sensores en cada unidad, midiendo: aceleración en los ejes X, Y y Z; velocidad angular (giroscopios) en X, Y y Z; y campo magnético (magnetómetros) en X, Y y Z.

3.2.1 Dimensiones

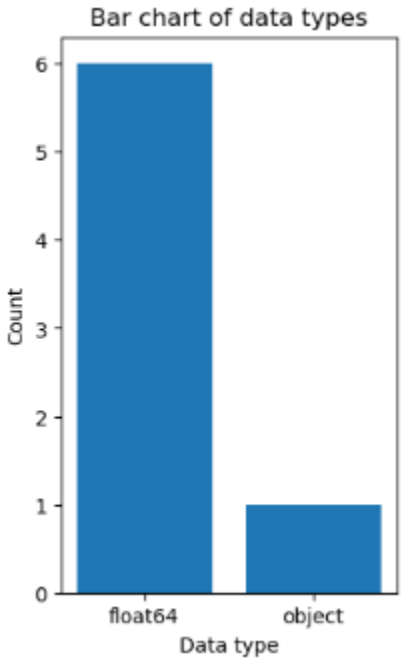
En lo que respecta a dimensiones, podemos reconocer que contamos con un total de 1,140,000 lecturas en total, 45 características (eje x, y y z del sensor acelerómetro, eje x, y y z del sensor giroscopio y eje x, y y z del sensor magnetómetro para cada una de las 5 partes del cuerpo) y 3 multi índices correspondientes a la actividad, sujeto y segundo en el que fue tomada la medición. La estructura de la base de datos queda, entonces, de la siguiente manera mostrando las primeras 5 filas;

```
Dimensiones del dataset: (1140000, 45)
Primeras 5 filas:
                                T_xacc  T_yacc  T_zacc   T_xgyro   T_ygyro  \
Activity Subject Time (s)
1        1       1           8.1305  1.0349  5.4217 -0.009461  0.001915
                 1           8.1305  1.0202  5.3843 -0.009368  0.023485
                 1           8.1604  1.0201  5.3622  0.015046  0.014330
                 1           8.1603  1.0052  5.3770  0.006892  0.018045
                 1           8.1605  1.0275  5.3473  0.008811  0.030433

                                 T_zgyro   T_xmag    T_ymag   T_zmag  RA_xacc  ...  \
Activity Subject Time (s)                                                    ...
1        1       1           -0.003424 -0.78712 -0.069654  0.15730  0.70097  ...
                 1            0.001953 -0.78717 -0.068275  0.15890  0.71829  ...
                 1            0.000204 -0.78664 -0.068277  0.15879  0.69849  ...
                 1            0.005649 -0.78529 -0.069849  0.15912  0.72799  ...
                 1           -0.005346 -0.78742 -0.068796  0.15916  0.71572  ...

                                RL_zmag  LL_xacc  LL_yacc  LL_zacc  LL_xgyro  \
Activity Subject Time (s)
1        1       1          -0.036453  -2.8071  -9.0812   2.6220 -0.000232
                 1          -0.034005  -2.8146  -9.0737   2.6218 -0.014784
                 1          -0.036489  -2.8221  -9.0886   2.6366 -0.012770
                 1          -0.036151  -2.8071  -9.0811   2.6070 -0.005725
                 1          -0.033807  -2.8146  -9.0737   2.6218 -0.003929

                                LL_ygyro  LL_zgyro  LL_xmag  LL_ymag   LL_zmag
Activity Subject Time (s)
1        1       1          -0.012092 -0.004457  0.74017  0.30053 -0.057730
                 1          -0.016477  0.002789  0.73937  0.30183 -0.057514
                 1           0.005717 -0.007918  0.73955  0.30052 -0.057219
                 1           0.009620  0.006555  0.74029  0.30184 -0.057750
                 1          -0.008371  0.002816  0.73845  0.30090 -0.057527

[5 rows x 45 columns]
```

Donde se observa un tamaño de la base de datos correspondiente a 1,140,000 filas, propias de la cantidad de muestras con las que contamos a lo largo del tiempo, y 45 columnas, correspondientes a las lecturas propias de cada eje para cada sensor.

3.2.2 Atributos, distribución y resúmenes estadísticos

Antes de proceder con cualquier técnica de preprocesamiento de datos, debemos entender la estructura de estos [2], lo cual involucra examinar la distribución, escalas, valores atípicos y correlaciones entre características, entre otros. Es debido a lo anterior que procede la descripción de los atributos con los que contamos, así como sus resúmenes estadísticos. Es importante aclarar que, como bien mencionan los mismos autores, no poseemos con datos nulos, por lo que resulta innecesario el filtrarlos. Primeramente, observamos los tipos de datos:



Donde podemos ver que, todas las columnas son de tipo flotante, por lo que contamos con información meramente cuantitativa, lo cual es importante para saber cómo proceder con las técnicas de preprocesamiento y clasificadores [3].

En tema de distribución, por clase, contamos con lo siguiente:

```
1     60000
11    60000
18    60000
17    60000
16    60000
15    60000
14    60000
13    60000
12    60000
10    60000
2     60000
9     60000
8     60000
7     60000
6     60000
5     60000
4     60000
3     60000
19    60000
Name: Activity, dtype: int64
```

Como se puede observar las clases se encuentran balanceadas, es decir, se cuenta con la misma cantidad de ejemplos para cada una de las diferentes clases, esto es de gran utilidad puesto que no tenemos que aplicar ninguna tecnica para el balanceo de clases, esto nos permite evitar el overfitting en el caso de clases con mayor cantidad de ejemplos y el underfitting en clases minoritarias.

Respecto a los resúmenes estadísticos, debido a la carga computacional de técnicas como la transformación Fourier y otras [4], se optó por obtener los siguientes estadísticos, que resultan ser lo suficientemente básicos como para agilizar el proceso de obtención de estos, pero a la vez lo suficientemente descriptivos para brindarnos información valiosa sobre los datos:

• Media: El valor promedio de la aceleración o velocidad angular para cada ventana.

• Desviación estándar: La medida de la dispersión de los valores de aceleración o velocidad angular alrededor de la media.

• Varianza: El grado de dispersión que indica cómo la aceleración o velocidad angular difiere de la media. También complementa la desviación estándar.

• Mediana: El valor central de la aceleración o velocidad angular ordenada en cada ventana.

• Mínimo: El valor más pequeño registrado de aceleración o velocidad angular.

• Máximo: El valor más grande registrado de aceleración o velocidad angular.

Obteniendo consigo los valores siguientes:

	T_xacc	T_yacc	T_zacc	T_xgyro	T_ygyro	T_zgyro
count	1140000.000000	1140000.000000	1140000.000000	1140000.000000	1140000.000000	1140000.000000
mean	7.765766	-0.811036	2.768845	-0.002796	0.013695	-0.003312
std	5.637887	2.623027	3.538260	0.794011	0.691040	0.310766
min	-99.715000	-49.941000	-62.664000	-27.851000	-23.598000	-12.067000
25%	6.907000	-1.509500	0.899845	-0.162892	-0.102480	-0.097480
50%	8.830300	-0.389530	2.703700	0.000461	0.017438	-0.002664
75%	9.690500	0.413620	4.405200	0.164820	0.131310	0.088826
max	93.694000	41.013000	120.530000	27.671000	14.379000	19.262000

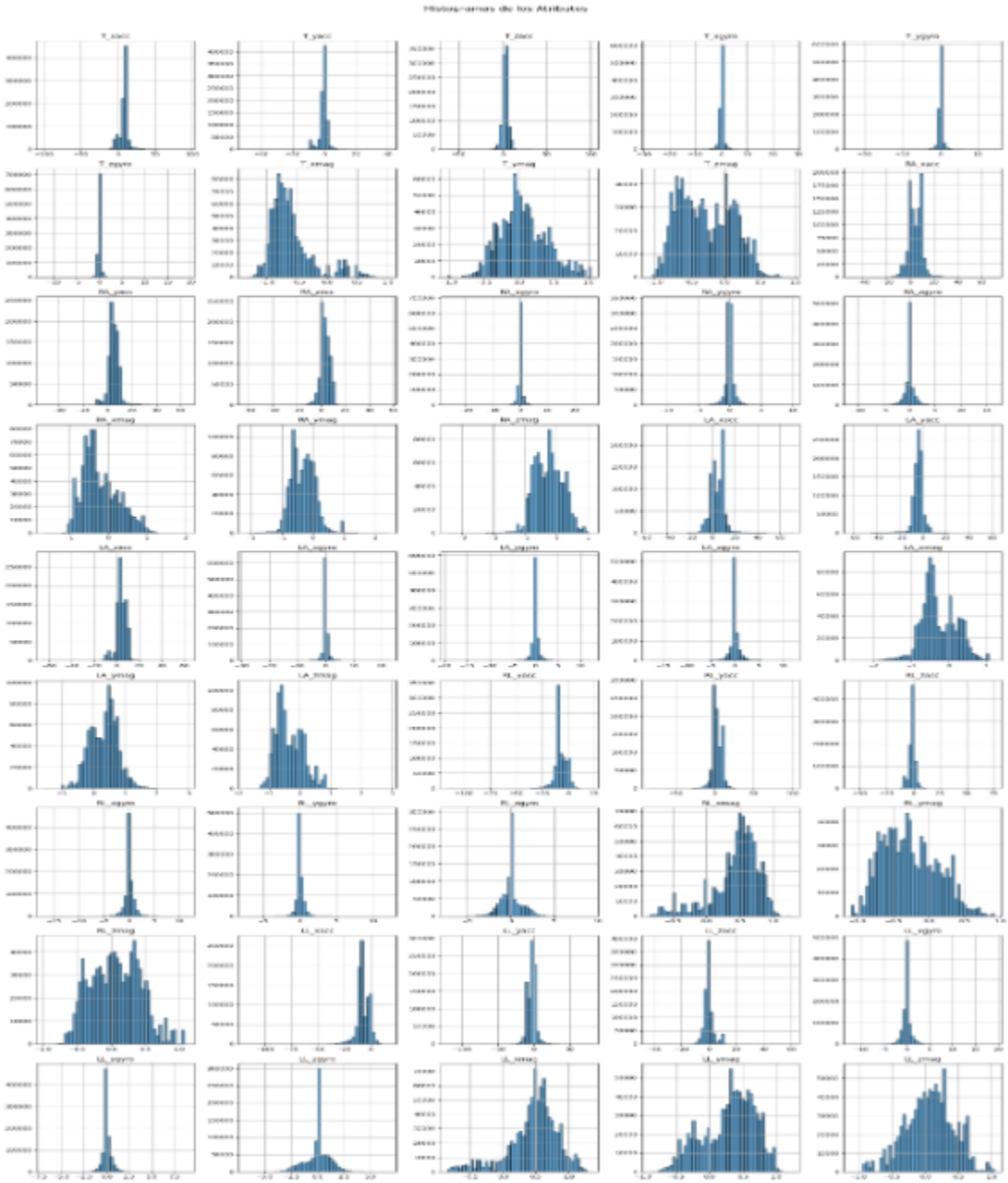
Analizando el resumen estadístico podemos notar que en general los magnetómetros tienen una desviación estándar muy pequeña, así como un rango de valores bastante reducido en comparación con los demás sensores, por otra parte se puede ver como en los giroscopios y acelerómetros tenemos un rango de valores amplio, sin embargo analizando la media podemos ver que estos valores máximos y mínimos distan bastante de la media, por lo que podría tratarse de outliers ya sea por fallo en la medición del sensor, movimientos bruscos, ruido o alguna otra causa, por otro lado analizando el primer y el tercer cuartil podemos ver más marcado el estos valores extremos, por ejemplo en el acelerómetro del eje X del torso podemos ver como el primer cuartil nos dice que el 25% de los datos son menores o iguales a 6.92 pero el valor mínimo es –99.71, a su vez, el 75% de los datos son menores o iguales a 9.68 pero el valor máximo asciende a 93.42 por lo que podemos interpretar que el 50% de los datos se encuentran entre 6.92 y 9.68 un rango bastante pequeño al de la muestra total – 99.71 a 93.42, al ser diferentes tipos de sensores tienen diferentes tipos de medidas, por esta razón se puede ver como los rangos de valores de cada tipo de sensor son diferentes entre sí, además de depender de la posición donde es colocado el sensor, por ejemplo la desviación estándar del acelerómetro en el eje Y del torso es de 2.6 que es menor a la desviación estándar del acelerómetro en el eje Y del brazo derecho 4.57.

Para una mejor visualización de los estadísticos, se proponen los siguientes gráficos, los cuales ayudarán a identificar con más facilidad patrones en los datos y la correlación entre los mismos, lo cual contribuirá a la decisión entre permanencia o desalojo de ciertas características para evitar la redundancia y disminuir dimensionalidad [5]. [6]

3.2.3 Visualizaciones de los datos e interpretaciones

Histograma de los atributos

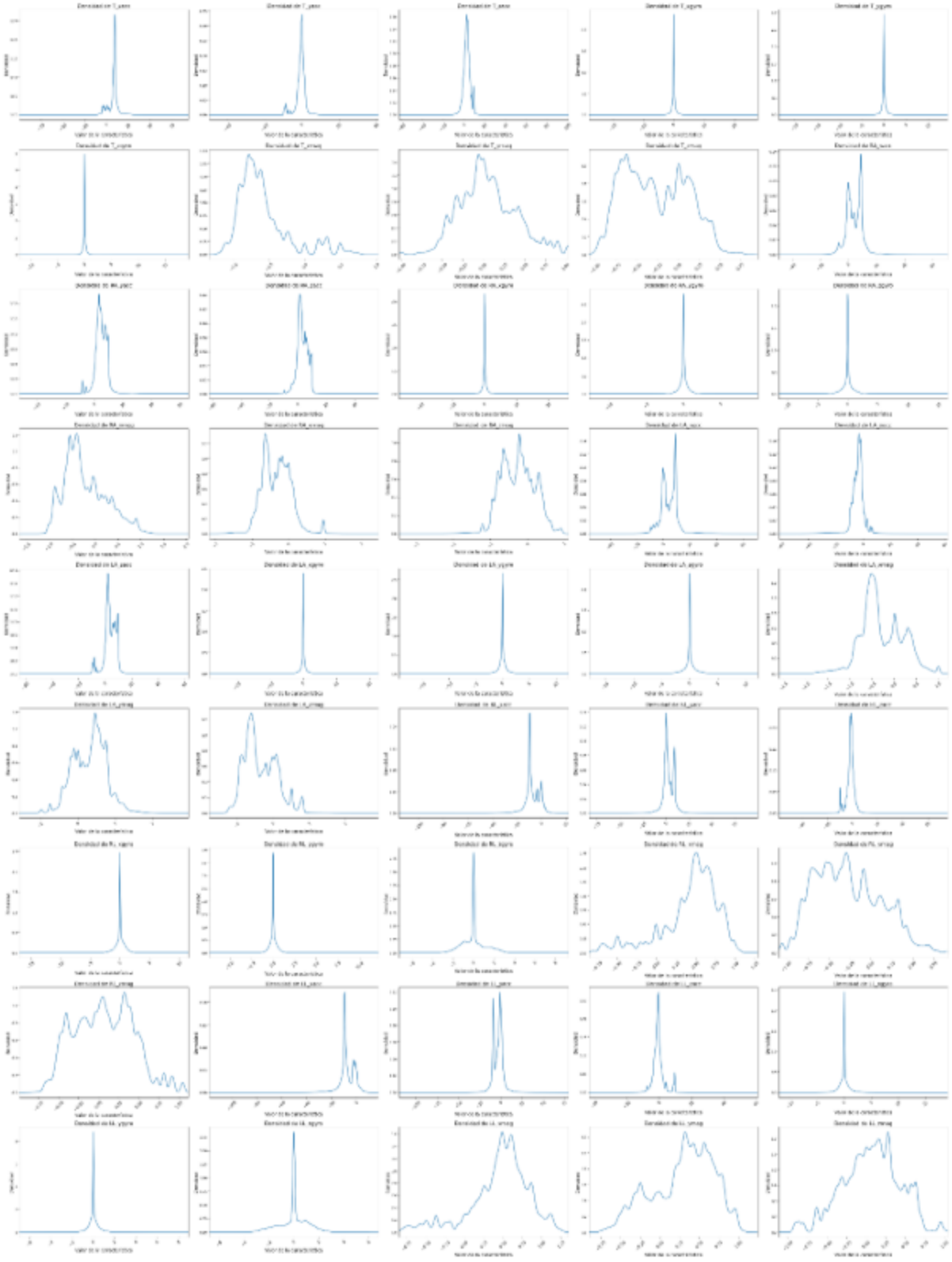
El análisis de la distribución de los atributos es un paso esencial en la exploración de datos, ya que nos permite comprender cómo se comportan las variables en nuestro dataset. Para ello, utilizamos histogramas, los cuales muestran la frecuencia con la que aparecen ciertos valores dentro de cada atributo. Estos gráficos son útiles porque permiten, revelar la distribución de los datos, detectar la presencia de outliers, ayudar a tomar decisiones respecto a técnicas de preprocesamiento, etc.



