Implementación de IA/ML con Sensores Inerciales para Análisis de Dominadas en Biomecánica Deportiva

1. Marco Teórico General

La **Inteligencia Artificial (IA)**, específicamente el **aprendizaje automático (ML)** y el **aprendizaje profundo (DL)**, ha cobrado gran relevancia en biomecánica en años recientes. Estas técnicas permiten analizar datos de movimiento humano (cinemática, cinética, EMG, etc.) de manera objetiva y automatizada. En biomecánica deportiva, ML se emplea para tareas como **reconocimiento de actividades**, **estimación de parámetros biomecánicos** y **detección de eventos clave** en gestos deportivos. Por ejemplo, modelos de ML se han usado para estimar ángulos articulares y fuerzas articulares a partir de sensores, mostrando alta precisión frente a métodos tradicionales.

Metodologías de desarrollo: Para guiar proyectos de ML en biomecánica, se suele seguir un flujo similar a CRISP-DM adaptado a ML (conocido como **CRISP-ML**). Este enfoque define fases iterativas: comprensión del dominio y los datos, preparación de datos, modelado, evaluación y despliegue, incorporando consideraciones específicas de ML (como gestión de riesgos y calidad). La idea es mantener un proceso trazable y repetible que permita volver atrás y refinar pasos según los resultados obtenidos.

Casos de estudio: En diversos ejercicios deportivos ya se han aplicado ML para detección automática y análisis de movimiento:

- *Sentadillas (squat)*: Se han entrenado modelos para reconocer sentadillas y evaluar su técnica. Por ejemplo, sistemas portables pueden clasificar si una sentadilla es correcta o no, o detectar fases como la posición baja y alta, mediante señales de acelerómetros en la cintura o piernas. También se han predicho cargas en articulaciones durante la sentadilla usando ML en lugar de dinámica inversa clásica.
- *Saltos*: Con un solo IMU en el cuerpo es posible predecir la altura de un salto vertical o determinar parámetros como tiempo de vuelo. Modelos de regresión no lineal (p. ej., bosques aleatorios o redes neuronales) han logrado estimar la altura de salto a partir de características de aceleración con errores pequeños, ofreciendo una alternativa a las plataformas de contacto.
- *Carrera*: En análisis de la marcha y carrera, ML se usa para detectar eventos como contacto y despegue del pie, o para clasificar el patrón de pisada (talón vs punta) a partir de IMUs en los pies. Por ejemplo, redes neuronales recurrentes pueden identificar el momento de impacto con un error de solo milisegundos comparado con sistemas de referencia. Asimismo, modelos han clasificado la técnica de carrera o

estimado variables como longitud de zancada y fuerzas de reacción al suelo combinando datos inerciales y algoritmos de ML.

Tareas genéricas de modelado: Según el objetivo, el problema se puede plantear de distintas formas: (a) **Clasificación Para** reconocer la actividad realizada (p. ej., identificar si un movimiento es una sentadilla, un salto o una dominada) o la calidad del movimiento (correcto/incorrecto); (b) **Regresión** para predecir valores continuos como ángulos articulares, velocidad, fuerza o puntuaciones de desempeño; (c) **Detección de eventos** para encontrar instantes específicos en la señal (por ejemplo, el inicio y fin de una repetición, o el momento de mayor flexión de codo). También existen enfoques de **segmentación** temporal, donde se asigna una etiqueta de fase a cada momento de la secuencia (similar a una clasificación frame a frame).

Técnicas de ML utilizadas: Tradicionalmente, en reconocimiento de acciones con sensores corporales se emplearon algoritmos clásicos de ML como *k*-NN, árboles de decisión, bosques aleatorios, máquinas de vectores de soporte (**SVM**) o análisis discriminante. Estos algoritmos usan conjuntos de características manuales extraídas de las señales y han logrado buenos resultados en tareas sencillas. Sin embargo, en la última década el **aprendizaje profundo** ha ganado popularidad debido a su capacidad para extraer automáticamente patrones complejos. Modelos como las **redes neuronales convolucionales (CNN)** y **redes neuronales recurrentes (RNN)** (p. ej., con celdas LSTM) se han aplicado exitosamente a datos temporales de IMUs. Más recientemente, se han introducido arquitecturas avanzadas con mecanismos de **atención** y modelos tipo **transformer**, demostrando mejorar la precisión en tareas de reconocimiento de actividades humanas. Un ejemplo concreto: Jeong et al. emplearon una CNN profunda sobre datos de acelerómetro de muñeca para clasificar ejercicios como dominadas, remo con barra, press de banca, dips, sentadillas, peso muerto, etc., alcanzando hasta un **96% de exactitud** en la clasificación multi-ejercicio. En problemas de **regresión biomecánica** (p. ej., estimar ángulos o momentos articulares), estudios comparativos han hallado que tanto métodos de ML tradicionales como los modernos pueden ser eficaces; por ejemplo, al predecir la cinemática 3D de miembros inferiores con IMUs, un bosque aleatorio y una CNN obtuvieron los menores errores frente a SVM u otros modelos. En general, la combinación de sensores portables con modelos ML bien entrenados se vislumbra como una herramienta capaz de **suplir equipos de laboratorio** costosos (como la captura de movimiento óptica) en muchos análisis.

2. Aplicación Específica en Dominadas

En el caso particular de las **dominadas** (ejercicio de tracción vertical en barra fija), el objetivo es utilizar IMUs y algoritmos IA/ML para analizar automáticamente la ejecución. Varias sub-tareas específicas se abordan en este contexto:

- **Segmentación del gesto en fases**: Una dominada completa puede dividirse en fases clave, por ejemplo: **inicio**(posición colgado, brazos extendidos), ascenso hasta **"catch"** (máxima altura, con mentón sobre la barra) y **descenso/fin** (retorno a la posición inferior). Un sistema de ML puede aprender a segmentar la señal en estas fases. Esto equivale a detectar eventos como el comienzo y el final de cada repetición y el instante pico superior. En la práctica, suelen identificarse estos puntos buscando características en las señales inerciales (p. ej., un cambio de dirección en la aceleración vertical marca el punto más alto). Algoritmos de detección de eventos pueden ser reglas basadas en umbrales o modelos entrenados para clasificar cada instante como "subida" vs "bajada". Por ejemplo, en movimientos explosivos se ha logrado segmentar fases con alta exactitud usando un solo sensor y ML. En dominadas, podría entrenarse un modelo secuencial (como una RNN/LSTM) que etiqueta cada frame de tiempo con la fase correspondiente, o usar un enfoque de detección de picos en la señal filtrada para delimitar las fases.
- **Detección automática de repeticiones**: Contar cuántas dominadas realiza el atleta es un requisito básico. Usando IMUs, esto se puede lograr detectando ciclos en los datos. Métodos simples identifican repeticiones encontrando picos en la aceleración vertical o rotacional del torso. Métodos basados en ML pueden aprender patrones completos de una repetición. Una estrategia es combinar un **clasificador de actividad** con un módulo de conteo: por ejemplo, Patalas-Maliszewska et al. proponen un sistema con un **módulo de reconocimiento de actividad (ARM)** para identificar cuándo el sujeto está haciendo dominadas (versus otros ejercicios o pausa) y luego un **módulo de conteo de repeticiones (RCM)** que registra cada ciclo detectado. En su implementación, una red CNN procesa ventanas cortas de datos para distinguir entre "dominada" u otras actividades, y una vez que se identifica la actividad, el sistema contabiliza las repeticiones detectando transiciones en la salida del clasificador. Los resultados reportados son muy satisfactorios: con ventanas deslizantes solapadas y datos crudos de IMU, el modelo alcanzó ~0.92 de **accuracy** en la clasificación de ejercicio, y en conteo de repeticiones logró un **93% de acierto (error ≤ ±1 repetición)**, que sube a 97% permitiendo un error de ±2 repeticiones. Esto indica que el contador automático rara vez se desvía por más de una repetición del conteo real.
- **Estimación/regresión de ángulos articulares**: Otra aplicación de IA es inferir parámetros cinemáticos, como los ángulos de codo y hombro durante la dominada, directamente desde los datos del IMU. Esto equivaldría a un modelo de regresión que mapea las características del sensor (aceleración, velocidad angular, orientación) a valores de ángulo en grados. Tradicionalmente, si se coloca un IMU en cada segmento (por ejemplo, uno en antebrazo y otro en brazo), se podría calcular el ángulo del codo mediante la diferencia de orientaciones (integrando los giroscopios y usando filtros de fusión sensor). Pero con ML se puede ir más allá: por ejemplo, estimar **ángulos de múltiples articulaciones** o hacerlo con menos

sensores de los necesarios físicamente. Estudios de rehabilitación han demostrado que un modelo puede aprender a predecir el ángulo de rodilla o cadera usando las aceleraciones medidas en un solo sensor en el muslo. En dominadas, podría usarse un modelo entrenado con datos de referencia (como captura de movimiento) para predecir en tiempo real el ángulo de codo con solo un IMU en la muñeca o brazo. Moghadam et al. compararon varios algoritmos de ML para estimar curvas completas de ángulos articulados de la marcha usando IMUs, encontrando que las redes CNN y bosques aleatorios daban los menores errores respecto a la referencia óptica. Aunque no hay mucha literatura específica de dominadas, es razonable pensar que un **modelo secuencial (p. ej. LSTM)** entrenado con datos cinemáticos de dominadas podría predecir con pocos grados de error la flexión/extensión de codo y hombro durante cada repetición.