Se puede apreciar como en los giroscopios y acelerómetros los histogramas tienden a una distribución cercana a la normal, mientras que en los magnetómetros no, así como se aprecia la presencia de outliers en los giroscopios y acelerómetros al haber frecuencias muy bajas en valores muy lejanos, esto se debe principalmente a que estas mediciones provienen de múltiples fuentes de variabilidad como pueden ser, ruido en el sensor, vibraciones mecánicas, errores de cuantización, variaciones en la posición y orientación del sensor.

Gráfica de densidad para la distribución de los atributos

Los gráficos de densidad nos permiten visualizar la distribución de los atributos de manera más suave y continua en comparación con los histogramas. Estos ayudan a identificar patrones y tendencias de una manera más clara, así como facilitan la detección de multimodalidad (picos), etc.



Se puede apreciar las gráficas de densidad de los acelerómetros y giroscopios tienden a tener una distribución normal, así como la presencia de varios picos en las gráficas de los magnetómetros, el hecho de que las gráficas de densidad de los acelerómetros y giroscopios tiendan a tener una distribución normal se puede explicar debido al teorema del límite central. El teorema del límite centra nos dice que, la suma de un conjunto de variables aleatorias, que, por supuesto es en sí misma una variable aleatoria, tiene una distribución que se vuelve cada vez más gaussiana a medida que aumenta el número de términos en la suma (Walker, 1969).

Gráfica de cajas y bigotes para la distribución de los atributos

Los gráficos de caja y bigotes (boxplots) son una herramienta fundamental para el análisis exploratorio de datos, ya que permiten visualizar la distribución y variabilidad de cada atributo de manera concisa.

Se puede apreciar cómo se refuerza lo analizado anteriormente, el en el caso de los acelerómetros podemos ver como los bigotes se extienden en un rango muy amplio, confirmando la presencia de outliers, se puede ver como en el caso de los magnetómetros y giroscopios la mediana tiende a 0.

Como se mencionó anteriormente en la gráfica de densidad los datos tienden a una distribución normal, además se vio en el histograma la presencia de outliers, así como el gráfico de caja y bigotes, esto se debe a que estas mediciones provienen de múltiples fuentes de variabilidad como pueden ser, ruido en el sensor, vibraciones mecánicas, errores de cuantización, variaciones en la posición y orientación del sensor, esto permite que podamos aplicar el teorema del límite central a la suma de estos datos, razón por la que se decide dejar estos valores atípicos, ya que contribuyen a que se cumpla el teorema del límite central y los datos tiendan a una distribución normal.

Matriz de correlación de los atributos

La matriz de correlación es una herramienta fundamental en el análisis de datos, ya que nos permite identificar la relación entre los diferentes atributos del dataset. En el contexto del análisis de movimiento basado en sensores, la correlación nos ayuda a detectar qué variables tienen una relación lineal significativa, lo que puede ser útil para reducir dimensionalidad, evitar redundancia, evitar colinealidad, entre otros.

A pesar de lo anterior, Schober, Boer y Schwarte (2018) advierten que la clasificación de los coeficientes de correlación en niveles como "débil" o "moderado" es arbitraria y puede variar según el criterio empleado. Aunque valores extremos suelen generar consenso, los intermedios pueden ser interpretados de diferentes maneras dependiendo de la regla utilizada.

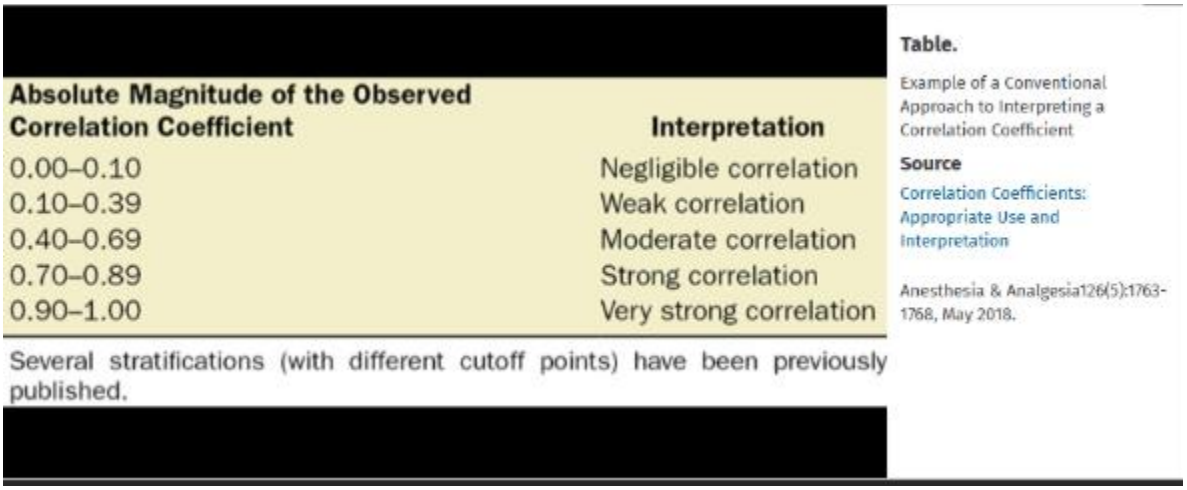


Table.

Example of a Conventional Approach to Interpreting a Correlation Coefficient

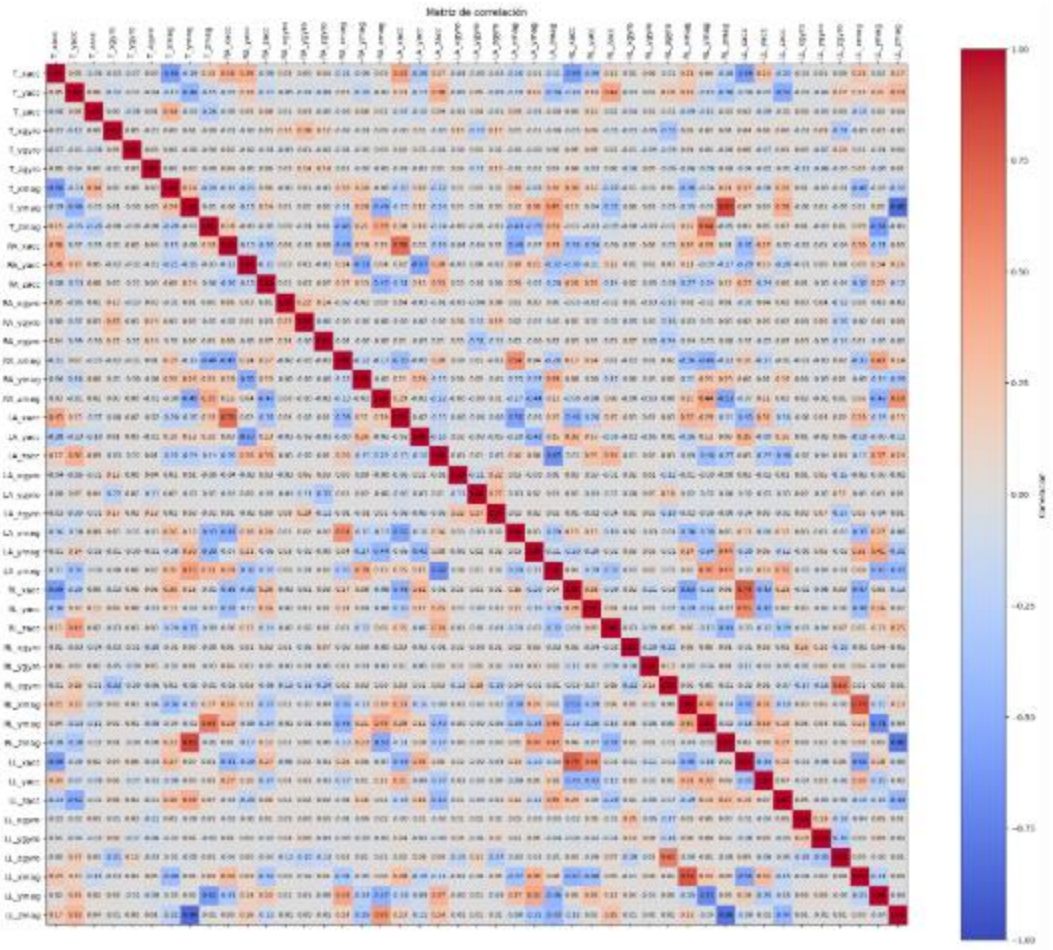
Source

Correlation Coefficients: Appropriate Use and Interpretation

Anesthesia & Analgesia126(5):1763-1768, May 2018.

Absolute Magnitude of the Observed Correlation Coefficient	Interpretation
0.00–0.10	Negligible correlation
0.10–0.39	Weak correlation
0.40–0.69	Moderate correlation
0.70–0.89	Strong correlation
0.90–1.00	Very strong correlation

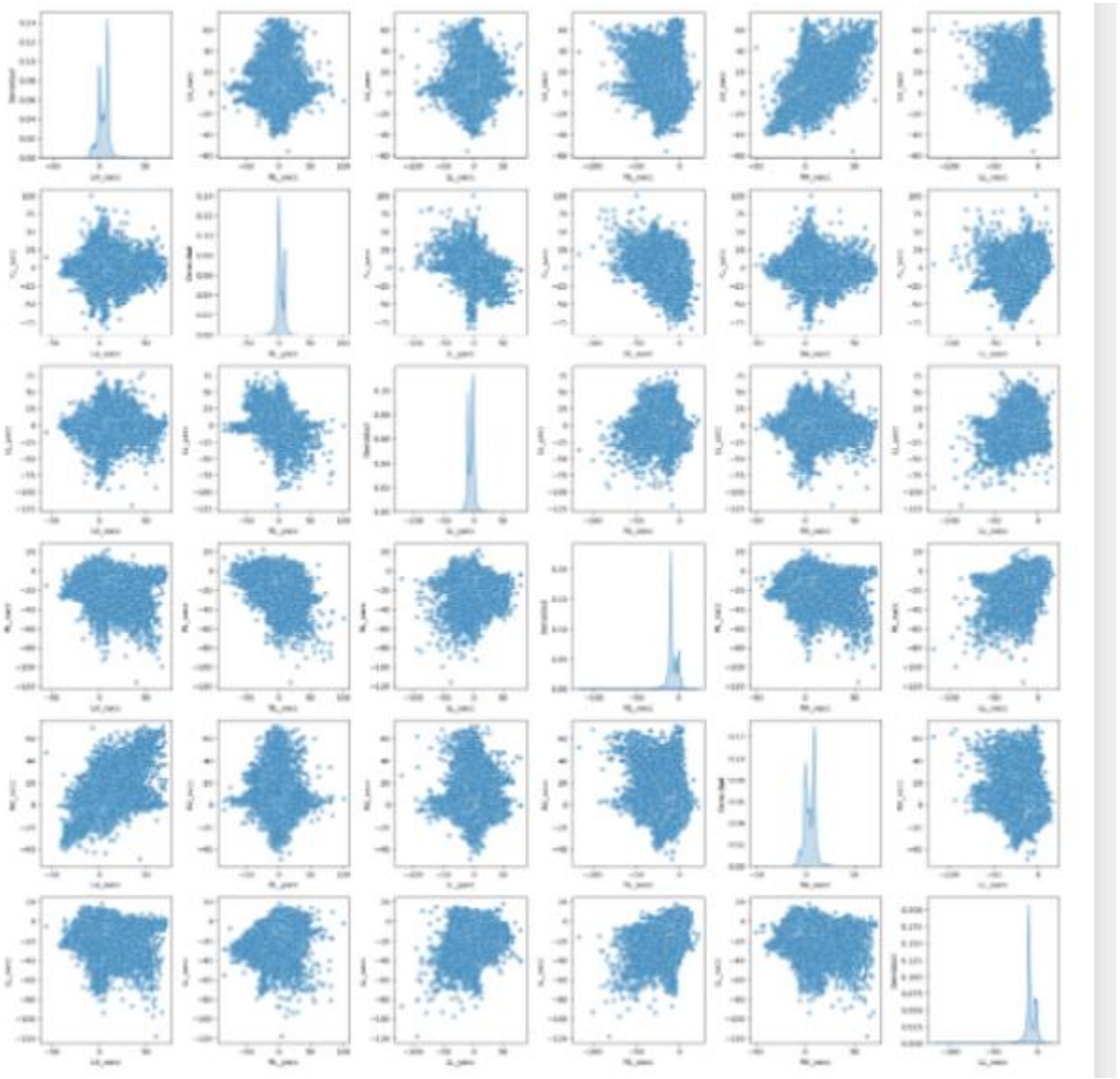
Several stratifications (with different cutoff points) have been previously published.



Analizando la matriz de correlación se puede ver como la mayoría de los atributos no tienen una alta correlación entre sí, sin embargo, se puede ver ciertos grados de correlación entre el acelerómetro del eje X de la pierna derecha y la pierna izquierda lo cual es algo normal debido a que a medida que una pierna empieza a acelerar la otra lo hará igual para mantener un equilibrio.

Matriz de dispersión de los atributos

La matriz de dispersión (scatter plots matrix) es una herramienta visual clave para analizar la relación entre pares de atributos dentro del dataset.



A grandes rasgos podemos observar una dispersión considerable y aparentemente irregular respecto a todos los pares de datos.

3.2.4 Escalamiento

Después de realizar un análisis a los datos se concluyó por utilizar la técnica de estandarización

Los motivos para elegir la estandarización sobre otras técnicas de preprocesamiento están fuertemente relacionadas con el análisis que se tuvo de los datos, en primer lugar, en el resumen estadístico logramos ver como los rangos de valores entre los diferentes sensores eran muy

diferentes entre sí, parte de esto se debe a que las unidades de medición de los sensores son bastante distintas entre sí desde m/s², rad/s, microtesla, de acuerdo con Aurélien Géron (2019) los algoritmos de machine learning no tienen un buen desempeño cuando las entradas numéricas de los atributos tienen escalas muy distintas, además de que no contamos con un valor mínimo y máximo definidos, algo que puede ser perjudicial si quisiéramos aplicar min-max scaler, por otra parte, analizando las gráficas de caja y bigote podemos ver cómo hay valores extremos, es decir existen muchos valores atípicos, que, como menciona Paula Borras (2023), la estandarización es mejor opción que la normalización cuando se tienen valores atípicos porque estos datos pueden afectar significativamente a la escala usada en la normalización, de igual manera, analizando el gráfico de densidad y los histogramas podemos ver como los acelerómetros y giroscopios tienden a tener una gráfica parecida a una distribución normal, algo bastante beneficioso si queremos usar estandarización de acuerdo con Sachin Vinay (2021), además como se mencionó anteriormente, Walker (1969) nos dice que esto se debe al teorema del límite central.