- **Clasificación de variantes técnicas**: Las dominadas tienen variaciones (pronación vs supinación de agarre – dominada vs. chin-up –, uso de impulso de piernas como en *kipping pull-ups* vs estricto, amplitud de movimiento incompleta, etc.). Un sistema basado en ML puede ser entrenado para **reconocer automáticamente la variante** o incluso detectar defectos técnicos. Por ejemplo, se podría clasificar si el atleta está realizando una dominada estricta o con impulso (kipping) analizando la señal: un impulso generará un patrón distinto de aceleración del torso y de las piernas. Del mismo modo, una dominada con agarre supino podría diferenciarse por el patrón de movimiento del antebrazo. En ausencia de trabajos específicos publicados sobre esto, se puede extrapolar de ejercicios similares. En el caso de *flexiones de pecho* (push-ups), por ejemplo, ya se ha logrado clasificar variaciones (estándar, con aplauso, con apoyo de rodillas) usando datos inerciales con algoritmos de clasificación supervisada. Para las dominadas, un **clasificador multiclase** (posiblemente una CNN 1D sobre la señal temporal de IMU) podría distinguir entre varios tipos de dominada. Adicionalmente, la IA puede servir para identificar *errores de forma*: por ejemplo, si el usuario no desciende completamente (ángulo de codo no llega a 180°) o no supera la barbilla, un modelo entrenado con ejemplos etiquetados de “correcto/incorrecto” podría detectar estas situaciones para retroalimentación.

Modelos y variables empleados: En los trabajos prototípicos que incluyen dominadas, se han usado generalmente **redes neuronales profundas** (p. ej., CNN) para el reconocimiento de la actividad y contar repeticiones. Estas redes toman como entrada ventanas cortas de señal multivariada (aceleraciones y giroscopios de uno o varios IMUs). Algunas arquitecturas utilizan capas convolucionales seguidas de capas densas, a veces complementadas con capas recurrentes o de pooling temporal, para captar la dinámica de cada repetición. Las **variables más relevantes** suelen provenir de la aceleración vertical del cuerpo o del movimiento angular del segmento superior: en una dominada, la aceleración vertical del pecho o muñeca muestra un patrón oscilante por cada repetición (positivo al subir, negativo al bajar), y suele tener un punto cero o inversión en la cima. Asimismo, la orientación del torso cambia ligeramente (inclinación hacia atrás al subir). Un estudio multi-ejercicio encontró que ubicar un IMU en la muñeca dominante proporciona información suficiente para distinguir ejercicios de la parte superior del cuerpo, lo que sugiere que para clasificar dominadas vs otros ejercicios es crucial la señal de la extremidad superior. Sin embargo, para estimar ángulos de codo con precisión podría requerirse un IMU en el brazo o antebrazo adicional. En cuanto a **evaluación**, la validación de modelos

en dominadas típicamente involucra comparar los resultados con anotaciones manuales o sistemas de referencia. Por ejemplo, para la segmentación en fases se mediría la diferencia de tiempo (error en ms) con respecto a un video etiquetado (¿detectó el modelo el inicio de la repetición al mismo tiempo que el humano?). Para la clasificación de variantes, se reportan métricas de precisión por clase. En la regresión de ángulos, se calculan errores medios en grados respecto a un goniómetro o vídeo-análisis.

En caso de no encontrarse literatura dedicada exclusivamente a dominadas, es útil apoyarse en estudios de **ejercicios de tracción similares**. Por ejemplo, **remos con barra** o jalones en polea implican patrones de tracción donde el torso se mueve menos pero los brazos realizan flexión similar; la detección de repeticiones en esos ejercicios con IMU sería análoga. Otro ejemplo son las *dips* o fondos, estudiados junto a dominadas y sentadillas en sistemas de reconocimiento de ejercicio – la dinámica invertida (empujar hacia arriba en lugar de tirar) produce señales distintas, pero el enfoque de reconocimiento es semejante. En resumen, aun si las dominadas no han sido investigadas tan ampliamente, **los métodos de IA desarrollados para actividades afines en entrenamiento de fuerza** pueden transferirse: usar sensores inerciales corporales, procesar sus señales con algoritmos de ML entrenados supervisadamente, y así identificar automáticamente repeticiones, fases y calidad de la ejecución.

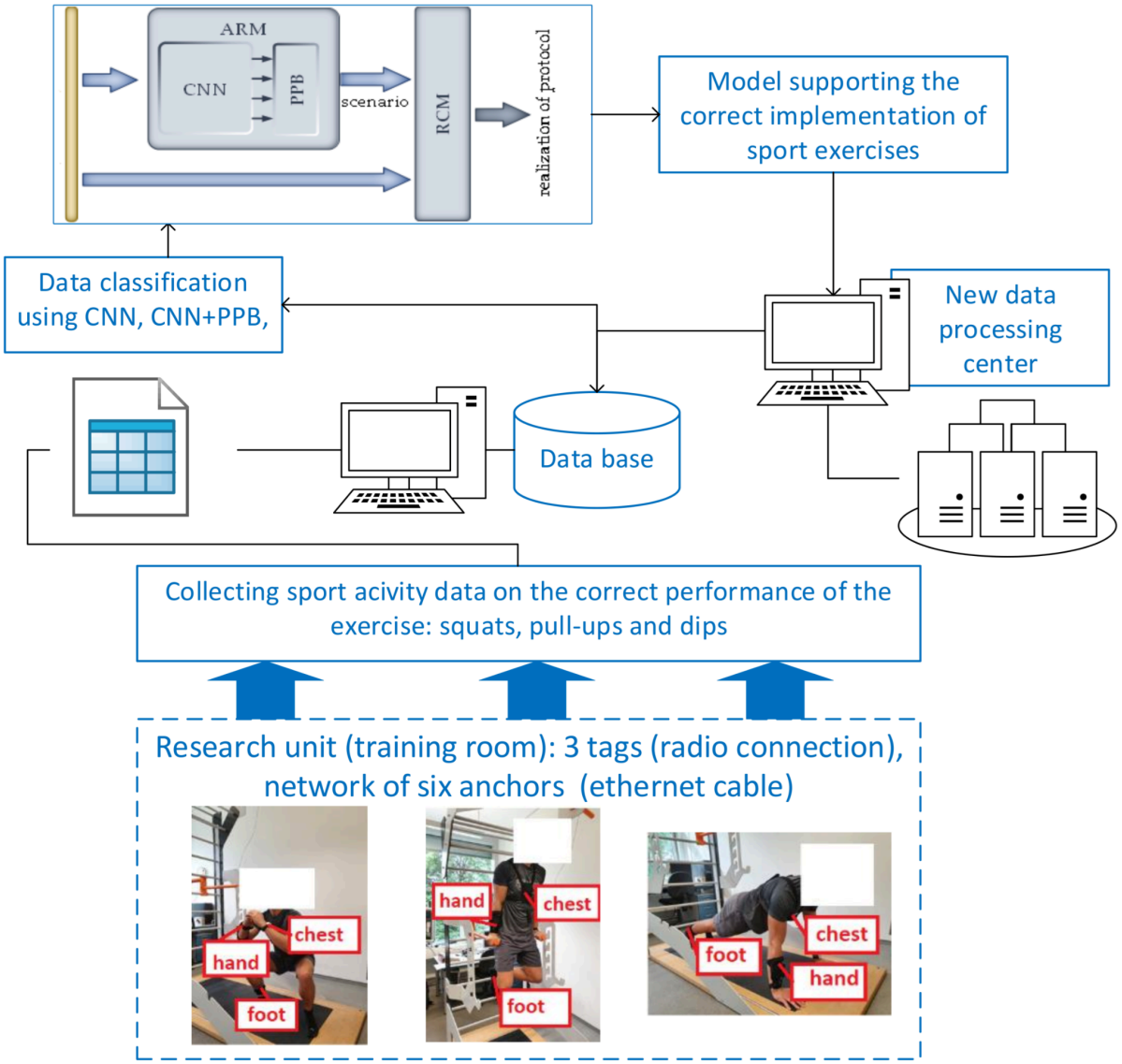

*Figura 1: Ejemplo de arquitectura para un sistema de análisis de ejercicios con IMUs. Incluye un módulo de reconocimiento de actividad (ARM) basado en una CNN con bloque de post-procesamiento (PPB) para segmentar el escenario de ejercicios, y un módulo de conteo de repeticiones (RCM). En este caso se usaron **3 sensores inerciales** colocados en **mano**, **pecho** y **pie** del deportista, junto con anclajes de ultrawideband para posicionamiento. El flujo va desde la **adquisición de datos** durante ejercicios (sentadillas, dominadas, dips) en un entorno controlado, el almacenamiento en base de datos, la **clasificación de la actividad** actual mediante la CNN, hasta el conteo de repeticiones detectadas. Adaptado de Patalas-Maliszewska et al. (2023).*