Al usar estandarización la parte de las escalas diferentes de los sensores deja de ser un problema y todas las características aportan de manera equitativa al modelo, algo bastante favorable para modelos como regresión lineal y Redes Neuronales de acuerdo con Sachin Vinay (2021) al decir que estos algoritmos utilizan una técnica de optimización "Gradient descent" que requiere que los datos tengan la misma escala debido a que puede ayudar a que converja más rápido, de igual forma favorece a algoritmos como KNN, SVM y K-means ya que, según Liu (2022), la estandarización es útil para aquellos algoritmos basados en el cálculo de distancias entre puntos, ya que requieren que los datos se dispongan en una escala uniforme.

3.2.5 Segmentación en ventanas y extracción de características

Debido a que ha sido demostrado que una ventana de 3s sin superposición es tiempo suficiente para el reconocimiento de la actividad humana de acuerdo con estudios previos (Banos, O et al, 2014), optamos por seccionar nuestro conjunto de datos para dar paso a la extracción de características. Dado que la frecuencia de muestreo para este dataset es de 25hz, tenemos que 25 ejemplos son capturados cada segundo, para cada uno de los 3 ejes presentes en cada uno de los sensores. Con esta información podemos obtener el número de muestras en cada ventana (25 x 3 = 75 muestras por ventana de tiempo).

Asimismo, como resultado de experimentos previos realizados por Dernbach, S. et al (2012), se decidió por extraer 16 características propias para cada ventana de tiempo de cada sensor, siendo estas las siguientes:

- Valor medio para cada uno de los 3 ejes
- Desviación estándar para cada uno de los 3 ejes
- Valor máximo para cada uno de los 3 ejes
- Correlación entre cada par de ejes (x – y, x – z y y – z, tanto para el acelerómetro como el giroscopio)

- Valor medio de la magnitud
- Desviación estándar de la magnitud
- Magnitud del área bajo la curva (AUC, fig. 1)
- Diferencia de medias de magnitud entre lecturas consecutivas (fig 2)

, donde la magnitud de la señal representa la contribución en general de la aceleración de los 3 ejes (fig 3)

$$AUC = \sum_{t=1}^{T} magnitude(t)$$

(fig 1)

$$meandif = \frac{1}{T-1} \sum_{t=2}^{T} magnitude(t) - magnitude(t-1)$$

(fig 2)

$$Magnitude(x, y, z, t) = \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2},$$

(fig 3)

Se trabajó con el dataset agregado entre los ejemplares de acelerómetro, giroscopio y magnetómetro. Se tomó la decisión de usar los 3 sensores, dado que según García-Ceja et al. (2018), experimentos previos donde se han calculado las 16 características en estos sensores han mostrado un alto rendimiento en modelos para el reconocimiento de actividades humanas.

Table 7
Performance metrics results for Opportunity dataset. Average (standard deviation).

	Accuracy	Recall	Specificity
Accelerometer view	0.843 (0.037)	0.790 (0.066)	0.925 (0.016)
Gyroscope view	0.821 (0.025)	0.692 (0.043)	0.914 (0.011)
Magnetometer view	0.889 (0.024)	0.855 (0.051)	0.948 (0.012)
Aggregated views	0.914 (0.020)	0.862 (0.036)	0.957 (0.009)
Multi-View Stacking	**0.925 (0.026)**	**0.905 (0.043)**	**0.965 (0.011)**
Reported in Sagha et al. [68]	Average accuracy of 0.83		

Observando la estructura general de las primeras filas tenemos:

```
In [24]: data_features.head()
```

Out[24]:

Activity	Window Number	Body Part	Sensor Type	mean_x	mean_y	mean_z	std_x	std_y	std_z	max_x	max_y	max_z	corr_xy	corr_xz	corr_yz
1	0	T	acc	8.015509	1.058076	5.553903	0.129444	0.039797	0.191729	8.160500	1.176200	6.181400	-0.818854	-0.915012	0.764440
			gyro	0.007880	0.038442	-0.033583	0.014530	0.035587	0.037627	0.045403	0.214780	0.015614	0.479487	0.123089	-0.251151
			mag	-0.789640	-0.068427	0.141308	0.023081	0.001542	0.017152	-0.785290	-0.054353	0.159160	-0.758304	0.981221	-0.823023
		RA	acc	0.701996	5.677145	7.950649	0.026060	0.037776	0.029033	0.759300	5.755700	8.020800	-0.389789	0.156030	-0.505236
			gyro	0.011151	-0.021966	-0.031992	0.010110	0.007139	0.007618	0.038867	0.014142	0.013977	0.010500	-0.173294	0.002163

```
In [25]: print(data_features.columns)

Index(['mean_x', 'mean_y', 'mean_z', 'std_x', 'std_y', 'std_z', 'max_x',
       'max_y', 'max_z', 'corr_xy', 'corr_xz', 'corr_yz', 'mean_magnitude',
       'std_magnitude', 'auc_magnitude', 'mean_diff_magnitude'],
      dtype='object')
```

Aquí se aprecia la totalidad de las 16 columnas, donde cada columna corresponde a una característica extraída, debido a que cada ejemplo es un vector de características, por esta razón analizando la estructura del dataset tenemos 1,140,000 muestras en total entonces al hacer ventanas de 3 segundos de 25 muestras por cada uno entonces deberemos tener 1,140,000/75 muestras que son 15,200

```
Ventanas totales: 15200
Train ventanas: (11400, 242)
Test ventanas: (3800, 242)
```

3.2.6 Resultados de Clasificadores (Regresión Logística, Bosques Aleatorios, Perceptrón)

En este apartado se presentan los resultados de los tres clasificadores seleccionados—**regresión logística**, **bosques aleatorios** y **perceptrón multicapa**—utilizando como métrica de evaluación la **exactitud (accuracy)**.

Con el fin de estimar de manera robusta la capacidad de generalización de cada modelo, se implementó **validación cruzada** con GroupKFold y Leave One Group Out, esto debido a que estamos trabajando con muestreos de 8 sujetos de prueba por lo que debemos asegurarnos de que no hayan muestras de un mismo sujeto dentro de entrenamiento y pruebas a la vez, esto debido a que el modelo puede aprender de los modelos de un sujeto y al probarlo en pruebas acertará

teniendo un tipo de data leakage, para evitar esto usamos GroupKFold que se asegura de que al momento de hacer cross validation agarre a un grupo entero de sujetos y evalue con ellos, de manera similar Leave One Group Out es similar dejando 1 solo grupo de sujetos como validación.

A continuación, en los apartados siguientes se detallan los scores de entrenamiento, test y validación cruzada para cada clasificador:

1. Regresión Logística

Resultados

- **Training set score:** 0.9932
- **Test set score:** 0.9268
- **GroupKFold 5-fold CV:** [0.9166, 0.8842, 0.9237, 0.8137, 0.8953] (media = 0.8867)
- **GroupKFold 6-fold CV:** [0.9084, 0.8842, 0.9237, 0.8137, 0.8953, 0.8900] (media = 0.8859)

Interpretación

La regresión logística muestra un desempeño excelente en el conjunto de entrenamiento (99.32 %), lo que indica que la función sigmoide y la penalización empleada son capaces de ajustar casi perfectamente las observaciones. En el conjunto de prueba, el score del 92.68 % confirma su buena capacidad de generalización, aunque ligeramente inferior al de entrenamiento.

La validación cruzada arroja medias cercanas al 88.6 % en ambos esquemas (5-fold y 6-fold), con cierta dispersión entre pliegues (el mínimo en torno a 81 % y el máximo alrededor de 92 %). Esta variabilidad sugiere que, en grupos con distribuciones de características menos favorecedoras para un modelo lineal, la regresión logística alcanza un rendimiento algo menor. No obstante, la diferencia de ~0.04 entre las medias de CV y el test set indica un bajo grado de sobreajuste y una estabilidad razonable frente a cambios en la partición de los datos.

2. Bosques Aleatorios

Resultados

- **Training set score:** 0.9996
- **Test set score:** 0.9218
- **GroupKFold 5-fold CV:** [0.8700, 0.9237, 0.9158, 0.9684, 0.8637] (media = 0.9083)
- **GroupKFold 6-fold CV:** [0.9016, 0.9237, 0.9158, 0.9684, 0.8637, 0.9000] (media = 0.9122)

Interpretación

El bosque aleatorio alcanza prácticamente el 100 % en el entrenamiento, lo que revela un ajuste casi completo de las muestras sobre las que se entrenaron los árboles. Pese a este aparente

sobreajuste, el desempeño en el test set (92.18 %) y las medias de CV (90.83 % en 5-fold; 91.22 % en 6-fold) permanecen altos y consistentes.
La agregación de múltiples árboles (bagging) y la selección aleatoria de variables en cada división permiten reducir la varianza global, compensando el exceso de ajuste de cada árbol individual. La ligera ventaja de la media de 6-fold sobre la de 5-fold sugiere que más particiones pueden aportar una evaluación marginalmente más robusta, aunque ambas configuraciones reflejan una generalización sólida por encima del 90 %.

3. Perceptrón Multicapa (MLP)

Resultados

- **Training set score:** 0.9675
- **Test set score:** 0.9003
- **GroupKFold 5-fold CV:** [0.8413, 0.8037, 0.8316, 0.8232, 0.8937] (media = 0.8387)
- **GroupKFold 6-fold CV:** [0.8595, 0.8037, 0.8316, 0.8232, 0.8937, 0.8447] (media = 0.8427)

Interpretación
El perceptrón multicapa logra un 96.75 % en entrenamiento, indicador de buena capacidad de ajuste, y un 90.03 % en test, lo que denota una caída más pronunciada que en los otros modelos. La validación cruzada refleja medias entre 83.9 % y 84.3 %, con rangos que van del 80.4 % al 89.4 %.
Esta mayor variabilidad y el gap entrenamiento-CV (~0.13) apuntan a que la red es sensible a la configuración de sus hiperparámetros (profundidad, número de neuronas, tasa de aprendizaje) y al preprocesamiento de las características. En pliegues con datos menos representativos o más ruidosos, el MLP muestra un desempeño reducido, lo que sugiere necesidad de mayor afinamiento y quizá de mayor volumen de datos para estabilizar su entrenamiento.

Tomando en cuenta la información anterior, se puede decir que la buena (> 90%) pprecisión de los diversos clasificadores se sostiene en un pipeline optimizado: primero, empleamos ventanas de **3 s sin solapamiento**, que capturan ciclos completos de cada actividad y mejoran la calidad de los patrones extraídos (Banos et al., 2014).

Posteriormente, la **extracción de 16 características** estadísticas —media, desviación estándar, máximos, correlaciones, AUC, etc.— por ventana condensa la información triaxial de acelerómetro, giroscopio y magnetómetro en descriptores altamente discriminativos (Dernbach et al., 2012). Después, la **estandarización (z-score)** uniformiza las escalas y permite que los optimizadores basados en gradiente converjan más rápidamente y sin oscilaciones ineficientes (Borras, 2023). Además, la tendencia de las **distribuciones de aceleraciones y rotaciones** a

aproximarse a la normalidad, según el **Teorema del Límite Central**, respalda estadísticamente esta elección de escalado. Por último, la mínima variación entre los resultados de **5-fold y 6-fold CV** (diferencias medias < 0.001) confirma la baja varianza en las particiones y refuerza la solidez de la generalización de los modelos (Arlot & Celisse, 2010).

3.2.7 Reducción de dimensionalidad (PCA y Kernel PCA)

En ese apartado presentaremos las distintas técnicas de reducción de características usadas para optimizar los modelos entrenados. Las técnicas presentadas son las siguientes:

- Sin reducción de dimensionalidad
- PCA
- Kernel PCA

Los resultados obtenidos se presentan a continuación:

```
Resultados finales:
                Modelo  Sin Reducción Dimensionalidad       PCA  Kernel PCA
0        Random Forest                        0.921842  0.909474    0.092632
1  Logistic Regression                        0.926842  0.928684    0.085000
2           Perceptron                        0.900263  0.877895    0.101053
```

1. Efecto de PCA

- **Random Forest**: la exactitud desciende ligeramente de 0.9218 a 0.9095 al pasar de todas las variables a los componentes principales. Esto indica que, aunque PCA captura la mayor parte de la varianza global, elimina algunas dimensiones que contenían información discriminativa para los árboles, reduciendo su capacidad de partición óptima.
- **Regresión Logística**: se observa un pequeño aumento de 0.9268 a 0.9287, lo cual es consistente con que PCA tiende a filtrar ruido y colinealidad, beneficiando a modelos lineales que asumen independencia parcial entre características.
- **Perceptrón Multicapa**: la accuracy baja de 0.9003 a 0.8779, sugiriendo que la representación reducida por PCA restringe la capacidad del MLP para aprender ciertas interacciones no lineales presentes en el conjunto original.

En conjunto, PCA demuestra ser útil para modelos lineales como la regresión logística, pero puede restar potencia a algoritmos que dependen de información de alta resolución o relaciones no lineales complejas.

2. Comportamiento de Kernel PCA

La aplicación de Kernel PCA (KPCA) a nuestros tres clasificadores provocó una caída drástica del accuracy hasta valores cercanos al azar (~0.09–0.10). Este mal desempeño puede justificarse por varias limitaciones bien documentadas:

1. **Falta de garantías teóricas fuertes**
   Aunque KPCA extiende PCA mediante núcleos no lineales, su eficiencia media depende de "propiedades deseables" del conjunto de datos (distribución, dispersión) para asegurar una convergencia rápida. En ausencia de tales condiciones, los autovalores del operador kernel pueden no reflejar direcciones informativas, y la reducción degenera en ruido puro (Wang & Schölkopf, 2021).

2. **Escalabilidad deficiente**
   Francis y Raimond (2017) demostraron que, en tareas de clasificación sobre grandes volúmenes de datos, las aproximaciones a KPCA (Nyström, RNCA, SKPCA) superan con creces al KPCA estándar, cuya exactitud es muy baja si no se usan métodos de aproximación o submuestreo estratégico (Francis & Raimond, 2017).

3. **Ausencia de correlaciones no lineales significativas**
   Cuando el dataset no presenta relaciones no lineales fuertes o estas se distribuyen en múltiples escalas, KPCA no aporta ventajas respecto a un PCA lineal y, de hecho, puede introducir ruido en las dimensiones transformadas, reduciendo la separabilidad de clases (Amoeba, 2015).
4. **Desempeño al nivel del azar**
   La combinación de un kernel mal configurado (por defecto RBF sin ajuste de γ) y la transformación implícita a un espacio de dimensión elevada puede inducir tanto underfitting como overfitting, llevando a accuracies prácticamente aleatorias (~0.10) en tareas de clasificación (Zhu et al., 2018).

En resumen, el colapso de la performance tras aplicar KPCA señala un **desajuste** entre la transformación no lineal y la estructura real de los datos de DailySportsandActivities. Tomando en cuenta lo anterior, es posible proporcionar las siguientes recomendaciones a la hora de evaluar métricas de desempeño aplicando esta técnica de reducción de características:

- Realizar **búsqueda de hiperparámetros** (tipo de kernel y γ) mediante grid search.
- Evaluar **técnicas de aproximación** (Nyström, RNCA, SKPCA) en lugar del KPCA estándar.
- Confirmar previamente la presencia de **relaciones no lineales** significativas antes de aplicar mapeos kernel.

3.2.7 Métricas de desempeño

En el contexto de la evaluación de los clasificadores, se emplea la validación cruzada con 2-fold, 5-fold, y 10-fold, tomando como referencia el estudio de Hsu et al. (2018) [19]. El uso de diferentes estrategias de validación cruzada permite obtener una evaluación más exhaustiva del modelo, minimizando la varianza de los resultados y mejorando la capacidad de generalización del modelo al no depender de una única partición del conjunto de datos.

Exactitud/Accuracy

Se calcula como la proporción de predicciones correctas sobre el total de muestras. Es una medida de la consistencia entre las predicciones del modelo y los resultados reales.

Donde:

- TP (True Positives): Casos positivos correctamente clasificados.
- TN (True Negatives): Casos negativos correctamente clasificados.
- FP (False Positives): Casos negativos incorrectamente clasificados como positivos.
- FN (False Negatives): Casos positivos incorrectamente clasificados como negativos.

Precisión

Indica la proporción de predicciones positivas que son realmente correctas, y es relevante cuando los costos de los falsos positivos son elevados.

Recall/Sensibilidad

Mide la capacidad del modelo para identificar correctamente los casos positivos, siendo crucial cuando los falsos negativos son costosos. Es una medida de cuántos de los verdaderos casos positivos han sido capturados por el modelo.

F1-Score

Es la media armónica entre la precisión y el recall, proporcionando una medida balanceada entre ambas, especialmente útil cuando las clases están desbalanceadas.

Las métricas utilizadas para evaluar los clasificadores incluyen precisión, recall, F1-score y exactitud. Estas métricas fueron definidas en el trabajo [19], quienes explican que el F1-score, la precisión y el recall son fundamentales para la evaluación del desempeño de los clasificadores en tareas de reconocimiento de actividades.

Resultados - Regresión Logística

	CV	Accuracy	Precision	F1-score	Recall
0	KFold 3	0.865789	0.897113	0.858787	0.865789
1	LeaveOneGroupOut	0.886579	0.896122	0.870130	0.886579

Interpretación

- **Accuracy y Recall** crecieron de 0.8658 (KFold-3) a 0.8866 (LOGO), indicando que, al agrupar por la variable de grupo, el modelo clasifica correctamente un mayor porcentaje de casos verdaderos.
- **Precision** se mantiene estable alrededor de 0.896, lo que sugiere que la proporción de verdaderos positivos entre todas las predicciones positivas no varía significativamente al cambiar el esquema.
- El **F1-score** experimenta una mejora (0.8588 → 0.8701) al emplear LOGO, reflejando que el aumento simultáneo de recall y la estabilidad de precision fortalecen el balance global.
- **Conclusión**: la regresión logística gana robustez frente a la división por grupos, con mayor recall sin sacrificar precision, lo cual la hace especialmente adecuada cuando interesa minimizar falsos negativos en poblaciones heterogéneas.

Resultados – Bosques Aleatorios

	CV	Accuracy	Precision	F1-score	Recall
0	KFold 3	0.883772	0.920852	0.876375	0.883772
1	LeaveOneGroupOut	0.912281	0.911580	0.900473	0.912281

Interpretación

- Al pasar de KFold-3 a LOGO, la **accuracy** sube de 0.8838 a 0.9123, señal de que el modelo generaliza aún mejor cuando valida dejando fuera grupos completos.
- La **precisión** desciende ligeramente (0.9209 → 0.9116), indicando un pequeño aumento de falsos positivos en LOGO, pero el **recall** compensa con un ascenso notable (0.8838 → 0.9123).
- El **F1-score**, que combina ambos, mejora de 0.8764 a 0.9005, mostrando que el trade-off entre precisión y recall es favorable.
- **Conclusión**: los bosques aleatorios, gracias a su bagging y selección de características, maximizan su capacidad de detección en validaciones agrupadas, siendo muy aptos cuando la homogeneidad dentro de grupos no garantiza independencia total.

Resultados - Perceptrón Multicapa

	CV	Accuracy	Precision	F1-score	Recall
0	KFold 3	0.827368	0.855802	0.818908	0.827368
1	LeaveOneGroupOut	0.849912	0.858793	0.828952	0.849912

Interpretación

- El **accuracy** mejora de 0.8274 a 0.8499 al usar LOGO, lo que muestra que la red capta patrones de grupo con mayor fidelidad cuando estos se retiran por completo en cada iteración.
- Tanto **precisión** (0.8558 → 0.8588) como **recall** (0.8274 → 0.8499) aumentan moderadamente en LOGO, reflejando una generalización más equilibrada.
- El **F1-score** sube de 0.8189 a 0.8290, confirmando el beneficio de validar sobre grupos completos.
- **Conclusión**: aunque el perceptrón multicapa parte de valores inferiores a los otros modelos, la tendencia a mejorar en LOGO revela su capacidad para adaptarse a variaciones intragrupo si se entrena adecuadamente.

De manera general, el análisis anterior sugiere que LOGO es el esquema más estricto y realista cuando los datos están organizados en grupos con posibles distribuciones distintas, especialmente si el objetivo es maximizar la detección de casos positivos (recall) sin sacrificar excesivamente la precisión.

3.1.8 Método estadístico para la comparación de clasificadores

Para realizar una evaluación objetiva del rendimiento de los clasificadores empleados en este trabajo, es esencial aplicar métodos estadísticos que permitan determinar las diferencias observadas en las métricas de desempeño son estadísticamente significativas.

Elección del método estadístico

Debido a que en este trabajo se comparan más de dos clasificadores sobre un mismo conjunto de datos utilizando validación cruzada, el método más adecuado es el **Test de Friedman**, seguido de un análisis **Post-hoc de Nemenyi**. Esta elección se fundamenta por las siguientes razones:

El test de Friedman es un método no paramétrico utilizado para detectar diferencias entre varios algoritmos que han sido evaluados sobre múltiples particiones de un dataset. A diferencia de otros

análisis, este test no asume ninguna normalidad ni homogeneidad de varianzas, lo cual es útil cuando se trabaja con métricas de clasificación que no siguen una distribución normal o casi siempre.

De ser así, cuando el test de friedman indica la existencia de diferencias estadísticas significativas entre los clasificadores, se aplica un análisis Nemenyi. Para compara todos los clasificadores entre sí por pares e identificar cuales algoritmos presentan diferencias significativas.

Este enfoque ha sido utilizado y recomendado por Demšar (2006) sobre el uso del test de friedman en conjunto con el análisis post-hoc de Nemenyi. Y en particular, se recomienda el uso de pruebas paramétricas para comparar el rendimiento entre múltiples clasificadores de datos HAR dado que las distribuciones de métricas como la exactitud de o la F1 no son siempre normales ni homogéneas.

Evaluación estadística de los clasificadores bajo KFold-3

Para determinar si las diferencias observadas en accuracy entre los tres clasificadores (Regresión Logística, Bosques Aleatorios y Perceptrón Multicapa) son estadísticamente significativas cuando se evalúan mediante **KFold-3**, aplicamos el **test de Friedman**. Esta prueba no paramétrica compara múltiples algoritmos sobre un mismo conjunto de particiones, controlando el efecto de las repeticiones y sin asumir normalidad en los datos.

- **Estadístico de Friedman ($\chi^2_{(2)}$):** 4.6667
- **p-valor:** 0.0970

```
Friedman statistic: 4.6667, p-value: 0.0970
```

Interpretación
Bajo la hipótesis nula del test de Friedman, se asume que todos los clasificadores tienen **el mismo rendimiento medio.** Dado que el p-valor obtenido (0.0970) es **mayor** que el umbral habitual de significancia ($\alpha = 0.05$), **no podemos rechazar** la hipótesis nula. En otras palabras, con un nivel de confianza del 95 %, **no existen diferencias estadísticamente significativas** en el rendimiento de los tres modelos bajo validación KFold-3.

Evaluación estadística de los clasificadores bajo LOGO

Para comprobar si las diferencias observadas entre Regresión Logística, Bosques Aleatorios y Perceptrón Multicapa cuando se emplea **LeaveOneGroupOut (LOGO)** son estadísticamente significativas, se aplicó de nuevo el **test de Friedman** sobre las repeticiones por grupo.

- **Estadístico de Friedman ($\chi^2_{(2)}$):** 4.3333

- **p-valor:** 0.1146

```
Friedman statistic: 4.3333, p-value: 0.1146
```

Interpretación
La hipótesis nula del test afirma que los tres clasificadores presentan el **mismo rendimiento medio** en términos de exactitud. Al obtener un p-valor de 0.1146, superior al umbral de significancia $\alpha = 0.05$, **no podemos rechazar** dicha hipótesis. Esto implica que, con un nivel de confianza del 95 %, **no existe evidencia suficiente** para afirmar que alguno de los modelos rinda consistentemente mejor o peor que los otros bajo el esquema LOGO.

De manera general, es posible concluir que no existen diferencias significativas en los clasificadores entrenados para las 2 técnicas de validación, empleados, por lo tanto, se puede afirmar con el 95% de confianza de que el rendimiento es similar para cada uno de ellos.

3.1.9 - Estrategias de fusión

En este apartado presentamos la evaluación de tres **métodos de ensamblaje** aplicados a un conjunto de **tres clasificadores base**: **regresión logística**, **bosques aleatorios** y **Naive Bayes**. Las estrategias de ensamblaje analizadas son:

1. **Random Forest** (bagging de árboles de decisión)
2. **AdaBoost** (boosting adaptativo)
3. **Voting** (votación hard/soft entre modelos heterogéneos)

Para cada combinación de clasificador base y método de ensamblaje se calculan las métricas de desempeño:

- **Exactitud (Accuracy)**
- **Precisión (Precision)**
- **Puntuación F1 (F1-score)**
- **Exhaustividad (Recall)**

La robustez de los resultados se comprueba con dos esquemas de validación cruzada:

- **GroupKFold = 3**, donde los datos se dividen en tres pliegues manteniendo la estructura de grupos.
- **LeaveOneGroupOut (LOGO)**, que evalúa iterativamente entrenando sobre todos los grupos menos uno y testeando en el grupo excluido.

Este análisis permite comparar el impacto de cada método de ensamblaje tanto en la capacidad de generalización como en el equilibrio entre precisión y exhaustividad, bajo escenarios de

agrupamiento distintos. A continuación, se exponen los resultados cuantitativos y su correspondiente interpretación.

Evaluación de métodos de ensamblaje bajo KFold-3

	CV	Random Forest Accuracy	Random Forest Precision	Random Forest F1	Random Forest Recall	Voting Accuracy	Voting Precision	Voting F1	Voting Recall	AdaBoost Accuracy	AdaBoost Precision	AdaBoost F1	AdaBoost Recall
0	KFold 3	0.875877	0.904634	0.865401	0.875877	0.878509	0.909148	0.870868	0.878509	0.435614	0.437238	0.384218	0.435614
1	LeaveOneGroupOut	0.905439	0.912004	0.891906	0.905439	0.899035	0.907688	0.886011	0.899035	0.538333	0.507831	0.477914	0.538333
2	Sin Cross Validation	0.913421	0.939896	0.904811	0.913421	0.957632	0.965618	0.956653	0.957632	0.643158	0.640368	0.589854	0.643158

Random Forest

- **Accuracy (0.8758)** y **recall (0.8758)**: el bosque estándar clasifica correctamente casi el 88 % de los ejemplos, detectando de forma equilibrada positivos y negativos.
- **Precision (0.9046)**: muy alta proporción de verdaderos positivos entre las predicciones positivas, refleja baja tasa de falsos positivos.
- **F1-score (0.8654)**: la combinación de precision y recall muestra un balance sólido, cercano a las métricas de accuracy y recall.

Voting

- **Mejora ligera** en todas las métricas respecto a Random Forest:
  - **Accuracy** sube a 0.8785 ($\Delta \approx +0.0027$).
  - **Precision** alcanza 0.9091 y **F1-score** 0.8708.
- **Recall (0.8785)**: también mejora, lo que indica que la votación de varios modelos (por ejemplo, regresión logística + RF + MLP) aporta robustez frente a instancias difíciles.
- **Interpretación**: el ensamblaje por votación capitaliza la diversidad de errores entre los clasificadores base, reduciendo falsos positivos y falsos negativos de forma conjunta.

AdaBoost

- **Rendimiento extremadamente bajo** en todas las métricas (accuracy 0.4356; precision 0.4372; F1-score 0.3842; recall 0.4356).
- **Causa probable**: AdaBoost enfatiza ejemplares mal clasificados en iteraciones sucesivas; si existen outliers o ruido significativo en los grupos definidos por KFold, el algoritmo puede sobreajustar estas muestras atípicas, degradando severamente su capacidad de generalización.
- **Conclusión**: bajo la configuración y parámetros empleados, AdaBoost no resulta adecuado para este dataset; requiere una limpieza rigurosa de ruido o ajustes de los hiperparámetros (profundidad de los estimadores base, tasa de aprendizaje) antes de considerarlo viable.

Conclusiones

1. **Voting** se alza como la alternativa más eficaz, superando ligeramente al Random Forest individual en las cuatro métricas.
2. **Random Forest** mantiene un desempeño muy bueno por sí mismo, con alta precisión y recall equilibrados.
3. **AdaBoost** presenta un colapso de performance, probablemente por su sensibilidad al ruido y a los outliers; será necesario revisar su configuración o aplicar técnicas de preprocesamiento más agresivas.

Este análisis evidencia que, aunque ensamblar múltiples modelos suele mejorar la robustez, no todos los métodos de boosting garantizan buenos resultados sin un ajuste cuidadoso, mientras que un esquema de votación simple puede ofrecer ganancias consistentes con poca complejidad adicional.

Evaluación de métodos de ensamblaje bajo LOGO

	CV	Random Forest Accuracy	Random Forest Precision	Random Forest F1	Random Forest Recall	Voting Accuracy	Voting Precision	Voting F1	Voting Recall	AdaBoost Accuracy	AdaBoost Precision	AdaBoost F1	AdaBoost Recall
0	KFold 3	0.875877	0.904634	0.865401	0.875877	0.878509	0.909148	0.870868	0.878509	0.435614	0.437238	0.384218	0.435614
1	LeaveOneGroupOut	0.905439	0.912004	0.891906	0.905439	0.899035	0.907688	0.886011	0.899035	0.538333	0.507831	0.477914	0.538333

Random Forest

- **Accuracy (0.9054)** y **recall (0.9054)**: se consolida un rendimiento muy alto, detectando correctamente más del 90 % de las observaciones cuando se deja fuera cada grupo completo.
- **Precision (0.9120)**: ligeramente superior a la obtenida con KFold-3, lo que indica menos falsos positivos en validación por grupo.
- **F1-score (0.8919)**: refleja un excelente equilibrio entre precision y recall.

Voting

- **Accuracy (0.8990)**: muy cercana a la de Random Forest, aunque ligeramente inferior ($\Delta \approx -0.0064$).
- **Precision (0.9076)** y **recall (0.8990)**: ambas métricas mantienen valores altos, demostrando que el votador ofrece robustez incluso al validar por grupos enteros.
- **F1-score (0.8860)**: un poco más bajo que el de Random Forest, pero aún dentro de un rango muy competitivo.

AdaBoost

- **Accuracies y métricas de performance** permanecen muy bajas (accuracy 0.5383; precision 0.5078; F1-score 0.4779; recall 0.5383), reproducen el colapso observado con KFold-3.
- **Interpretación**: la estrategia de reforzar errores sucesivos sigue siendo demasiado sensible a outliers o grupos con características atípicas. De nuevo, muestra la necesidad de un ajuste drástico de hiperparámetros y preprocesamiento antes de poder extraer valor de AdaBoost en este contexto.