3. Preparación de Datos

El éxito de los modelos IA/ML depende en gran medida de una **buena preparación de los datos** de entrenamiento, especialmente en biomecánica donde los sensores inerciales generan datos de series temporales multivariadas. A continuación se describen las mejores prácticas en esta etapa:

- **Estructuración y etiquetado del dataset**: Primero, se debe diseñar cómo recopilar y organizar los datos de IMUs. Esto implica decidir la colocación de sensores (ej. un IMU en pecho para capturar movimiento del tronco en dominadas, o en muñeca para brazo), la frecuencia de muestreo (ej. 50-200 Hz suelen usarse para captura de movimiento humano), y registrar simultáneamente una referencia si es posible (video, marcadores, etc. para validar). Cada sesión de grabación debe anotarse cuidadosamente: por ejemplo, marcar en qué intervalos de tiempo se realizaron dominadas y cuántas reps, o en qué instantes comenzó y terminó cada repetición (estas etiquetas temporales son el *ground truth* para entrenamiento). Si se van a clasificar variantes, cada repetición se etiqueta con su categoría ("dominada estricta", "dominada kipping", etc.). Es importante estructurar los datos en *trazas identificables* (por sujeto, sesión, ejercicio) para luego facilitar la división en entrenamiento y prueba sin mezclar indebidamente (ver evitación de *leakage* abajo).
- **Preprocesamiento de las señales**: Las salidas de los IMUs pueden contener ruido y offset. Es habitual aplicar filtros o transformaciones antes de extraer características o alimentar un modelo:
    - *Filtrado*: se puede aplicar un filtro pasa-bajos (e.g., 5-10 Hz) para eliminar ruido de alta frecuencia que no corresponda al movimiento humano voluntario, o un filtro pasa-altos suave para remover la componente gravitatoria de la aceleración si interesa resaltar solo aceleraciones dinámicas. También se pueden usar filtros de mediana para eliminar picos abruptos (artefactos).
    - *Calibración*: al inicio de la sesión, conviene calibrar los sensores (p. ej., cero de acelerómetros cuando están quietos, alineación de ejes con la gravedad). También alinear los ejes del sensor con ejes anatómicos si es posible, aunque muchas veces esto se aprende automáticamente.
    - *Normalización*: es común normalizar las señales para estandarizar la escala. Por ejemplo, escalar cada canal de aceleración a unidades "g" y cada canal de giroscopio a °/s, y luego hacer una normalización estadística (restar media y dividir por desviación estándar) para cada tipo de señal. En el estudio de Patalas-Maliszewska se probó usar datos brutos vs. datos normalizados y se observó un ligero impacto en la precisión – la normalización a veces mejora la convergencia de los modelos ML.
    - *Segmentación inicial*: en lugar de alimentar toda la secuencia larga a un modelo, se suele trocear la señal en segmentos o **ventanas** de duración fija que luego serán clasificadas o analizadas individualmente (ver siguiente punto). Antes de segmentar, puede ser útil recortar porciones inútiles (por ej., periodos de descanso largos fuera de las repeticiones, a menos que se quieran incluir como clase "no ejercicio").

- **Extracción de características (features)**: Existen dos enfoques: uno basado en **características manuales** (feature engineering) y otro basado en **aprendizaje automático de características** (p. ej. con redes profundas). En el primer enfoque, para cada ventana de señal se calculan una serie de descriptores numéricos que capturan información relevante del movimiento:
    - *Características temporales*: estadísticas como media, desviación estándar, mediana, máximos y mínimos de la aceleración en cada eje durante la ventana; **número de picos** o cruces por cero (indicadores de repeticiones dentro de la ventana); duración de la fase positiva vs negativa; pendiente máxima (derivada) indicando explosividad; etc. Por ejemplo, en un trabajo se usaron características como la **cantidad y sincronización de picos**, coeficientes de un modelo autorregresivo, y medidas de simetría temporal de la señal, mostrando la diversidad de descriptores que pueden extraerse.
    - *Características frecuenciales*: transformando la señal al dominio de la frecuencia (mediante FFT o wavelets) se obtienen rasgos como energía en ciertas bandas, frecuencia dominante, entropía espectral, etc. En el caso de dominadas, podría ser útil la energía en bajas frecuencias (<5 Hz) que correspondería al ritmo repetitivo de las reps.
    - *Características cinemáticas específicas*: si se dispone de orientación del IMU (a través de quaterniones o ángulos de Euler), se pueden derivar ángulos relativos (p. ej., ángulo estimado de codo) o detectar posturas extremas. También combinaciones de sensores – por ejemplo, diferencia de aceleración entre pecho y pierna para detectar balanceo.
- Herramientas automatizadas como *TSFresh* pueden extraer miles de posibles features de series temporales. De hecho, Moghadam et al. reportan haber generado **45,704 características** de sensores (IMU+EMG) para predecir cinemática, reduciéndolas luego al subset más informativo. En muchos casos es conveniente aplicar **reducción de dimensionalidad** (p. ej., técnicas de selección de características, análisis PCA) para evitar alimentar al modelo con datos redundantes o irrelevantes, reduciendo así el riesgo de sobreajuste y el costo computacional.
  En el segundo enfoque (aprendizaje profundo), en vez de calcular manualmente los features, se alimentan directamente las secuencias crudas o filtradas a una red neuronal (p. ej. una CNN 1D) que automáticamente aprenderá filtros optimizados para extraer las características más discriminativas. Este enfoque *end-to-end* ha ganado popularidad pues a menudo supera al desempeño de features manuales tradicionales, siempre que haya suficientes datos para entrenar. Por ejemplo, una CNN puede aprender un filtro que esencialmente detecte el patrón de aceleración característico de una dominada completa, algo que manualmente hubiéramos capturado con “número de picos” y “amplitud”, pero la red lo aprende sola.
- **Segmentación en ventanas móviles**: Para alimentar al modelo de ML, es habitual segmentar la señal continua en **ventanas deslizantes** (*sliding windows*) de duración fija. La elección del tamaño de ventana es importante: debe ser lo suficientemente larga para capturar la estructura completa del evento de interés (p. ej., al menos una repetición completa en ejercicios cíclicos) pero no tan larga que mezcle múltiples eventos o que diluya la información en mucho ruido. En reconocimiento de actividad diaria con acelerómetros, típicamente se usan ventanas de 2-5 segundos con un solapamiento del 50%. En entrenamiento de fuerza, si nos centramos en detectar repeticiones individuales, una ventana podría ser del tamaño aproximado de una

repetición (~1–2 s dependiendo de la velocidad de ejecución). Por ejemplo, para dominadas (que suelen ejecutarse en ~1 s subida + 1 s bajada), una ventana de 2 segundos podría capturar una repetición completa. Se puede usar un solapamiento alto (p. ej. 50-75%) para no perder la transición entre ventanas. El solapamiento asegura que aunque la repetición caiga justo en el borde de una ventana, será captada en otra ventana desplazada. En el estudio de Patalas-Maliszewska, probaron ventanas solapadas vs no solapadas: con solapamiento lograron mejor accuracy (0.92 vs 0.88) en la detección de actividad, evidenciando la utilidad de ventanas móviles para captar suficiente contexto temporal.
Cada ventana recibe la etiqueta correspondiente según la tarea: por ejemplo, si estamos clasificando la actividad, una ventana durante una dominada se etiqueta como “dominada”; si estamos detectando fases, cada ventana (o cada punto dentro de la ventana) podría etiquetarse con la fase actual. Es importante evitar mezclar en la misma ventana dos estados diferentes (por eso a veces se descartan ventanas que contienen fronteras de repeticiones, o se usan técnicas como *label smoothing* cuando hay transición).