Conclusiones bajo LOGO

1. **Random Forest** mejora ligeramente sus métricas frente a KFold-3, consolidándose como la técnica más fiable cuando se evalúan grupos completos.
2. **Voting** conserva un desempeño muy cercano al de Random Forest, aunque pierde la leve ventaja que tenía en KFold-3.
3. **AdaBoost** vuelve a presentar un desempeño inaceptable, lo que refuerza la conclusión de que no es adecuado sin un rediseño de su configuración.

Este análisis muestra que, bajo LeaveOneGroupOut, el ensamblaje por votación sigue siendo una alternativa válida, pero el Random Forest aislado se alza como el método más sólido para datos con agrupamientos marcados.

Evaluación estadística de los métodos de ensamblaje (Random Forest, AdaBoost y Voting)

Con el objetivo de analizar si las diferencias observadas en la métrica de *accuracy* entre los métodos de ensamblaje (Random Forest, AdaBoost y Voting) son estadísticamente significativas, se aplicó el **test no paramétrico de Friedman**, considerando en conjunto los resultados obtenidos bajo los dos esquemas de validación implementados: **GroupKFold (3 pliegues)** y **LeaveOneGroupOut (LOGO)**.

- **Estadístico de Friedman ($\chi^2_{(2)}$):** 3.0000
- **p-valor:** 0.2231

Interpretación

La hipótesis nula del test de Friedman sostiene que los tres métodos de ensamblaje evaluados presentan el mismo rendimiento medio en términos de exactitud. En este caso, el valor p obtenido (0.2231) es **mayor** que el nivel de significancia convencional ($\alpha = 0.05$), lo que impide rechazar la hipótesis nula.

Esto significa que, con un nivel de confianza del 95 %, **no existe evidencia estadísticamente significativa** para afirmar que alguno de los métodos de ensamblaje (Random Forest, AdaBoost o Voting) tenga un rendimiento significativamente superior o inferior a los demás en cuanto a

accuracy, al considerar tanto la validación cruzada por grupos como la validación de tipo LeaveOneGroupOut.

Aunque se observaron tendencias claras en los resultados (por ejemplo, Voting y Random Forest superando a AdaBoost en la mayoría de los casos), desde el punto de vista estadístico, estas diferencias **no pueden considerarse concluyentes** sin un tamaño de muestra mayor o un análisis complementario post-hoc.

3.3 Base de Datos: MHEALTH Dataset

El conjunto de datos recolectado comprende grabaciones de movimiento corporal y signos vitales de diez voluntarios con perfiles diversos mientras realizaban 12 actividades físicas. Los sensores fueron colocados en el ***pecho, muñeca derecha y tobillo izquierdo*** del sujeto.

El uso de múltiples sensores permite medir el movimiento experimentado por diversas partes del cuerpo, como la aceleración, la tasa de giro y la orientación del campo magnético, capturando de mejor forma la dinámica del cuerpo. Todas las modalidades de sensores se grabaron a una tasa de muestreo de 50 Hz, lo cual se considera suficiente para capturar la actividad humana.

Conjunto de Actividades

El conjunto de actividades es el siguiente:

- L1: Estar de pie (1 min)
- L2: Sentado y relajado (1 min)
- L3: Acostado (1 min)
- L4: Caminar (1 min)
- L5: Subir escaleras (1 min)
- L6: Flexiones de cintura hacia adelante (20x)
- L7: Elevación frontal de los brazos (20x)
- L8: Flexión de rodillas (agacharse) (20x)
- L9: Andar en bicicleta (1 min)
- L10: Trotar (1 min)
- L11: Correr (1 min)
- L12: Saltar hacia adelante y hacia atrás (20x)

3.3.1 Dimensiones

Como explican [11], .shape da el número exacto de muestras (filas) y características (columnas) del DataFrame, lo que permite conocer de inmediato la estructura del conjunto de datos (p. 31). Asimismo, .head() muestra las primeras filas del DataFrame, y al pasar un número como argumento, se puede especificar cuántas filas visualizar, facilitando la inspección inicial de los datos (pp. 14-15).

Ya teniendo toda la información almacenada en un DataFrame, se validaron sus dimensiones utilizando .shape para asegurarse de que el número de filas y columnas coincidiera con lo esperado. Posteriormente, se utilizaron las primeras 5 filas del DataFrame con .head() para revisar su contenido y confirmar que los datos se habían cargado correctamente y que la estructura era la adecuada para continuar con el análisis**.**

```
Dimensiones del DataFrame combinado: (1215745, 24)
Primeras 5 filas:
```

	acc_chest_x	acc_chest_y	acc_chest_z	ecg_lead1	ecg_lead2	acc_ankle_x	acc_ankle_y	acc_ankle_z	gyro_ankle_x	gyro_ankle_y	...
0	-9.8184	0.009971	0.29563	0.004186	0.004186	2.1849	-9.6967	0.63077	0.103900	-0.84053	...
1	-9.8489	0.524040	0.37348	0.004186	0.016745	2.3876	-9.5080	0.68389	0.085343	-0.83865	...
2	-9.6602	0.181850	0.43742	0.016745	0.037677	2.4086	-9.5674	0.68113	0.085343	-0.83865	...
3	-9.6507	0.214220	0.24033	0.079540	0.117220	2.1814	-9.4301	0.55031	0.085343	-0.83865	...
4	-9.7030	0.303890	0.31156	0.221870	0.205130	2.4173	-9.3889	0.71098	0.085343	-0.83865	...

5 rows × 24 columns

Teniendo como resultado un total de ***1,215,745 filas y 24 columnas***, las cuales ya serán descritas e interpretadas más adelante.

Posterior a ello, se procedió con la identificación de datos ausentes, ya que según McKinney (2017), es crucial asegurarse de que los valores faltantes se identifiquen correctamente antes de proceder con cualquier análisis, ya que los datos incompletos pueden afectar la precisión y validez de los resultados.

```
Valores faltantes por columna:
acc_chest_x     0
acc_chest_y     0
acc_chest_z     0
ecg_lead1       0
ecg_lead2       0
acc_ankle_x     0
acc_ankle_y     0
acc_ankle_z     0
gyro_ankle_x    0
gyro_ankle_y    0
gyro_ankle_z    0
mag_ankle_x     0
mag_ankle_y     0
mag_ankle_z     0
acc_wrist_x     0
acc_wrist_y     0
acc_wrist_z     0
gyro_wrist_x    0
gyro_wrist_y    0
gyro_wrist_z    0
mag_wrist_x     0
mag_wrist_y     0
mag_wrist_z     0
num_activity    0
dtype: int64
```

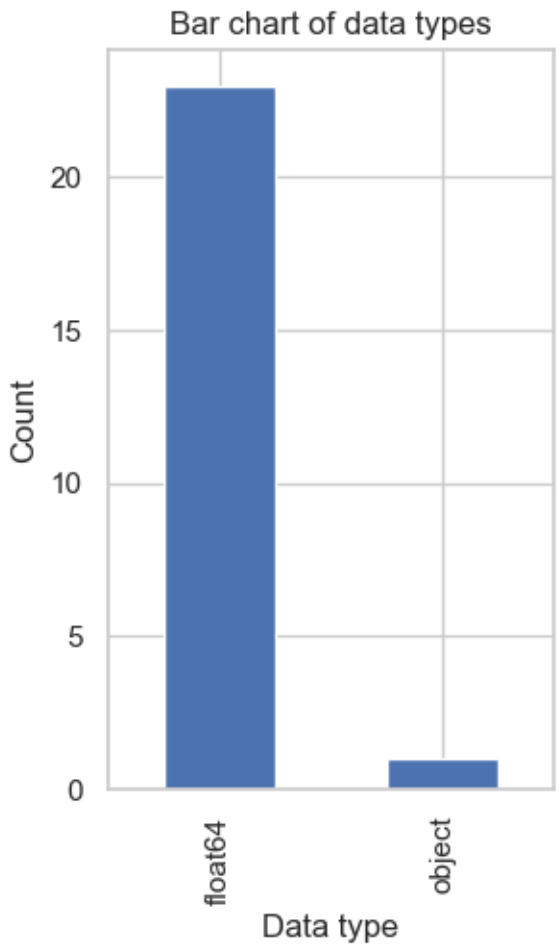
```
¿Existen valores faltantes en el dataset?
False
```

En este caso, no se encontraron valores faltantes.

Asimismo, dado que la clase nula (0), no representa algo representativo de alguna actividad, más que la ausencia o reposo de alguna, se optó por eliminar dicha clase. Lo que redujo significativamente las dimensiones del dataset, con un total de ***343, 195 filas y 24 columnas***.

3.3.2 Atributos, distribución y resúmenes estadísticos

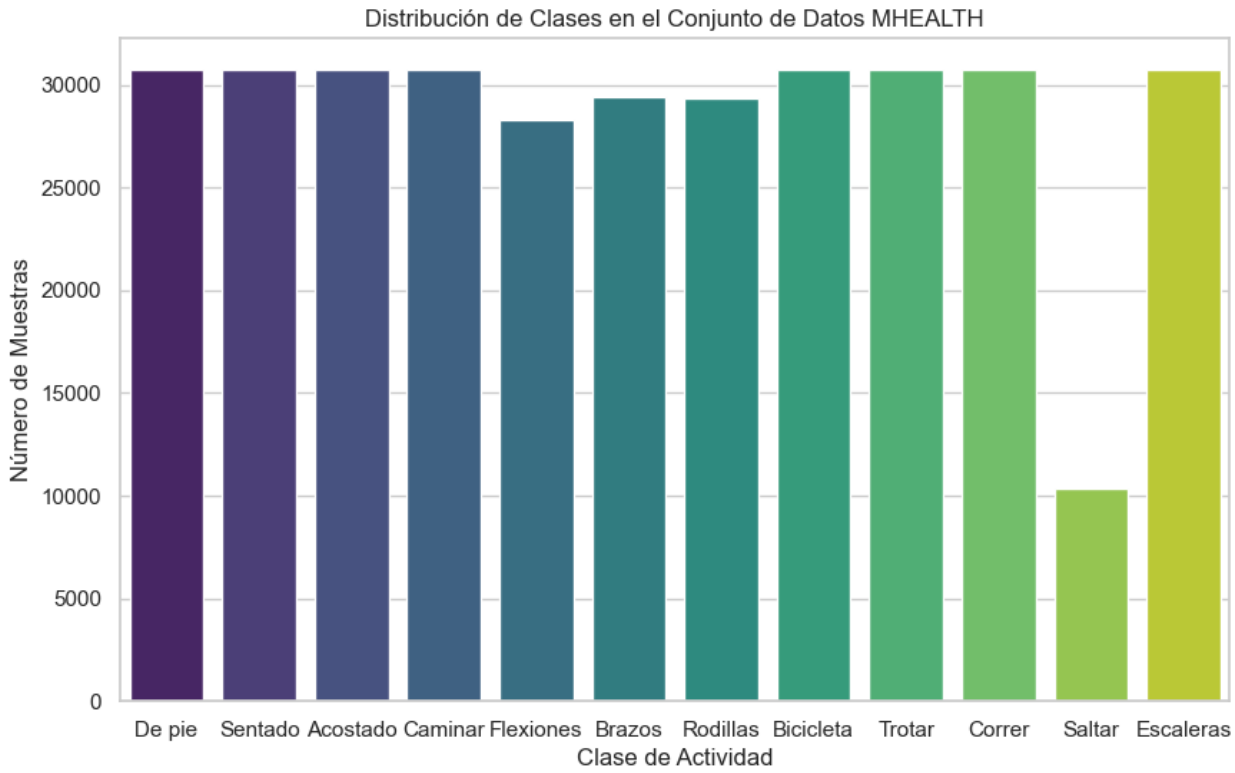
En cuanto a los tipos de datos presentes en este dataset, en la gráfica siguiente se muestra que todos son de tipo flotante (64), algo que resulta adecuado para evitar errores en el análisis y mejorar el rendimiento computacional [12].



El conjunto de datos empleado en este estudio presenta una amplia variedad de actividades humanas registradas, abarcando tanto movimientos estáticos como dinámicos, así como ejercicios localizados. La variable objetivo activity_name comprende un total de 12 clases distintas, con una distribución equilibrada en las actividades principales y ligera variabilidad en las actividades complementarias.

Las clases “De pie”, “Sentado”, “Acostado”, “Caminar”, “Bicicleta”, “Trotar”, “Correr” y “Escaleras” contienen 30,720 registros cada una, lo que indica una recolección balanceada y sistemática para estas actividades base. Estas representan tanto posturas estáticas como movimientos locomotores de baja y alta intensidad, lo cual es ideal para entrenar modelos generalizables.

Por otro lado, las clases “Brazos” (29,441), “Rodillas” (29,337), “Flexiones” (28,315) y “Saltar” (10,342) presentan una menor cantidad de instancias. En particular, la clase “Saltar” tiene una frecuencia marcadamente menor, lo cual puede representar un reto para los clasificadores al momento de identificar patrones representativos de esta categoría.



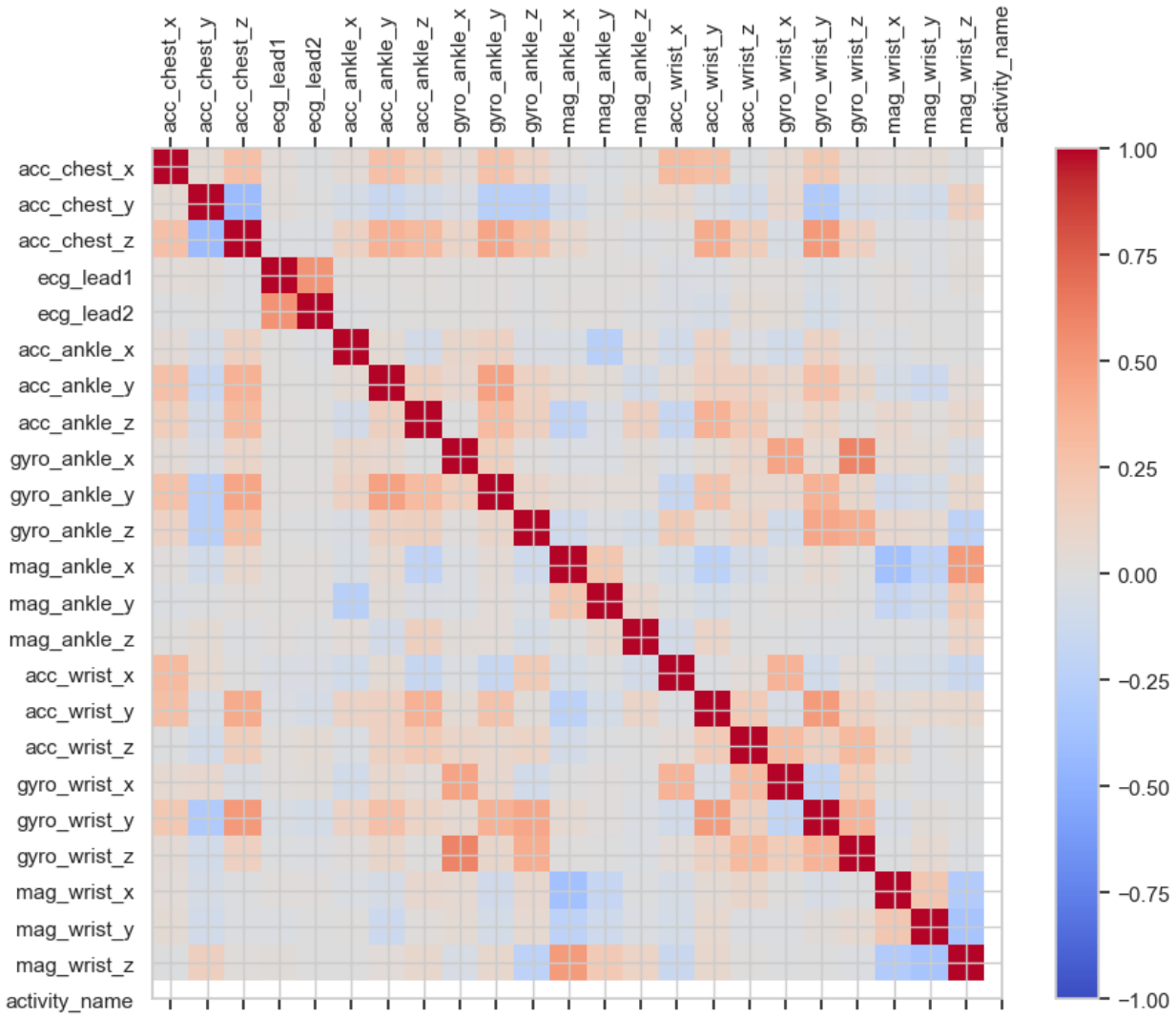
Más adelante se trabajará con esta información para tomar decisiones y poder observar la diferencia de desempeño de los clasificadores con un desbalance considerable respecto con aquellos clasificadores entrenados bajo un conjunto de datos con puntos sintéticos para las clases minoritarias con la ayuda del SMOTE. [13].

Correlación entre atributos

En el conjunto de datos, se observa una alta correlación entre los sensores de acelerómetro y giroscopio ubicados en las mismas partes del cuerpo. Por ejemplo, el acelerómetro y giroscopio del tobillo (como acc_ankle_y y gyro_ankle_y) tienen correlaciones destacadas, lo cual indica que los movimientos en esa zona tienden a estar coordinados y son consistentes. Lo mismo ocurre con los sensores de la muñeca, en donde acc_wrist_y muestra una correlación fuerte con gyro_wrist_y.

También se identifica que las lecturas de acelerómetros en diferentes partes del cuerpo están relacionadas, especialmente entre el pecho, el tobillo y la muñeca. Por ejemplo, acc_chest_x está moderadamente correlacionado con acc_ankle_y y acc_wrist_x, lo cual sugiere que ciertos movimientos del torso están asociados con reacciones en las extremidades.

En contraste, las señales de ECG (ecg_lead1 y ecg_lead2) muestran muy baja correlación con los sensores de movimiento (acelerómetros, giroscopios y magnetómetros), lo cual es esperado, ya que estas mediciones corresponden a señales fisiológicas internas del corazón y no a movimiento externo. Esto refuerza su valor como fuente independiente de información para modelos de salud.



Resumen Estadístico

El conjunto de datos tiene 343,195 registros por variable, lo que permite una base sólida para análisis. Los acelerómetros del pecho muestran medias negativas, especialmente en el eje X (-7.48), lo que podría indicar inclinación corporal o efecto de la gravedad. La variabilidad en los tres ejes es amplia, reflejando movimientos diversos.

Las señales de ECG están centradas cerca de cero, con amplitudes de hasta ±8.5, lo cual es típico en datos crudos de electrocardiograma. La desviación estándar baja sugiere que la mayor parte de los valores están cerca de la media, aunque hay picos notables.

En el tobillo, el acelerómetro en el eje Y tiene una media cercana a -9, consistente con la dirección de la gravedad. Los valores extremos y la alta desviación estándar reflejan movimientos intensos. Los giroscopios del tobillo tienen variaciones menores, como es esperable al medir velocidad angular.

Finalmente, en la muñeca, los acelerómetros muestran movimientos intensos, con valores extremos y alta variabilidad. Los giroscopios también reflejan actividad considerable. Los magnetómetros, al igual que en el tobillo, presentan datos ruidosos que podrían requerir normalización.

3.1.3 Escalamiento

Después de realizar un análisis a los datos se concluyó que hacer una estandarización es lo mejor.

En primer lugar, los datos contemplan diferentes escalas. Algunos valores varían en un rango estrecho, mientras que otros tienen dispersión más grande (ej., aceleración, giroscopio).

A su vez, sensores como acelerómetros y giroscopios tienen unidades distintas (m/s^2 vs. rad/s), lo que puede afectar modelos como SVM o regresión logística. Algoritmos como KNN, SVM y Redes Neuronales se ven afectados por escalas diferentes [7].

En otros casos se recomendaría una normalización en el caso de usar redes neuronales, pero, ya que en la gráfica de caja y bigotes observamos que hay valores muchos valores atípicos (colas largas), Como la normalización depende de $Xmin$ y $Xmax$, estos valores extremos pueden comprimir demasiado los valores intermedios.

Por lo tanto, dado que los datos tienen diferentes escalas, outliers y correlaciones mixtas, la mejor transformación a aplicar es la estandarización para optimizar el rendimiento en algoritmos de Machine Learning.

3.1.4 Segmentación en ventanas y extracción de características

De acuerdo con la evidencia científica reportada en la literatura especializada, donde se ha establecido que una ventana temporal de 3 segundos sin solapamiento resulta suficiente para el reconocimiento de actividades humanas (Banos et al., 2014), se procedió a la segmentación del conjunto de datos como etapa preliminar para el proceso de extracción de características. Cabe destacar que la frecuencia de muestreo del sistema de adquisición empleado en el dataset asciende a 50 Hz, lo que implica una captura de cincuenta muestras por unidad de tiempo (segundo) para cada uno de los tres ejes ortogonales en cada sensor inercial.

Para la extracción de características, se seleccionaron aquellas que han demostrado ser efectivas en estudios previos (Dernbach et al., 2012; Zhang y Sawchuk, 2012). En el caso de los **acelerómetros y giroscopios**, se obtuvieron 16 características por sensor. Estas son:

- **Valor medio (mean)** de cada eje (X, Y, Z): representa la aceleración o rotación promedio en cada dirección, lo cual puede indicar la orientación o tendencia general del movimiento.
- **Desviación estándar (std)** de cada eje: cuantifica la variabilidad o dispersión del movimiento respecto a la media; valores altos indican movimientos irregulares o bruscos.
- **Valor máximo (max)** de cada eje: refleja el pico de actividad en esa dirección, útil para detectar movimientos intensos o eventos específicos.
- **Correlación entre pares de ejes** (X–Y, X–Z, Y–Z): mide cómo se relacionan los movimientos entre ejes; por ejemplo, una alta correlación entre X e Y puede indicar un movimiento diagonal.
- **Magnitud media**: se calcula como la raíz cuadrada de la suma de los cuadrados de los tres ejes ($\sqrt{x^2 + y^2 + z^2}$), y representa la intensidad global del movimiento.
- **Desviación estándar de la magnitud**: muestra la variación de la intensidad del movimiento a lo largo del tiempo.
- **Área bajo la curva (AUC) de la magnitud**: representa la suma acumulada de la magnitud a lo largo del tiempo, útil para medir la duración y fuerza total del movimiento.
- **Diferencia media de magnitud entre lecturas consecutivas**: cuantifica los cambios bruscos entre momentos contiguos, capturando transiciones rápidas o movimientos súbitos.

En cuanto al **magnetómetro triaxial**, se siguió el enfoque propuesto por Altun y Barshan (2010), extrayendo 6 características clave:

- **Valor mínimo y máximo**: indican el rango de intensidad del campo magnético registrado, lo cual puede estar influido por la orientación del dispositivo y la presencia de metales.
- **Valor medio**: representa el nivel promedio del campo magnético en cada eje.
- **Varianza**: mide la dispersión de los datos alrededor de la media, similar a la desviación estándar, pero sin aplicar raíz cuadrada; útil para evaluar estabilidad o fluctuaciones.
- **Asimetría (skewness)**: indica si los datos están sesgados hacia la derecha o izquierda; por ejemplo, una asimetría positiva sugiere que hay más valores bajos y algunos valores muy altos.
- **Curtosis (kurtosis)**: mide la concentración de los datos alrededor de la media y la presencia de valores extremos; una curtosis alta indica picos agudos y colas largas.

Lo que resulta en un conjunto de datos final con **116 columnas, y una más para el tipo de actividad**.

	acc_chest_x_mean	acc_chest_x_std	acc_chest_x_max	acc_chest_y_mean	acc_chest_y_std	acc_chest_y_max	acc_chest_z_mean
0	-9.730361	0.143112	-9.1065	0.208722	0.136345	0.69219	0.845242
1	-9.745109	0.128526	-9.3312	0.215160	0.150341	0.60054	0.826381
2	-9.756214	0.124798	-9.3337	0.248450	0.138132	0.75182	0.806151
3	-9.738831	0.125749	-9.4184	0.298682	0.135002	0.66073	0.756566
4	-9.740250	0.136765	-9.2256	0.332279	0.137650	0.81272	0.809624

5 rows × 117 columns

El procesamiento de los datos se realizó mediante la segmentación en ventanas de tamaño fijo, lo que permite capturar patrones temporales y variaciones en las señales. El número total de muestras registradas fue de 343,195, lo que dio lugar a 2,287 ventanas completas, cada una de ellas representando una secuencia de 150 muestras (50 muestras por cada uno de los tres ejes).

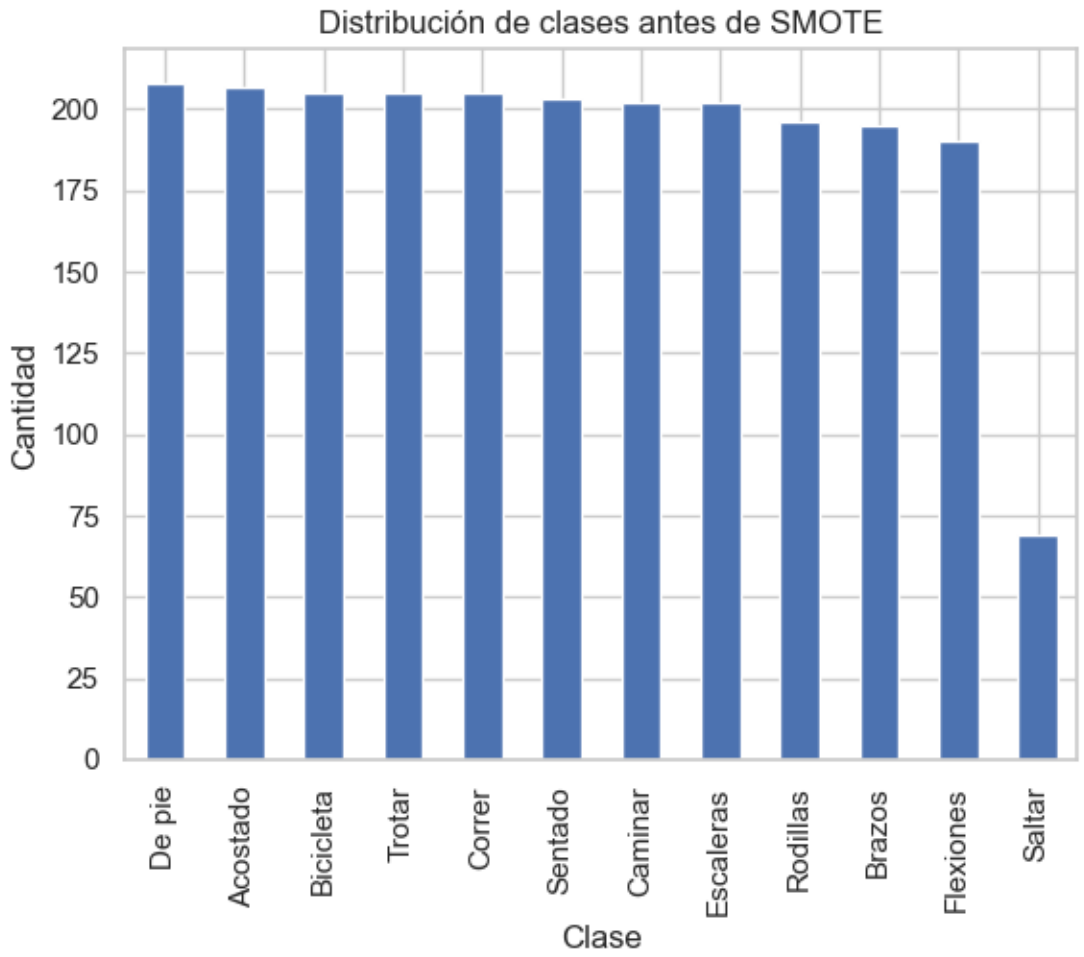
```
df_features.shape
```

```
(2287, 117)
```

En cuanto a la distribución de muestras por actividad, muestra un conjunto relativamente equilibrado entre la mayoría de las clases, lo cual es beneficioso para el entrenamiento del modelo, ya que reduce el riesgo de sesgo hacia actividades más representadas. Actividades como "De pie", "Acostado", "Bicicleta", "Trotar", "Correr" y "Sentado" cuentan con un número similar de instancias (alrededor de 200), asegurando una representación adecuada en el conjunto de datos. Sin embargo, se observa que actividades como "Flexiones", "Rodillas", "Brazos" y especialmente "Saltar" tienen menos registros, siendo esta última notablemente menor con solo 69 muestras.

3.1.5 Conjuntos de entrenamiento y de prueba

Se realizó una partición del conjunto de datos en subconjuntos de entrenamiento y prueba. Inicialmente, se separaron las características (X) y las etiquetas (y). Para analizar la distribución de clases, se visualizó un gráfico de barras donde se observó un desbalance considerable entre las actividades, destacando especialmente la baja representación de la clase *Saltar*.



Para abordar este problema, se aplicó la técnica de sobremuestreo **SMOTE (Synthetic Minority Over-sampling Technique) [13]**. Esta se integró directamente en un pipeline junto con el preprocesamiento y el clasificador correspondiente. Utilizar pipelines permite encapsular todo el flujo de trabajo (escalado, generación de muestras sintéticas y entrenamiento del modelo).

En particular, el pipeline asegura que:

1. La estandarización (StandardScaler) se entrena solo con los datos del conjunto de entrenamiento en cada fold, sin exponer información del conjunto de prueba.

2. SMOTE se aplica únicamente sobre los datos de entrenamiento, garantizando que el modelo no vea instancias sintéticas durante la evaluación.

De esta forma, el pipeline garantiza que todas las transformaciones ocurran dentro del proceso de validación cruzada, simulando lo que ocurriría en un entorno de producción. Esto mejora la validez de las métricas de desempeño y previene resultados inflados artificialmente.

```
print(y_train_resampled.value_counts())
```

```
activity_name
Correr       146
De pie       146
Rodillas     146
Flexiones    146
Brazos       146
Caminar      146
Bicicleta    146
Saltar       146
Escaleras    146
Acostado     146
Trotar       146
Sentado      146
```

La división de los datos se realizó utilizando un **split estratificado del 70/30**, lo que significa que se preservó la proporción original de clases en los conjuntos (Introduction to Machine Learning with Python).

3.1.6 Entrenamiento de clasificadores y reducción de características

En este proceso, entrenaremos tres clasificadores distintos (Regresión Logística, Bosques Aleatorios y Perceptrón) utilizando la base de datos de entrenamiento. Posteriormente, utilizaremos la base de datos de prueba para medir el desempeño de cada clasificador entrenado en términos de exactitud.