- **Balanceo de clases**: En los datos recolectados, a menudo algunas clases están **desbalanceadas**. Por ejemplo, puede haber muchas más ventanas de “no ejercicio” o descanso que ventanas de “dominada” efectiva, o quizás dentro de dominadas, la mayoría son con técnica correcta y muy pocas muestras de técnica incorrecta. El desbalance puede sesgar al modelo a favorecer siempre la clase mayoritaria. Para mitigar esto:
    - Se pueden aplicar técnicas de sobremuestreo de la clase minoritaria, como **SMOTE** (Synthetic Minority Over-sampling Technique), que genera muestras sintéticas nuevas interpolando las existentes de la clase minoritaria, para ampliar su representación. Por ejemplo, si solo se tienen 5 repeticiones etiquetadas como “incorrectas”, SMOTE podría crear varias más variando ligeramente las señales.
    - Alternativamente, realizar **submuestreo** de la clase mayoritaria (descartar aleatoriamente algunas muestras de la clase abundante) para balancear, aunque esto desperdicia datos.
    - Otra estrategia es usar **pesos de clase** en el algoritmo de entrenamiento: penalizar más los errores en la clase minoritaria que en la mayoritaria. Muchos algoritmos (p. ej. SVM, redes neuronales) permiten introducir un peso o factor de costo distinto por clase. Así, el modelo se entrenará esforzándose más en acertar la clase minoritaria.
    - Finalmente, es recomendable recolectar datos de forma que haya cierto equilibrio; por ejemplo, en las sesiones de entrenamiento de modelos, inducir voluntariamente algunos ejemplos de la clase minoritaria (p.ej., pedir a algunos sujetos que hagan deliberadamente dominadas con técnica incorrecta para tener más muestras de ese caso).
- **División de datos de entrenamiento/prueba y evitar *data leakage***: La correcta separación de los datos en conjuntos de entrenamiento, validación y prueba es crucial para evaluar el modelo justamente. Un error común es el *leakage* (fuga de información), que ocurre cuando datos utilizados para entrenar “se cuelan” indirectamente en el conjunto de prueba, inflando artificialmente el desempeño. En señales temporales y datos de múltiples sujetos, hay consideraciones especiales:

- Si las repeticiones de un sujeto aparecen en entrenamiento, no se deberían usar otras repeticiones del **mismo sujeto** en prueba, porque el modelo podría simplemente haber aprendido patrones específicos de esa persona (su ritmo, su forma) en vez de generalizar. Por ello, es preferible una separación **por sujeto**: por ejemplo, entrenar el modelo con datos de 8 participantes y reservar los datos de 2 participantes nunca vistos para probar (**Leave-One-Subject-Out**, en iteración). Esto evalúa la capacidad de generalización a individuos nuevos.
- Similarmente, si se tienen múltiples sesiones por sujeto en distintos días, se puede hacer separación por sesión (de modo que el modelo pruebe en una sesión distinta a las de entrenamiento, evitando que patrones de ruido o calibración específicos de una sesión se filtren).
- Al usar ventanas solapadas, asegurarse de que ventanas que se traslapan en el tiempo no queden repartidas entre entrenamiento y prueba. Por ejemplo, si una repetición genera 10 ventanas solapadas, todas ellas deberían pertenecer al mismo grupo (entrenamiento o prueba), ya que comparten mucha información. De lo contrario, el modelo podría ver parte de una repetición en entrenamiento y otra parte en prueba.
- Nunca utilizar *features* calculadas globalmente mezclando datos de entrenamiento y prueba. Por ejemplo, si se hace normalización por z-score, calcular la media y desviación solo con datos de entrenamiento y aplicar esa transformación a los de prueba (y no recalcularla incluyendo prueba). Incluir datos de prueba en la normalización es otro tipo de leakage.

Siguiendo estas prácticas se asegura que el modelo entrenado sea evaluado en datos verdaderamente independientes, reflejando su rendimiento real. En estudios recientes de biomecánica, se insiste en pruebas robustas: Rivadulla et al. sugieren validaciones cruzadas que involucren diferentes laboratorios para garantizar que los algoritmos no estén sobreajustados a un equipamiento o configuración específica. De hecho, Dumphart et al. mostraron que un detector de eventos de marcha entrenado en un laboratorio, al usarse en otro con distinta frecuencia de muestreo, presentaba errores de hasta 10 ms en la detección de eventos. Esto resalta la importancia de pruebas con datos verdaderamente nuevos y posiblemente heterogéneos, para confirmar la **generalización** del modelo.

4. Entrenamiento y Validación del Modelo

Una vez preparados los datos y definida la arquitectura de modelo IA/ML a utilizar, se procede al entrenamiento y validación. En este proceso se deben seguir prácticas rigurosas para obtener un modelo con buen rendimiento y evitar sobreajuste:

- **Procedimientos de validación cruzada**: Dado que típicamente el volumen de datos en estudios biomecánicos no es masivo (decenas de sujetos, miles de repeticiones a lo sumo), se aprovecha al máximo la información mediante **validación cruzada**. Existen distintas estrategias:
    - **k-fold cross-validation**: El dataset se divide en *k* subconjuntos (folds). Se entrenan *k* modelos diferentes, cada vez usando *k-1* folds como entrenamiento y 1 fold distinto como validación. Luego se promedian los resultados. Un caso común es 5-fold o 10-fold CV. Esto da una estimación más robusta del rendimiento que una simple partición train/test única.
    - **Leave-One-Subject-Out (LOSO)**: Como mencionado antes, se itera tomando los datos de un sujeto como conjunto de prueba y entrenando con los de los demás. Esto es especialmente útil para evaluar cómo el modelo funciona en un individuo completamente nuevo. Si el modelo se va a aplicar en personas no vistas, LOSO CV es un buen indicador. Muchas investigaciones en HAR con wearables reportan resultados LOSO porque es el escenario más desafiante pero más realista en términos de generalización.
    - **Hold-out con validación interna**: A veces se reserva un conjunto de *test* fijo (por ejemplo, 20% de los sujetos) y con el 80% restante se hace k-fold CV para desarrollo/validación. Esto permite afinar hiperparámetros en la CV interna y al final hacer una evaluación final en el hold-out.
- **Métricas de desempeño**: Para evaluar clasificadores, no basta con la *accuracy* global (porcentaje de aciertos). En problemas con múltiples clases o clases desequilibradas, se reportan métricas adicionales:
    - **Precisión (Precision)**: Proporción de predicciones positivas que realmente son positivas. En términos de matriz de confusión: `Precision = TP / (TP + FP)`. Una precisión alta significa que cuando el modelo dice "esto es una dominada correcta", pocas veces se equivoca.
    - **Exhaustividad (Recall)**: Proporción de positivos verdaderos que el modelo logra encontrar. `Recall = TP / (TP + FN)`. Un recall alto indica que captura casi todas las ocurrencias reales de la clase. (En español también se le llama *sensibilidad*).
    - **Puntuación F1**: Media armónica de Precision y Recall: `F1 = 2 * (Prec * Rec) / (Prec + Rec)`. Es útil para resumir el equilibrio entre ambos (especialmente cuando una es mucho más baja que la otra). Un F1 cercano a 1 indica buen balance precisión/exhaustividad.
    - **Curva ROC y AUC**: La curva ROC (Receiver Operating Characteristic) traza la relación entre la tasa de verdaderos positivos vs falsos positivos a diferentes umbrales de decisión. El **AUC** (Area Under Curve) resume en un número el rendimiento: AUC = 1 sería perfecto, 0.5 equivalente al azar. Suele usarse en clasificación binaria. Por ejemplo, para detectar “buena técnica” vs

"mala técnica", un AUC alto >0.9 indicaría que el modelo separa bien ambos casos.
    - **MCC (Matthews Correlation Coefficient)**: Es una métrica que tiene en cuenta TP, TN, FP, FN en un solo valor de -1 a 1. MCC=1 es predicción perfecta, 0 es aleatoria, -1 es opuesta a la realidad. Es especialmente informativa con clases desbalanceadas, ya que la accuracy puede ser engañosamente alta si el modelo solo acierta la clase mayoritaria. En evaluación de modelos de clasificación de movimiento, un MCC alto asegura que el modelo está acertando en todas las categorías y no solo evitando errores triviales.
    - **Otras**: dependiendo del contexto, se puede reportar *specificity* (especificidad), *NPV* (valor predictivo negativo), etc. En problemas de detección de eventos se reportan **tiempos de anticipación o retraso promedio** (ej., "el evento de inicio de rep se detectó con 50 ms de adelanto respecto al valor real"). En regresión de ángulos, se usan típicamente el **error cuadrático medio (MSE)** o el **error absoluto medio (MAE)** en grados.
- Es importante presentar métricas para cada clase o una media balanceada, especialmente si tenemos, por ejemplo, clase "sin repeticiones" vs "dominada": un modelo trivial que siempre diga "sin rep" podría tener 90% accuracy si efectivamente el 90% del tiempo el sujeto está en reposo, pero su recall de la clase "dominada" sería 0 (lo cual sería inaceptable). Por ello, se suelen incluir matrices de confusión completas y métricas por clase. En la literatura de análisis de entrenamiento de fuerza, se han reportado clasificaciones de ~70-90% de accuracy según el ejercicio y modelo, pero enfatizando la necesidad de mirar más allá de la cifra global y ver si ciertos casos (p. ej. mala técnica) están quedando desatendidos.
- **Regularización y prevención de sobreajuste**: Dado el relativamente bajo número de muestras y la alta dimensionalidad de los datos sensor + posibles parámetros de un modelo profundo, existe riesgo de **sobreajuste (overfitting)** – es decir, que el modelo aprenda detalles específicos del set de entrenamiento que no generalizan a nuevos datos. Para evitarlo, se emplean varias técnicas:
    - **Regularización L2**: Consiste en añadir un término de penalización en la función de pérdida proporcional al cuadrado de los pesos (norma L2). Esto fuerza a los pesos de la red a no crecer demasiado, manteniendo el modelo más simple (prefiere repartir los pesos en vez de depender de uno muy grande). En regresión lineal y SVM es equivalente a *ridge regression* o *weight decay*. La regularización L2 suele mejorar la capacidad de generalización reduciendo sobreajuste.
    - **Dropout (Desconexión aleatoria)**: En redes neuronales, la técnica de *dropout* apaga aleatoriamente un porcentaje de neuronas en cada iteración de entrenamiento. Por ejemplo, con 50% dropout en la capa oculta, cada vez solo la mitad de las unidades están activas. Esto previene que la red dependa mucho de combinaciones específicas de neuronas, obligándola a aprender representaciones redundantes y más robustas. En la fase de prueba no se aplica dropout pero los pesos están "promediados" implícitamente por haber entrenado así. Dropout suele ser muy efectivo en redes densas; en CNN también se usa a la salida de bloques convolucionales.