A continuación, se presentan los resultados de exactitud (accuracy) para cada modelo, evaluados con datos de prueba y con validación cruzada de 5 y 10 particiones:

1. Regresión Logística

- **Exactitud en entrenamiento:** 1.000
- **Exactitud en prueba:** 0.9738
- **Cross-validation (5 folds):** media ≈ 0.9817
- **Cross-validation (10 folds):** media ≈ 0.9823

2. Random Forest

- **Exactitud en entrenamiento:** 1.000
- **Exactitud en prueba:** 0.9811
- **Cross-validation (5 folds):** media ≈ 0.9829
- **Cross-validation (10 folds):** media ≈ 0.9817

3. Perceptrón

- **Exactitud en entrenamiento:** 0.9954
- **Exactitud en prueba:** 0.9680
- **Cross-validation (5 folds):** media ≈ 0.9777
- **Cross-validation (10 folds):** media ≈ 0.9732

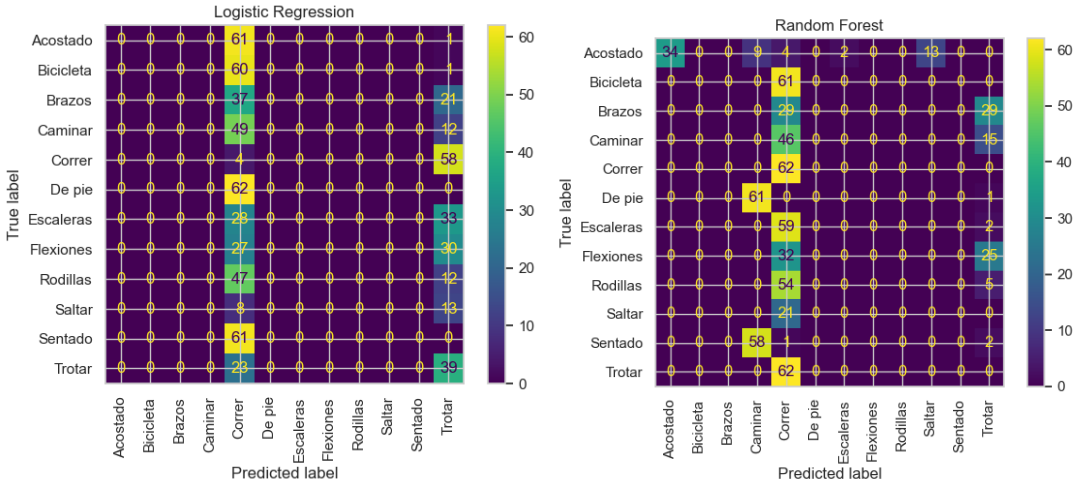
Evaluación con StratifiedKFold

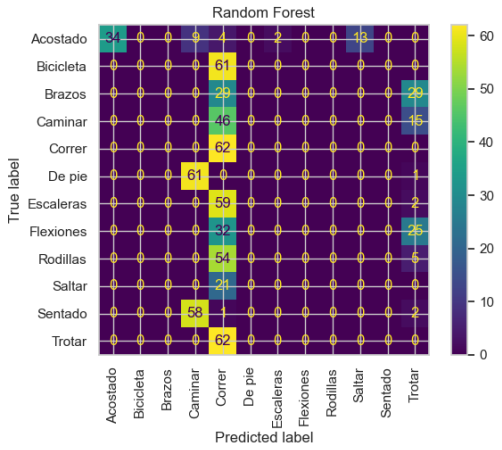
Para tener una evaluación más robusta frente a desequilibrios en la distribución de clases, se aplicó **validación cruzada estratificada.** Esto garantiza que cada partición de entrenamiento y prueba mantenga la proporción de clases original, lo cual es especialmente relevante cuando existen clases subrepresentadas.

Los resultados fueron:

- **Regresión Logística:** 0.9803 ± 0.0050
- **Random Forest**: 0.9851 ± 0.0021
- **Perceptrón:** 0.9707 ± 0.0091

En cuanto a los resultados, podemos decir que Random Forest fue el mejor modelo en cuanto a exactitud y estabilidad.





Reducción de Características

Se utilizaron tres métodos para reducir dimensionalidad y optimizar modelos:

1. **Análisis de Componentes Principales (PCA)**: Al retener el 95% de la varianza con 48 componentes (vs. 116 originales), como señala Aljarrah y Ali (2019), este umbral del 95% es un estándar para equilibrar simplificación y preservación de información.
2. **Kernel PCA (RBF)**: Se aplicó la técnica de Kernel Principal Component Analysis (Kernel PCA) sobre los datos escalados de entrenamiento para reducir la dimensionalidad a 47 componentes. El parámetro kernel="rbf" se utilizó para aplicar un kernel Radial Basis Function (RBF), que es adecuado para capturar relaciones no lineales entre las características.
3. **Sequential Forward Selection (SFS)**: SFS comienza con un conjunto vacío y añade progresivamente la característica que aporta mayor mejora en la métrica de interés (Zhang y Sawchuk, 2011), hasta que no se observe una mejora significativa o se alcance un número deseado de características.
4. **Sin reducción de características**.

3.1.7 Métricas de desempeño

En el contexto de la evaluación de los clasificadores, se emplea la validación cruzada con 2-fold, 5-fold, y 10-fold, tomando como referencia el estudio de Hsu et al. (2018) [19]. El uso de diferentes estrategias de validación cruzada permite obtener una evaluación más exhaustiva del modelo, minimizando la varianza de los resultados y mejorando la capacidad de generalización del modelo al no depender de una única partición del conjunto de datos.

Exactitud/Accuracy

Se calcula como la proporción de predicciones correctas sobre el total de muestras. Es una medida de la consistencia entre las predicciones del modelo y los resultados reales.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Donde:

- TP (True Positives): Casos positivos correctamente clasificados.
- TN (True Negatives): Casos negativos correctamente clasificados.
- FP (False Positives): Casos negativos incorrectamente clasificados como positivos.
- FN (False Negatives): Casos positivos incorrectamente clasificados como negativos.

Precisión

Indica la proporción de predicciones positivas que son realmente correctas, y es relevante cuando los costos de los falsos positivos son elevados.

$$Precision = \frac{TP}{TP + FP}$$

Recall/Sensibilidad

Mide la capacidad del modelo para identificar correctamente los casos positivos, siendo crucial cuando los falsos negativos son costosos. Es una medida de cuántos de los verdaderos casos positivos han sido capturados por el modelo.

$$Recall = \frac{TP}{TP + FN}$$

F1-Score

Es la media armónica entre la precisión y el recall, proporcionando una medida balanceada entre ambas, especialmente útil cuando las clases están desbalanceadas.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Las métricas utilizadas para evaluar los clasificadores incluyen precisión, recall, F1-score y exactitud. Estas métricas fueron definidas en el trabajo [19], quienes explican que el F1-score, la precisión y el recall son fundamentales para la evaluación del desempeño de los clasificadores en tareas de reconocimiento de actividades.

Métricas de Regresión Logística

Dataset	CV (folds)	Accuracy	Precision	F1-score	Recall
Original	2	0.971461	0.972611	0.971330	0.971461
Original	5	0.981724	0.982367	0.981604	0.981705
Original	10	0.981727	0.982919	0.981684	0.981865
PCA	2	0.965753	0.967290	0.965569	0.965753
PCA	5	0.978873	0.979416	0.978752	0.978851
PCA	10	0.980016	0.981346	0.979929	0.980079
Kernel PCA	2	0.640982	0.814731	0.677735	0.640982
Kernel PCA	5	0.655845	0.833040	0.691752	0.656264
Kernel PCA	10	0.651834	0.834369	0.689229	0.652381
SFS	2	0.968607	0.969838	0.968437	0.968607
SFS	5	0.978305	0.978975	0.978228	0.978314
SFS	10	0.978880	0.980074	0.978651	0.978849

Se observa una mejora constante en el desempeño del modelo a medida que aumenta el número de particiones en la validación cruzada, lo que indica una mayor estabilidad y capacidad de generalización. En particular, la diferencia aproximada de un punto porcentual entre 2 folds y 5 o 10 folds refleja que una mayor cantidad de particiones mejora la representatividad de los datos en cada iteración, mientras que la similitud entre 5 y 10 folds sugiere que más particiones adicionales no aportan mejoras significativas.

Al comparar las técnicas de reducción y selección de características, los modelos entrenados con los datos originales y con selección secuencial de características (SFS) alcanzan el mejor desempeño. Esto puede explicarse porque SFS selecciona las características más relevantes de forma explícita, optimizando la información utilizada por el modelo sin perder variables significativas, mientras que trabajar con los datos originales mantiene toda la información disponible. Por otro lado, PCA logra una reducción dimensional efectiva, preservando la mayor parte de la varianza en menos componentes, lo que explica su buen desempeño aunque ligeramente inferior. En contraste, Kernel PCA presenta resultados inferiores posiblemente debido a que, si bien es capaz de capturar relaciones no lineales, su aplicación requiere una adecuada elección de kernel y parámetros; si estos no están óptimamente ajustados o si la complejidad introducida no se alinea con la estructura real de los datos, puede generar representaciones menos efectivas para el clasificador, impactando negativamente en el rendimiento.

Métricas de Bosques Aleatorios

Dataset	CV (folds)	Accuracy	Precision	F1-score	Recall
Original	2	0.981164	0.981709	0.981143	0.981164
Original	5	0.982869	0.983557	0.982837	0.982835
Original	10	0.981159	0.982387	0.981053	0.981190
PCA	2	0.962329	0.964337	0.962141	0.962329
PCA	5	0.972593	0.973261	0.972496	0.972586
PCA	10	0.972023	0.973669	0.971953	0.972063
Kernel PCA	2	0.868721	0.883556	0.871689	0.868721
Kernel PCA	5	0.883578	0.893270	0.885455	0.883544
Kernel PCA	10	0.889838	0.900841	0.891176	0.889365
SFS	2	0.983447	0.983773	0.983394	0.983447
SFS	5	0.982873	0.983513	0.982858	0.982835
SFS	10	0.982302	0.983296	0.982176	0.982341

Se observa que la validación con 5 folds ofrece el mejor desempeño en todas las métricas, con un Accuracy de 98.29%, Precision de 98.36%, F1-score de 98.29% y Recall de 98.28%.

Al comparar los diferentes métodos de reducción y selección de características, se aprecia que el modelo entrenado con características seleccionadas mediante SFS alcanza el rendimiento más alto en validación con 2 folds (Accuracy 98.34%) y mantiene un desempeño competitivo en las demás configuraciones. Los datos originales sin reducción también presentan resultados muy cercanos, especialmente en validaciones con 5 y 10 folds.

Por otro lado, PCA muestra un desempeño algo inferior, con Accuracy alrededor de 97.2% a 97.3% para 5 y 10 folds, mientras que Kernel PCA evidencia un rendimiento significativamente menor en todas las configuraciones, con Accuracy que oscila entre 86.8% y 88.9%, lo que sugiere que esta técnica no mejora ni preserva las características relevantes para el modelo en este caso.

Métricas de Perceptrón

Dataset	CV (folds)	Accuracy	Precision	F1-score	Recall
Original	2	0.966324	0.967630	0.966307	0.966324
Original	5	0.972604	0.974268	0.972557	0.972625
Original	10	0.973731	0.975579	0.973598	0.973770
PCA	2	0.944064	0.945445	0.944045	0.944064
PCA	5	0.952057	0.956098	0.951442	0.952031
PCA	10	0.958315	0.961299	0.957556	0.957817
Kernel PCA	2	0.545091	0.757180	0.577046	0.545091
Kernel PCA	5	0.557656	0.718337	0.560244	0.557816
Kernel PCA	10	0.570201	0.728318	0.574389	0.569643
SFS	2	0.948630	0.951367	0.948606	0.948630
SFS	5	0.968594	0.970852	0.968582	0.968525
SFS	10	0.968045	0.970560	0.968065	0.968095

Para el conjunto de datos original, el Perceptrón alcanza una precisión, recall y F1-score superiores al 96% en todos los escenarios, con un aumento paulatino que llega hasta aproximadamente 97.4% en precisión y 97.3% en F1-score al usar 10 folds. En cuanto a la reducción de dimensionalidad con PCA, el rendimiento es ligeramente inferior al conjunto original, aunque mantiene una tendencia similar de mejora al aumentar los folds. Las métricas alcanzan valores cercanos al 95-96% en precisión y F1-score con 10 folds.

Por otro lado, el Kernel PCA presenta un rendimiento significativamente más bajo en todas las métricas, con valores alrededor del 54-57% en exactitud y con una diferencia notable respecto a las otras técnicas.

Finalmente, la selección secuencial de características (SFS) muestra resultados muy cercanos a los del conjunto original, con métricas en el rango del 94-97%, lo que confirma la eficacia de esta técnica para mejorar la interpretabilidad sin sacrificar el desempeño.

3.1.8 Método estadístico para la comparación de clasificadores

Para realizar una evaluación objetiva del rendimiento de los clasificadores empleados en este trabajo, es esencial aplicar métodos estadísticos que permitan determinar las diferencias observadas en las métricas de desempeño son estadísticamente significativas.

Elección del método estadístico

Debido a que en este trabajo se comparan más de dos clasificadores sobre un mismo conjunto de datos utilizando validación cruzada, el método más adecuado es el **Test de Friedman**, seguido de un análisis **Post-hoc de Nemenyi**. Esta elección se fundamenta por las siguientes razones:

El test de Friedman es un método no paramétrico utilizado para detectar diferencias entre varios algoritmos que han sido evaluados sobre múltiples particiones de un dataset. A diferencia de otros

análisis, este test no asume ninguna normalidad ni homogeneidad de varianzas, lo cual es útil cuando se trabaja con métricas de clasificación que no siguen una distribución normal o casi siempre.

De ser así, cuando el test de friedman indica la existencia de diferencias estadísticas significativas entre los clasificadores, se aplica un análisis Nemenyi. Para compara todos los clasificadores entre sí por pares e identificar cuales algoritmos presentan diferencias significativas.

Este enfoque ha sido utilizado y recomendado por Demšar (2006) sobre el uso del test de friedman en conjunto con el análisis post-hoc de Nemenyi. Y en particular, se recomienda el uso de pruebas paramétricas para comparar el rendimiento entre múltiples clasificadores de datos HAR dado que las distribuciones de métricas como la exactitud de o la F1 no son siempre normales ni homogéneas.

Regresión Logística

```
--- Accuracy ---
Friedman statistic: 11.5532, p-value: 0.0091
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.961371  0.011804  0.532733
1  0.961371  1.000000  0.049612  0.826788
2  0.011804  0.049612  1.000000  0.315940
3  0.532733  0.826788  0.315940  1.000000

--- Precision ---
Friedman statistic: 12.1837, p-value: 0.0068
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.994838  0.011804  0.383051
1  0.994838  1.000000  0.025015  0.532733
2  0.011804  0.025015  1.000000  0.455933
3  0.383051  0.532733  0.455933  1.000000
```

```
--- F1-score ---
Friedman statistic: 12.1837, p-value: 0.0068
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.994838  0.011804  0.383051
1  0.994838  1.000000  0.025015  0.532733
2  0.011804  0.025015  1.000000  0.455933
3  0.383051  0.532733  0.455933  1.000000

--- Recall ---
Friedman statistic: 12.1837, p-value: 0.0068
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.994838  0.011804  0.383051
1  0.994838  1.000000  0.025015  0.532733
2  0.011804  0.025015  1.000000  0.455933
3  0.383051  0.532733  0.455933  1.000000
```

Los resultados del test de Friedman muestran que existen diferencias estadísticamente significativas entre los diferentes datasets evaluados, ya que para todas las métricas consideradas (Accuracy, Precision, F1-score y Recall) el valor p es menor a 0.05. Esto indica que al menos un par de los datasets produce un rendimiento diferente en el modelo de regresión logística bajo validación cruzada estratificada. Es importante aclarar que solo se tomaron en cuenta las de 5-folds [20].

Al realizar el test post-hoc de Nemenyi para identificar específicamente entre qué datasets existen estas diferencias, se observa que el dataset basado en Kernel PCA se comporta de manera significativamente inferior en comparación con los datasets Original y PCA. Los valores p obtenidos para estas comparaciones son menores a 0.05 en todas las métricas, confirmando que el rendimiento del modelo con Kernel PCA es estadísticamente peor que con los otros conjuntos de datos. En contraste, no se encontraron diferencias significativas entre los datasets Original, PCA y SFS, lo que sugiere que sus desempeños son comparables.

Bosques Aleatorios

```
--- Accuracy ---
Friedman statistic: 11.1250, p-value: 0.0111
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.611061  0.017325  0.994838
1  0.611061  1.000000  0.315940  0.760994
2  0.017325  0.315940  1.000000  0.035525
3  0.994838  0.760994  0.035525  1.000000

--- Precision ---
Friedman statistic: 10.6800, p-value: 0.0136
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.611061  0.017325  0.994838
1  0.611061  1.000000  0.315940  0.760994
2  0.017325  0.315940  1.000000  0.035525
3  0.994838  0.760994  0.035525  1.000000
```

```
--- F1-score ---
Friedman statistic: 10.6800, p-value: 0.0136
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.760994  0.035525  0.994838
1  0.760994  1.000000  0.315940  0.611061
2  0.035525  0.315940  1.000000  0.017325
3  0.994838  0.611061  0.017325  1.000000

--- Recall ---
Friedman statistic: 10.8367, p-value: 0.0126
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.688134  0.025015  1.000000
1  0.688134  1.000000  0.315940  0.688134
2  0.025015  0.315940  1.000000  0.025015
3  1.000000  0.688134  0.025015  1.000000
```

Los resultados del test de Friedman aplicado a las métricas de desempeño del clasificador Random Forest muestran diferencias estadísticamente significativas entre los cuatro conjuntos de datos evaluados (Original, PCA, Kernel PCA y SFS). Es importante aclarar que solo se tomaron en cuenta las de 5-folds [20].

Para las cuatro métricas, los resultados señalan que el conjunto basado en Kernel PCA (índice 2) se comporta significativamente peor que los demás. Esto se observa en los valores p muy bajos (menores a 0.05) en las comparaciones entre Kernel PCA y los otros conjuntos (Original, PCA y SFS), lo que sugiere que la reducción dimensional no lineal con Kernel PCA afectó negativamente el desempeño del modelo Random Forest.

Por otro lado, no se observan diferencias significativas entre el conjunto Original, el conjunto PCA y el conjunto SFS. Esto indica que las transformaciones lineales con PCA y la selección secuencial de características mantienen un rendimiento similar al conjunto original.

Perceptrón

```
--- Accuracy ---
Friedman statistic: 14.1250, p-value: 0.0027
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.203470  0.007913  0.994838
1  0.203470  1.000000  0.611061  0.315940
2  0.007913  0.611061  1.000000  0.017325
3  0.994838  0.315940  0.017325  1.000000

--- Precision ---
Friedman statistic: 13.5600, p-value: 0.0036
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.203470  0.007913  0.994838
1  0.203470  1.000000  0.611061  0.315940
2  0.007913  0.611061  1.000000  0.017325
3  0.994838  0.315940  0.017325  1.000000
```

```
--- F1-score ---
Friedman statistic: 13.5600, p-value: 0.0036
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.203470  0.007913  0.994838
1  0.203470  1.000000  0.611061  0.315940
2  0.007913  0.611061  1.000000  0.017325
3  0.994838  0.315940  0.017325  1.000000

--- Recall ---
Friedman statistic: 13.5600, p-value: 0.0036
Nemenyi post-hoc test:
          0         1         2         3
0  1.000000  0.203470  0.007913  0.994838
1  0.203470  1.000000  0.611061  0.315940
2  0.007913  0.611061  1.000000  0.017325
3  0.994838  0.315940  0.017325  1.000000
```

Los resultados indicaron diferencias significativas en las cuatro métricas evaluadas, con valores p menores a 0.05, confirmando que no todos los métodos de preprocesamiento ofrecen un rendimiento equivalente para el Perceptron.

Posteriormente, se aplicó la prueba post-hoc de Nemenyi para identificar entre qué pares de datasets se encontraban dichas diferencias. Esta prueba reveló que el conjunto Kernel PCA difiere significativamente del resto (Original, PCA y SFS) en todas las métricas analizadas. Sin embargo, no se encontraron diferencias significativas entre los datasets Original, PCA y SFS, lo que sugiere que estos tres métodos proporcionan un rendimiento comparable para el clasificador Perceptron.

3.1.9 Estrategias de fusión

Para evaluar distintas estrategias de combinación de clasificadores, se definieron tres algoritmos base seleccionados en función de los mejores resultados obtenidos en los experimentos previos, considerando las diferencias estadísticamente significativas. Tanto la Regresión Logística como el Random Forest se utilizaron con características reducidas mediante PCA y validación cruzada de 5 folds, ya que esta combinación mostró un desempeño estadísticamente competitivo frente a otras configuraciones analizadas. Por su parte, el Naive Bayes Gaussiano se empleó sin reducción de características, también con validación cruzada de 5 folds.

A partir de estos clasificadores base, se aplicaron tres estrategias de fusión. La primera consistió en utilizar un clasificador Random Forest de forma individual como método de agregación. La segunda estrategia implementó un modelo VotingClassifier con votación dura, combinando los tres clasificadores mencionados. Finalmente, se aplicó el algoritmo AdaBoost, utilizando árboles de decisión simples (decision stumps) como clasificadores base.

Para analizar si las diferencias observadas en las métricas de exactitud eran estadísticamente significativas, se aplicó el test no paramétrico de Friedman. Como complemento, se aplicó una prueba post-hoc de Nemenyi para identificar cuáles pares de clasificadores presentaban diferencias significativas específicas [22].

Análisis de Resultados

CV (folds)	Random Forest Accuracy	Voting Accuracy	AdaBoost Accuracy	Random Forest Precision	Voting Precision	AdaBoost Precision	Random Forest F1	Voting F1	AdaBoost F1	Random Forest Recall	Voting Recall	AdaBoost Recall
5	0.973167	0.9783	0.223764	0.97417	0.979022	0.117442	0.973166	0.978223	0.127844	0.973218	0.978295	0.224943

Los resultados mostraron que la estrategia basada en VotingClassifier fue la más efectiva, alcanzando una exactitud de 97.83%, así como la mayor precisión, F1 y recall entre todas las opciones evaluadas. Al combinar algoritmos lineales, probabilísticos y basados en árboles, la votación dura logra un equilibrio entre sesgo y varianza, ofreciendo una generalización más robusta.

El modelo de Random Forest utilizado individualmente con características reducidas mediante PCA también presentó un rendimiento muy sólido (97.32% de exactitud), con métricas muy cercanas a las del VotingClassifier.

En contraste, el modelo de AdaBoost obtuvo resultados notablemente inferiores, con una exactitud de apenas 22.38% y métricas de precisión, F1 y recall significativamente bajas. Este bajo desempeño puede explicarse por varias razones. Primero, AdaBoost es sensible a ruido y a datos desbalanceados o con clases difíciles de separar linealmente, especialmente cuando se utilizan clasificadores base muy simples, como los decision stumps. Aunque el conjunto fue previamente balanceado, es posible que las transformaciones del espacio mediante PCA y la complejidad de los patrones hayan hecho que estos clasificadores débiles no fueran capaces de aprender representaciones útiles. Además, AdaBoost enfatiza ejemplos mal clasificados en iteraciones sucesivas, lo que puede amplificar errores en lugar de corregirlos.

```
Friedman test statistic: 8.3158, p-value: 0.0156
Diferencias significativas detectadas. Aplicando prueba post-hoc de Nemenyi:
Matriz de p-values Nemenyi (comparaciones entre clasificadores):
          0         1         2
0  1.000000  0.802241  0.099292
1  0.802241  1.000000  0.019679
2  0.099292  0.019679  1.000000
```

El estadístico de Friedman resultó ser 8.3158 con un valor p de 0.0156, lo que indica que existen diferencias estadísticamente significativas entre al menos dos de los clasificadores con un nivel de significancia del 5%.

Para identificar cuáles pares de clasificadores presentan diferencias específicas, se realizó la prueba post-hoc de Nemenyi. Los resultados muestran que:

- La comparación entre Random Forest y Voting Classifier no es estadísticamente significativa ($p = 0.80$), lo que sugiere que ambos métodos tienen rendimientos similares en términos de exactitud.
- La comparación entre Voting Classifier y AdaBoost sí es estadísticamente significativa ($p = 0.0197 < 0.05$), mostrando que el rendimiento del AdaBoost es significativamente diferente — y en este contexto, inferior — al del Voting Classifier.

En conclusión, el AdaBoost se comporta de manera significativamente peor que el Voting Classifier, mientras que Random Forest y Voting Classifier muestran un desempeño similar.

4.0 Conclusiones Finales

El presente trabajo tuvo como propósito principal comparar el desempeño de diferentes modelos de aprendizaje automático aplicados al análisis de señales multivariadas, obtenidas tanto de sensores inerciales (acelerómetro, giroscopio y magnetómetro) como de registros electrocardiográficos (ECG). Con este fin, se utilizaron tres bases de datos distintas, lo que permitió evaluar la consistencia y generalización de los modelos implementados en distintos contextos.

Se identificó la importancia de los datos puesto que al estar trabajando con muestras de actividad humana se deben tomar en cuenta muchos aspectos para el entrenamiento correcto de los modelos, por ejemplo, tomar en cuenta los patrones de comportamiento de un sujeto de prueba, la estabilidad de los sensores, la variabilidad de los sujetos de prueba en cuanto a edad, peso, sexo, etc. Así como factores físicos respecto a la correlación entre datos.

El análisis comparativo de los resultados arrojó hallazgos relevantes:

La votación y la agregación con Random Forest demostraron ser las estrategias más efectivas, alcanzando precisiones superiores al 0.90 de forma consistente. Esto indica que la combinación de múltiples modelos puede capturar mejor la complejidad de los datos, logrando una mayor robustez frente a la variabilidad inherente de las señales.

AdaBoost, en cambio, mostró un desempeño significativamente inferior, con valores de precisión cercanos a 0.50. Este comportamiento sugiere una alta sensibilidad del modelo a ruido o datos desbalanceados, lo cual afectó negativamente su capacidad de generalización. Si bien AdaBoost puede ser potente en ciertos escenarios, sus limitaciones se evidencian cuando se trabaja con señales fisiológicas y sensores con alta variabilidad.

En conclusión, el objetivo central de comparar distintos modelos de aprendizaje automático fue alcanzado con éxito, demostrando que los enfoques basados en ensamblado robusto (como Random Forest y votación) ofrecen un rendimiento superior frente a métodos secuenciales como AdaBoost. Asimismo, el uso de técnicas de preprocesamiento como PCA se confirmó como una herramienta útil para optimizar los datos de entrada. Estos resultados son un aporte significativo

para futuras aplicaciones en clasificación de señales biomédicas y de movimiento, especialmente en contextos donde la precisión es crítica.

Referencias

1. https://archive.ics.uci.edu/dataset/341/smartphone+based+recognition+of+human+activities+and+postural+transitions
2. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2ª ed.). O'Reilly.
3. Elgendi, E.O. (2025). The Impacts of Data Science Applications on Construction Management. Journal of Construction Engineering Research and Management, AzharCERM.
4. TY, Et. Al. Leveraging Mobile Phone Sensors, Machine Learning and Explainable Artificial Intelligence to Predict Imminent Same-Day Binge Drinking Events to Support Just-In-Time Adaptive Interventions: A Feasibility Study (Preprint)
5. Lei Yu leiyu, Huan Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution.
6. Brandon M. Greenwell. Explainable Boosting Machines in R with the ebm Package
7. Hassn, B. M., Alomari, E. S., & Alrubaye, J. S. (2025). Adversarially Robust 1D-CNN for Malicious Traffic Detection in Network Security Applications. Journal of Cybersecurity & Applications.
8. Banos, O., Garcia, R., & Saez, A. (2014). MHEALTH [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5TW22.
9. Dernbach, S., Das, B., Krishnan, N. C., Thomas, B. L., & Cook, D. J. (2012). Simple and complex activity recognition through smartphones. En 2012 8th International Conference on Intelligent Environments (IE) (pp. 214–221). IEEE.
10. Igual, L., & Seguí, S. (2017). Introduction to Data Science: A Python Approach to Concepts, Techniques and Applications. Springer.
11. Altun, K., & Barshan, B. (2010). Human activity recognition using inertial/magnetic sensor units. En Proceedings of the International Workshop on Human Behavior Understanding (pp. 38–51). Bilkent University, Ankara, Turkey. Recuperado de https://kilyos.ee.bilkent.edu.tr/~billur/publ_list/hbu10.pdf
12. Data Carpentry. (2021). Tipos de datos y formatos. Recuperado de https://datacarpentry.github.io/python-ecology-lesson-es/04-data-types-and-format.html
13. N. V. Chawla, Et. Al (2002). SMOTE: Synthetic Minority Over-sampling Technique. Recueprado de: https://www.jair.org/index.php/jair/article/view/10302
14. Hui Han, Et. Al (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning.
15. Suddala, S. (2024). Automating the Data Science Lifecycle: CI/CD for Machine Learning Deployment. Recuperado de: https://www.researchgate.net/profile/Swathi-Suddala/publication/390591657_Automating_the_Data_Science_Lifecycle_CICD_for_Machine_Learning_Deployment/links/67f55dce76d4923a1a0073e6/Automating-the-Data-Science-Lifecycle-CI-CD-for-Machine-Learning-Deployment.pdf

16. Manikandan, V., & Neethirajan, S. (2025). Decoding Poultry Welfare from Sound—A Machine Learning Framework for Non-Invasive Acoustic Monitoring. Recuperado de https://www.mdpi.com/1424-8220/25/9/2912
17. GridSearchCV. Recuperado de https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
18. Xu, Y., Klein, B., Li, G., & Gopaluni, B. (2024). "XRF-Based Particle Sorting: A Comparative Evaluation of Feature Extraction Methods for Data Preprocessing". Recuperado de https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4897819
19. Chatterjee, S., Sinha, P., Gatla, R. K., & Kumar, D. G. (2025). Cyber-Resilient Detection of Power Quality Events with NSCT and PCA-SVM. Recuperado de https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10971426
20. U. Khan and S. Masood, "A Machine Learning Approach to Human Activity Recognition," 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), Waknaghat, India, 2020, pp. 167-171, doi: 10.1109/PDGC50313.2020.9315826.
21. E. Cavita-Huerta et al., “Clasificación de actividad física mediante señales de acelerometría,” Pädi Boletín Científico de Ciencias Básicas e Ingenierías, vol. 13, no. 2, 2024. [En línea]. https://repository.uaeh.edu.mx/revistas/index.php/icbi/article/download/12163/11108
22. F. E. Lesiuk y C. J. Seguel Aranda, “Análisis y comparación de modelos de machine learning,” Univ. Nacional del Comahue, 2025. [En línea]. Disponible en: http://rdi.uncoma.edu.ar/bitstream/handle/uncomaid/18513/Proyecto%20Integrador%20Profesional_%20Lesiuk_Seguel.pdf
23. Akkaya, B. (2021). The Effect of Recursive Feature Elimination with Cross-Validation Method on Classification Performance with Different Sizes of Datasets. Recuperado de https://www.researchgate.net/publication/354253728_The_Effect_of_Recursive_Feature_Elimination_with_Cross-Validation_Method_on_Classification_Performance_with_Different_Sizes_of_Datasets
24. Alhogail, A., & Alharbi, R. A. (2025). Effective ML-Based Android Malware Detection and Categorization. Recuperado de https://www.mdpi.com/2079-9292/14/8/1486
25. Vlaicu & Matei. (2025). Applications of Machine Learning Technology in Agricultural Data Mining. Recuperado de https://www.mdpi.com/2076-3417/15/10/5286