    - **Early Stopping (Detención temprana)**: Consiste en monitorear el desempeño en un conjunto de validación durante el entrenamiento e interrumpir el entrenamiento cuando la métrica de validación empeora (indicando inicio de sobreajuste). Por ejemplo, entrenar por máximo 100 epochs pero detener si tras X epochs el error de validación comienza a subir. Así se evita sobre-entrenar el modelo en los datos de entrenamiento.
    - **Data augmentation**: En visión por computador es muy común; en señales de IMU también se puede implementar. Por ejemplo, añadir pequeñas perturbaciones a las señales durante el entrenamiento: un ligero ruido, rotar el eje de referencia, o simular variaciones del sujeto (como escalar temporalmente la señal para simular distinta velocidad de ejecución). Esto aumenta efectivamente el tamaño del dataset y hace el modelo más inmune a variaciones. Cuidar que las transformaciones tengan sentido biomecánico (p. ej., invertir la señal de tiempo no tendría sentido físico, pero añadir ruido blanco de baja amplitud sí).
    - **Simplificación del modelo**: A veces, reducir la complejidad del modelo (menos capas o neuronas) regulariza de forma natural. Un modelo más pequeño tiene menos capacidad de sobreajuste. La selección de hiperparámetros (p. ej., profundidad de árbol, número de árboles en un Random Forest, número de neuronas en una capa oculta) puede guiarse mediante búsqueda en la validación para encontrar el mínimo modelo que logra buen desempeño.
- **Frameworks y herramientas**: Para implementar estos modelos, existen numerosos frameworks. Para algoritmos tradicionales (árboles, SVM, etc.) es muy popular **scikit-learn** (Python), que ofrece implementación eficiente de estos modelos y utilidades para validación cruzada y métricas. Para aprendizaje profundo, los frameworks dominantes son **TensorFlow** (y su API Keras) y **PyTorch**. En el ámbito académico de biomecánica se utiliza bastante Python con estas librerías, ya que facilitan construir prototipos de CNN/LSTM y entrenarlos en GPU. TensorFlow/Keras brinda muchas funciones listas (capas CNN1D, optimizadores con regularización, callbacks para early stopping, etc.) y PyTorch es muy flexible para personalizar arquitecturas. Algunos estudios también usan MATLAB (que tiene una toolbox de ML y deep learning) o incluso Weka/Orange para cosas más simples. En tiempos recientes, para despliegue en móviles o dispositivos, TensorFlow Lite y otras variantes especializadas entran en juego (ver sección siguiente). Un pipeline típico sería: extraer características con Python + scikit-learn, entrenar un Random Forest; o diseñar una red CNN-LSTM en Keras, entrenarla con GPU utilizando validación cruzada y varias épocas, y evaluar su matriz de confusión final sobre el set de prueba.

En resumen, la etapa de entrenamiento y validación exige rigor experimental: usar la **validación cruzada apropiada**, monitorear múltiples **métricas**, aplicar **regularización** para no sobreajustar, y documentar claramente los resultados. Lamentablemente, una revisión indicó que en varios sistemas automáticos para entrenamiento de fuerza publicados, había **errores metodológicos** y a menudo se pasaban por alto aspectos como la interpretabilidad y generalización. Por eso, se recomienda seguir estándares de desarrollo y reportar no solo altos porcentajes de acierto, sino proveer evidencias de que el modelo es **robusto** y **explicable** en contexto deportivo.

5. Inferencia en Tiempo Real y Optimización para Dispositivos Portables

Un aspecto clave de aplicar IA/ML en biomecánica deportiva es la posibilidad de llevar estos algoritmos al campo de entrenamiento en **tiempo real**, utilizando dispositivos portátiles (wearables, móviles o microcontroladores embebidos). Esto requiere optimizar los modelos para que sean lo suficientemente **eficientes** en cuanto a computación, memoria y consumo de energía, sin perder demasiada precisión.

- **Implementación de la inferencia en tiempo real**: La inferencia es el proceso de usar el modelo ya entrenado para producir predicciones sobre datos nuevos. En un escenario real-time con IMUs, implica que según el sensor genera datos (por ej., streaming vía Bluetooth de aceleraciones), el modelo procese inmediatamente cada ventana de datos y emita un resultado (por ejemplo, “1 repetición detectada” o “técnica incorrecta”). Para lograr funcionamiento en vivo, el pipeline debe estar diseñado con baja latencia:
    1. Se suelen utilizar **ventanas deslizantes en streaming**: a medida que llegan muestras, se llena un buffer de tamaño N (duración de la ventana) y cuando está completo se ejecuta la predicción del modelo en ese buffer. Si se solapa, puede que se calcule cada T milisegundos desplazando la ventana.
    2. El tiempo de cómputo de la predicción debe ser menor que el intervalo entre ventanas para no acumular retraso. Por ejemplo, si cada 0.5 s se evalúa una nueva ventana, la inferencia debe tomar mucho menos de 0.5 s.
    3. Se recomienda realizar pruebas de **profiling**: medir en el dispositivo objetivo cuánto tarda calcular features (si aplica) y correr el modelo. A veces es necesario recortar el modelo o simplificar features para cumplir restricciones de tiempo real.
- **Optimización y compresión de modelos**: Los modelos entrenados en entornos de PC pueden ser demasiado pesados para correr en un microcontrolador básico como un Arduino. Existen técnicas de **TinyML** para reducir el tamaño y complejidad del modelo:
    1. **Cuantización de modelo**: Consiste en reducir la precisión numérica de los pesos y operaciones del modelo. Típicamente, se convierten pesos de 32 bits en coma flotante a **enteros de 8 bits** (o coma fija de 8 bits. Esto reduce el tamaño del modelo a ~1/4 y suele aumentar la velocidad de inferencia (los microcontroladores pueden manejar enteros más rápido que floats, y además se aprovecha mejor la caché). La cuantización bien hecha apenas degrada la precisión del modelo (a veces menos de 1% de disminución en accuracy). Por ejemplo, TensorFlow Lite permite cuantizar un modelo entrenado al exportarlo, y la **Edge TPU** de Google utiliza cuantización para lograr alto rendimiento con bajo consumo.
    2. **Pruning (Poda)**: Implica eliminar conexiones/pesos poco importantes en la red neuronal, creando un modelo más esparso. Se pueden forzar a cero ciertos porcentajes de pesos (p. ej. eliminar un 30% de los pesos más cercanos a cero). La poda reduce la complejidad y tamaño del modelo, y si se combina con algoritmos que aprovechan la esparsidad, también acelera la inferencia. Idealmente, se realiza un entrenamiento con regularización L1 o

similar que incentive pesos cero, luego se prunan y se vuelve a afinar el modelo. Esta técnica puede reducir significativamente el número de operaciones sin gran pérdida de desempeño.

3. **Distillation (destilación)**: Es más compleja, pero consiste en entrenar un modelo pequeño ("estudiante") para imitar las salidas de un modelo grande ("profesor") en el conjunto de entrenamiento. De este modo, se transfiere el conocimiento a una arquitectura más compacta. Esto no garantiza eficiencia en microcontrolador, pero a veces logra modelos mucho más pequeños con desempeño cercano al original.
4. **Optimización de arquitectura**: A nivel de diseño, elegir un modelo más simple si es para dispositivo embebido. Por ejemplo, tal vez un modelo tipo Random Forest con 50 árboles pequeños pueda correr en un ARM Cortex-M, mientras que una LSTM grande no podría. O una CNN 1D con 1 o 2 capas en vez de 5 capas profundas, etc. También se pueden optimizar las features para reducir la entrada: en vez de alimentar 3 ejes de aceleración y 3 de giro, quizás solo 3 variables derivadas críticas.

- **Herramientas y frameworks especializados**: La comunidad TinyML ha desarrollado frameworks para facilitar la implementación en dispositivos de bajos recursos. **TensorFlow Lite Micro** es una versión de TF Lite que funciona en microcontroladores (sin sistema operativo, directamente en C/C++). Permite cargar modelos (cuantizados generalmente) y ejecutarlos con muy poca RAM (del orden de decenas de KB). Otras opciones incluyen **CMSIS-NN** (librerías optimizadas de redes neuronales para ARM Cortex-M), o plataformas como **Edge Impulse** que ayudan a entrenar e implementar modelos en diferentes placas (Arduino Nano 33 BLE Sense, etc.). Un ejemplo de éxito es un modelo 1D-CNN de muy baja carga, capaz de reconocer actividades humanas en un microcontrolador a 10 Hz. También la literatura reporta experimentos reduciendo tasa de muestreo para ahorrar computo con impacto mínimo en accuracy.
- **Ejemplos de dispositivos**: Un **Arduino Nano 33 BLE Sense** (Cortex-M4 a 64 MHz, 256 KB RAM) puede ejecutar modelos pequeños en tiempo real leyendo su propio IMU. También la familia **ESP32** (tiene potencia similar y conectividad Bluetooth/WiFi) es popular en proyectos de wearable ML. Para demandas mayores, existen microcontroladores con aceleradores dedicados, como la mencionada **Edge TPU** de Google (pensada para inferencia de redes cuantizadas a altas velocidades con bajo consumo) o chips como **STM32H7** (dual core, uno de ellos DSP). No obstante, el objetivo suele ser lograr que incluso la plataforma más sencilla logre correr el modelo – de ahí la importancia de la optimización.
- **Pipeline completo desde adquisición hasta visualización**: Integrar todo el flujo requiere ingeniería de sistemas además del modelo ML:
    1. **Adquisición**: Los sensores IMU obtienen datos (aceleración, giro) continuamente. Esto puede recolectarse por un microcontrolador conectado al sensor (ej., un Arduino con MPU6050) o por un smartphone con acelerómetro interno. Es común que los datos se lean en interrupciones de temporizador para asegurar periodicidad (p. ej., leer 100 muestras por segundo).
    2. **Preprocesamiento local**: El dispositivo puede filtrar o segmentar los datos en tiempo real. Por ejemplo, un Arduino podría calcular la media removiendo el offset, y mantener un buffer circular de muestras para la ventana actual.

3. **Inferencia del modelo**: Cada vez que se llena una ventana de datos (ej., 1 s de señal), se normalizan usando parámetros predefinidos y se computan las features requeridas (si el modelo es clásico) o se alimentan directamente a la red (si es un modelo neuronal). El modelo produce una predicción: por ejemplo, “clase = dominada” con cierta probabilidad, o “ángulo de codo = 90°”.
4. **Post-procesamiento**: Puede aplicarse lógica adicional, p.ej., un **bloque de post-proceso (PPB)** como en Patalas-Maliszewska et al. para suavizar las predicciones. Esto podría ser un filtro de mediana sobre las últimas N clasificaciones para evitar parpadeos entre clases. También conteo: si la clase “dominada” es continua por más de X instantes y luego cambia a “no dominada”, contar una repetición.
5. **Feedback/Visualización**: Finalmente, el sistema proporciona el resultado al usuario. En un móvil, puede ser una pantalla mostrando el conteo y tal vez un mensaje (“Técnica OK” o “Extiende más los brazos!”). En un microcontrolador, podría encender un LED o emitir un beep cada rep, o transmitir vía Bluetooth a otra interfaz. Por ejemplo, un prototipo wearable podría vibrar cuando detecta que la barbilla pasó la barra, para indicar éxito de la repetición.

- La **latencia** total desde que ocurre el evento hasta que se notifica debe ser corta. Para repeticiones de ejercicio, una latencia de ~0.1-0.2 s es generalmente tolerable (imperceptible para el usuario), pero más de medio segundo podría sentirse retrasado. En algunas aplicaciones críticas (p. ej. asistencia robótica), se requeriría latencias aún menores, pero para conteo de repeticiones un ligero retraso no es problemático.

En conclusión, lograr **inferencia en tiempo real** implica adaptar el modelo al hardware. Gracias a técnicas como cuantización y pruning, hoy es posible ejecutar redes neuronales simples en microcontroladores de muy bajos recursos. Esto abre la puerta a entrenadores personales portátiles: por ejemplo, unas bandas con IMU en brazos que cuentan dominadas y evalúan la técnica localmente sin necesidad de conexión a un PC. Aun así, siempre hay que verificar que la precisión del modelo no se degrade demasiado tras la compresión. Si un modelo resultara muy poco preciso al cuantizarlo, quizás la solución es optar por un modelo más simple desde el inicio (menos sensible a cuantización) o utilizar un dispositivo ligeramente más potente (p.ej., un smartphone) como apoyo.

6. Explicabilidad, Trazabilidad y Ética

Al introducir IA en el análisis deportivo, no solo importa la precisión, sino también **entender las decisiones del modelo**, garantizar la **trazabilidad de resultados** y cumplir con consideraciones éticas y legales. Este apartado cubre cómo abordar la **"caja negra"** de algunos modelos, validar su confianza, adaptarlos con el tiempo y respetar la privacidad de los datos:

- **Explicabilidad del modelo**: Las decisiones tomadas por un sistema de ML – por ejemplo, clasificar una repetición como incorrecta – deben poder explicarse de forma comprensible para generar confianza en usuarios y profesionales. Para modelos simples como árboles de decisión o reglas, la interpretación es directa (se pueden inspeccionar las reglas: ej. "Si el rango de movimiento < X, marcar rep como incompleta"). Sin embargo, para modelos complejos (p. ej. una red neural profunda que clasifica técnica), es necesario aplicar métodos de **XAI (eXplainable AI)**:
    - **Importancia de características**: En modelos tipo Random Forest o XGBoost, se puede calcular la importancia de cada feature en la predicción (basado en la reducción de impureza que genera en promedio). Esto indicaría, por ejemplo, que la "amplitud de aceleración vertical" es el atributo más influyente para detectar repeticiones. Ese conocimiento podría alinearse con la intuición del entrenador, validando el modelo.
    - **SHAP (SHapley Additive exPlanations)**: Es una técnica moderna que asigna a cada característica un valor de contribución a una predicción, fundamentado en teoría de juegos. Genera un vector de "importancias" para una instancia específica, que suman hasta la diferencia de la predicción con el promedio. Aplicado a una señal de dominada, SHAP podría mostrar que, para una repeticiones dada, las características derivadas del eje Z del acelerómetro tuvieron valores SHAP altos (indicando que gracias a ellas el modelo decidió que fue una dominada correcta) mientras que, por ejemplo, una duración corta tuvo un valor SHAP negativo (influyendo a clasificarla como incorrecta). Tales interpretaciones locales ayudan a explicar cada decisión del modelo.
    - **LIME (Local Interpretable Model-Agnostic Explanations)**: Similar propósito que SHAP, LIME construye un modelo lineal simple alrededor de la vecindad de la instancia a explicar. En nuestro contexto, podría crear una regresión lineal usando ligeras variaciones de la señal original para ver qué features inclina la decisión hacia "correcto" vs "incorrecto". Por ejemplo, descubrir que si se reduce artificialmente la amplitud de aceleración, la predicción cambia a "incorrecto" sugiere que la amplitud original era clave para "correcto".
    - **Visualización temporal**: Para datos temporales, existen métodos como relevancia temporal que pueden resaltar qué partes de la secuencia influyeron más. Si usáramos una red con mecanismo de atención para segmentar fases, las **"weights" de atención** podrían visualizarse para ver en qué momento la red detectó el fin de la repetición.
- El objetivo de explicabilidad es doble: por un lado, **verificar** que el modelo está aprendiendo algo coherente (no un artefacto espurio). Por ejemplo, si

inexplicablemente el modelo decide con una señal de la pierna algo sobre la dominada de brazos, quizá esté sobreajustando ruido o detectando un patrón casual. Por otro lado, proveer **retroalimentación entendible** al atleta o entrenador: un sistema explicable podría decir *"Repetición 5 no contó porque tu ángulo de codo solo llegó a 100° (menos del umbral)"* en vez de solo no contarla sin explicación. Esto facilita la aceptación de la tecnología en el entrenamiento.

- **Trazabilidad de resultados**: La trazabilidad se refiere a poder rastrear el origen de cada decisión y los datos involucrados. En un proyecto de biomecánica con IA, esto significa:
    - Mantener un **log detallado** durante la inferencia: por ejemplo, guardar los tiempos en que se detectaron repeticiones, los valores de las variables clave, etc. Así, si surge una duda ("¿ocurrió realmente 10 reps o el sistema contó mal?"), se puede revisar el registro.
    - Versionado de datos y modelos: Es importante llevar control de qué versión del modelo se usó en qué momento y con qué conjunto de entrenamiento. Si posteriormente se detecta un bug o se mejora el modelo, se puede comparar con la versión anterior. También, si un resultado parece anómalo, se puede replicar la inferencia offline con los mismos datos guardados para investigar.
    - Referenciabilidad a datos brutos: Siempre se recomienda guardar (al menos temporalmente) las **señales originales** o un resumen de ellas que llevó a una decisión. Esto es crítico en entornos donde las decisiones tienen implicaciones fuertes (p.ej., en rehabilitación, si un algoritmo sugiere un nivel de progreso y se basa una terapia en ello, conviene poder demostrar en qué datos se basó).
    - Calibración con sistemas de referencia: Para asegurar la trazabilidad metrológica, se suelen comparar las medidas de tiempo o ángulos del sistema IA contra un estándar. Por ejemplo, validar las detecciones de fase contra un sistema de fotogrametría: si la IA marca el pico de dominada con 30 ms de diferencia respecto al video, se documenta esa diferencia. Con este proceso de **validación cruzada**, se traza la correspondencia entre medidas de nuestro sistema y las reales, estableciendo márgenes de error conocidos. Estudios de eventos en marcha mencionan errores aceptables del orden de 10-20 ms en detectar contactos; en dominadas, diferencias de pocos centímetros en la altura final podrían traducirse en decenas de milisegundos de diferencia en detección, lo cual seguramente es tolerable. Lo importante es cuantificarlo y trazarlo.
- **Actualización y deriva de concepto**: Los modelos de ML pueden degradar su desempeño si las condiciones cambian respecto al entrenamiento inicial, fenómeno conocido como **concept drift**. En el contexto deportivo, pueden ocurrir cambios graduales o súbitos:
    - El atleta podría cambiar su técnica con el tiempo (por ejemplo, mejora su rango de movimiento, o adopta un estilo diferente). Un modelo entrenado con sus datos antiguos podría empezar a fallar en clasificar correctamente.
    - Si el dispositivo se usa con poblaciones diferentes a las entrenadas (digamos, se entrenó con deportistas jóvenes y ahora se usa con adultos mayores que hacen dominadas asistidas), el patrón de señal puede diferir.
    - Actualizaciones o cambios de los sensores (diferente modelo de IMU, distinta frecuencia) también pueden introducir drift.

- Para enfrentar esto, se deben planear **re-entrenamientos periódicos** o calibraciones: por ejemplo, cada cierto número de sesiones, recopilar nuevos datos del usuario y volver a entrenar el modelo incluyendo esos ejemplos. Si el dispositivo está en campo, podría usarse aprendizaje en línea o incremental; pero en muchos casos prácticos, se opta por recopilar datos de uso, enviarlos a un servidor para mejorar el modelo central y luego actualizar el firmware del dispositivo con el modelo mejorado (esto es común en dispositivos comerciales: las actualizaciones de software traen modelos refinados). Es importante alertar al usuario cuando el modelo se ha actualizado, especialmente si ello cambia la forma en que se interpretan sus métricas (por trazabilidad y honestidad científica).
  También conviene diseñar el sistema para detectar *drift*: monitorear si la distribución de las características en uso se está alejando de las del conjunto de entrenamiento original. Si se detecta, señalar la necesidad de recalibrar. Por ejemplo, si de pronto las aceleraciones promedio en dominadas de un usuario son mucho mayores que cualquiera del set de entrenamiento (quizá porque ahora añade lastre de peso), el sistema podría advertir "nuevo régimen detectado, se recomienda recalibrar el modelo con estos datos".
- **Consideraciones legales y de privacidad (GDPR, anonimización, model cards)**: El uso de sensores corporales y ML implica manejar **datos personales**, por lo que se deben seguir normativas de protección de datos:
  - **Privacidad y GDPR**: En la Unión Europea rige el RGPD (Reglamento General de Protección de Datos). Los datos de un wearable deportivo (acelerometría) aunque no identifiquen directamente a la persona, podrían considerarse datos personales de salud si revelan información sobre su condición física. Por tanto, se debe obtener **consentimiento informado** del usuario para recolectar y procesar sus datos con fines de análisis. Además, se debe informar qué se hace con esos datos, garantizar su seguridad (almacenamiento cifrado, transmisión segura) y permitir que el usuario los elimine si así lo desea. Un punto importante es la **anonimización**: para investigación o para mejorar modelos, los datos deben anonimizarse, es decir, desvincularse de la identidad del sujeto. Por ejemplo, en un dataset utilizado para re-entrenar el modelo global, no debería haber nombres ni identificadores personales; cada registro puede ser un código aleatorio. Incluso así, se debe evaluar que no haya riesgo de reidentificación (la acelerometría por sí sola difícilmente identifica a alguien, pero combinada con otros datos podría).
  - **Uso adecuado de los datos y sesgos**: Éticamente, hay que asegurar que el modelo no introduzca **discriminaciones** indebidas. En biomecánica deportiva, esto podría significar verificar que el modelo funcione igual de bien para hombres y mujeres, o para diferentes rangos de altura corporal, etc., si esos factores no están directamente incluidos. Si se descubre que el modelo tiene mayor error en, digamos, personas con movilidad limitada, habría que abordar ese sesgo incluyendo más datos de ese grupo o ajustando el algoritmo.
  - **Transparencia – Model Cards**: Mitchell et al. (2019) propusieron las **"Model Cards"** como documentación estandarizada para modelos de ML. En contexto de nuestro proyecto, sería muy útil crear una **ficha técnica del modelo** que describa: qué datos de entrenamiento se usaron (número de

sujetos, variedad), qué métricas de desempeño tiene (globales y por subgrupo, por ejemplo por sexo o nivel deportivo), para qué contexto está **diseñado** (p. ej., “modelo válido solo para dominadas con agarre pronado, rango de movimiento completo”), y advertencias de limitaciones. Esto ayuda a gestionar expectativas y responsabilidad: el usuario sabrá en qué condiciones el modelo es fiable. Si el sistema se comercializa, incluir algo equivalente a “Este modelo fue validado con atletas recreativos de 18-40 años; no ha sido probado en niños o poblaciones mayores” serviría para evitar un uso indebido o al menos para que el usuario entienda los riesgos.

- **Seguridad y responsabilidad**: Aunque no es estrictamente legal, éticamente se debe considerar la seguridad del usuario. Un modelo que da feedback errado podría inducir a sobreesfuerzo o técnica incorrecta persistente. Por ello, se podría incluir en las *model cards* o documentación una recomendación de siempre combinar el feedback automático con la supervisión de un entrenador, al menos inicialmente, y de no usar el sistema como única base para decisiones críticas (principio de responsabilidad humana). En Europa, los sistemas de decisión automática con impacto significativo requieren posibilidad de revisión humana. En deporte, el impacto no es legalmente crítico, pero sí podría afectar la salud del atleta. Así que mantener un **humano en el bucle** (ej. el entrenador revisa los resúmenes y valida) es deseable.

En resumen, incorporar IA en el análisis de dominadas va más allá de lograr buenos números de predicción. Se deben **explicar las razones** (usando métodos como SHAP para ver la contribución de las señales a cada decisión), mantener la **trazabilidad** de cada resultado (registros, comparaciones con referencias de laboratorio en milisegundos para estar seguros de la precisión) y **actualizar** el sistema conforme evoluciona el usuario o se detecten desvíos. Todo esto, mientras se respetan la **privacidad** de los datos del deportista y se comunica abiertamente el alcance y limitaciones del modelo mediante documentación transparente (al estilo *model cards*). Estas prácticas garantizan que la solución IA/ML sea **confiable, ética y sostenible** en el tiempo, generando confianza tanto en los atletas como en los profesionales que los entrenan.

Glosario de Términos Técnicos

- **Inteligencia Artificial (IA)**: Área de la computación que busca crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana. Incluye muchos subcampos, desde planificación y razonamiento hasta percepción. El ML y DL son sub-disciplinas de la IA.
- **Aprendizaje Automático (Machine Learning, ML)**: Conjunto de técnicas dentro de la IA que permiten a un sistema **aprender** patrones a partir de datos, en lugar de ser programado explícitamente con reglas fijas. Incluye algoritmos supervisados, no supervisados y por refuerzo. En biomecánica se usa para reconocer actividades, predecir valores biomecánicos, etc.
- **Aprendizaje Profundo (Deep Learning, DL)**: Subcampo de ML que utiliza **redes neuronales artificiales** con múltiples capas (profundas) para modelar datos complejos. Puede descubrir automáticamente representaciones de alto nivel en los datos. Ejemplo: una red profunda puede aprender características de la señal de un IMU que correspondan a patrones de movimiento, sin intervención humana.
- **Sensor Inercial (IMU)**: Dispositivo que combina usualmente un **acelerómetro** (mide aceleración lineal en 3 ejes) y un **giroscopio** (mide velocidad angular en 3 ejes), a veces también un **magnetómetro** (campo magnético, usado para orientación absoluta). Proporciona mediciones de movimiento del cuerpo donde esté colocado. Son pequeños y se usan como wearables para captar la cinemática humana.
- **CRISP-ML**: Adaptación del estándar CRISP-DM (Cross-Industry Standard Process for Data Mining) al ciclo de vida de proyectos de Machine Learning. Consta de fases: 1) Entendimiento del negocio/problema, 2) Entendimiento de los datos, 3) Preparación de los datos, 4) Modelado, 5) Evaluación, 6) Despliegue. Es iterativo y ... Es iterativo y adaptado a los proyectos de ML, asegurando retroalimentación constante y control de calidad en cada fase.
- **HAR (Human Activity Recognition)**: Reconocimiento automático de actividades humanas. En español a veces *Reconocimiento de Actividad Humana*. Consiste en clasificar qué acción realiza una persona (caminar, correr, saltar, etc.) a partir de sensores. En biomecánica deportiva, HAR abarca distinguir diferentes ejercicios o fases de un movimiento mediante datos de IMUs, vídeos, etc.
- **Sobreajuste (Overfitting)**: Ocurre cuando un modelo aprende demasiado los detalles o ruido del conjunto de entrenamiento y pierde capacidad de generalizar a nuevos datos. Un signo de sobreajuste es rendimiento casi perfecto en entrenamiento pero pobre en validación. Se combate con regularización, más datos, o modelos más simples.
- **Regularización**: Conjunto de técnicas para impedir el sobreajuste penalizando la complejidad del modelo. Por ejemplo, la **regularización L2** agrega un término que fuerza a que los pesos de un modelo sean lo más pequeños posibles, simplificando la solución. Otra forma es **regularización por abandono (dropout)**, que aleatoriamente desconecta neuronas durante el entrenamiento para que la red no dependa de combinaciones específicas de características.
- **Dropout**: Técnica de regularización en redes neuronales donde se “apagan” aleatoriamente un porcentaje de neuronas en cada iteración de entrenamiento. Esto obliga a la red a ser más robusta, ya que no puede confiar en una ruta específica para hacer una predicción. En validación/prueba, todas las neuronas están activas

pero el efecto es equivalente a promediar muchos sub-modelos, reduciendo el sobreajuste.

- **Métricas de clasificación**:
  - **Accuracy (Exactitud)**: Fracción de predicciones correctas sobre el total. Útil como indicador global, pero puede ser engañosa si las clases están desbalanceadas.
  - **Precisión (Precision)**: Porcentaje de instancias predichas como positivas que realmente lo son (TP/(TP+FP)). Alta precisión significa pocas alarmas falsas.
  - **Exhaustividad o Recall**: Porcentaje de instancias positivas reales que el modelo logra identificar (TP/(TP+FN)). Alto recall significa pocos falsos negativos (no se le escapan muchos positivos verdaderos).
  - **F1-Score**: Media armónica de precisión y recall, útil para resumir ambas en un solo valor.
  - **AUC-ROC**: Área bajo la curva ROC, mide la capacidad de separabilidad de clases del modelo en diferentes umbrales (1.0 es perfecta, 0.5 es aleatoria).
  - **MCC (Matthews Correlation Coefficient)**: Métrica entre -1 y 1 que considera verdaderos y falsos de ambas clases; es informativa aun con clases desequilibradas (1 es perfecto, 0 aleatorio, -1 totalmente erróneo).
- **Algoritmos y modelos comunes**:
  - **Árbol de decisión**: Modelo de clasificación/regresión basado en reglas if-then anidadas en forma de árbol. Fácil de interpretar pero puede sobreajustar si es muy profundo.
  - **Random Forest (Bosque aleatorio)**: Ensamble de muchos árboles de decisión entrenados sobre distintas muestras; suele mejorar generalización y manejar bien datos tabulares.
  - **SVM (Support Vector Machine)**: Clasificador que encuentra el hiperplano que mejor separa las clases en el espacio de características, utilizando *kernels* para casos no lineales. Eficaz en conjuntos de tamaño moderado y alta dimensión, aunque menos usado hoy en día frente a redes neuronales para grandes datasets.
  - **Red Neuronal Convolucional (CNN)**: Tipo de red profunda especializada en extraer características espaciales o temporales usando filtros (convoluciones). En señales de tiempo, las CNN 1D pueden captar patrones locales (ej. un pico de aceleración). Han demostrado alto rendimiento en reconocimiento de patrones complejos.
  - **Red Neuronal Recurrente (RNN)**: Red diseñada para secuencias, que mantiene estados internos para recordar información previa. Las variantes modernas como **LSTM (Long Short-Term Memory)** o GRU solucionan problemas de memoria a largo plazo en secuencias. Se usan para modelar dependencias temporales largas (p. ej., en series de IMU para captar dinámica en el tiempo).
  - **Transformer**: Arquitectura de red basada en mecanismos de atención en lugar de recurrencia. Permite capturar relaciones de largo alcance en secuencias de forma eficiente. En HAR empieza a usarse para secuencias largas de sensores, aunque es más común en NLP y visión.
- **Cuantización (Quantization)**: Proceso de reducir la precisión numérica de un modelo, por ejemplo pasar de pesos en coma flotante de 32 bits a enteros de 8 bits.

Esto **comprime** el modelo (menos memoria) y puede acelerar la inferencia en hardware sencillo sin unidad de coma flotante, con mínima pérdida de precisión si se hace correctamente.

- **Poda de modelo (Pruning)**: Técnica de compresión donde se eliminan (ponen a cero) pesos o neuronas poco relevantes de la red neuronal. El modelo resultante es más ligero y rápido, manteniendo casi el mismo rendimiento si la poda se hace sobre componentes realmente "irrelevantes". Puede realizarse iterativamente con re-entrenamiento para recuperar desempeño.
- **SHAP**: Acrónimo de *SHapley Additive exPlanations*. Método post-hoc de explicabilidad que atribuye a cada característica un valor que representa su contribución a la predicción de un modelo para una instancia dada. Se basa en la idea de los valores de Shapley de la teoría de juegos. Útil para explicar por qué el modelo tomó cierta decisión (p. ej., "esta variación en aceleración sumó +0.2 a la probabilidad de ser dominada correcta") de manera consistente y localmente precisa.
- **LIME**: *Local Interpretable Model-Agnostic Explanations*. Otro método de XAI que genera explicaciones locales ajustando un modelo interpretable (lineal o árbol corto) en la vecindad de la predicción a explicar. Ayuda a aproximar qué características impulsaron la decisión en esa región del espacio de datos.
- **Model Card (Tarjeta de modelo)**: Documento estandarizado que acompaña a un modelo de ML, describiendo sus *características de rendimiento, datos de entrenamiento, alcance previsto y limitaciones conocidas*. Propuesto para fomentar la transparencia, especialmente en modelos usados por públicos no expertos. Una model card para nuestro modelo de dominadas incluiría, por ejemplo: "Entrenado con 1000 reps de 20 individuos (15H/5M), preciso 95% en conteo de repeticiones, probado en edades 20-40; no validado para dominadas asistidas; puede fallar si el sensor se mueve del pecho."
- **RGPD (GDPR)**: Reglamento General de Protección de Datos de la UE (en inglés General Data Protection Regulation). Ley de privacidad que establece directrices sobre recolección, procesamiento y almacenamiento de datos personales. Requiere, entre otros, consentimiento explícito del usuario, minimización y anonimización de datos, derecho al olvido (el usuario puede solicitar borrar sus datos), y transparencia en el uso. En nuestro contexto, implica tratar los datos biométricos (acelerometría, etc.) como información sensible, garantizar que no se usen fuera del propósito acordado (p. ej., entrenamiento personal) y protegerlos de accesos no autorizados.
- **Deriva de concepto (Concept drift)**: Fenómeno donde la relación estadística entre las características de entrada y la salida objetivo cambia con el tiempo. El modelo "aprendido" se vuelve menos válido porque el proceso subyacente evolucionó. En biomecánica, esto puede suceder si el atleta modifica su técnica o condición física, o si se usa el modelo en un contexto diferente al original. Requiere recalibrar o reentrenar periódicamente el modelo para mantener la precisión.

Referencias y Lecturas Recomendadas

- Patalas-Maliszewska, J., Pajak, I., Krutz, P., Pajak, G., Rehm, M., Schlegel, H., & Dix, M. (2023). **Inertial Sensor-Based Sport Activity Advisory System Using Machine Learning Algorithms.** *Sensors, 23*(3), 1137. https://doi.org/10.3390/s23031137
- Hart, R., Smith, H., & Zhang, Y. (2021). **Systematic review of automatic assessment systems for resistance-training movement performance: A data science perspective.** *Computers in Biology and Medicine, 137*, 104779. https://doi.org/10.1016/j.compbiomed.2021.104779
- Cust, E. E., Sweeting, A. J., Ball, K., & Robertson, S. (2019). **Machine and deep learning for sport-specific movement recognition: a systematic review of model development and performance.** *Journal of Sports Sciences, 37*(5), 568–600. https://doi.org/10.1080/02640414.2018.1521769
- Jeong, P., Choe, M., Kim, N., Park, J., & Chung, J. (2019). **Physical workout classification using wrist accelerometer data by deep convolutional neural networks.** *IEEE Access, 7*, 182406–182414. https://doi.org/10.1109/ACCESS.2019.2960088
- Mohammadi Moghadam, S., Yeung, T., & Choisne, J. (2023). **A comparison of machine learning models' accuracy in predicting lower-limb joints' kinematics, kinetics, and muscle forces from wearable sensors.***Scientific Reports, 13*(1), 5046. https://doi.org/10.1038/s41598-023-31906-z
- Dindorf, C., Horst, F., Slijepčević, D., Dumphart, B., Dully, J., Zeppelzauer, M., Horsak, B., & Fröhlich, M. (2025). **Machine Learning in Biomechanics: Key Applications and Limitations in Walking, Running, and Sports Movements.** En M. J. Blondin, I. Fister Jr., & P. M. Pardalos (Eds.), *Artificial Intelligence, Optimization, and Data Sciences in Sports*. Springer, Cham. https://doi.org/10.1007/978-3-031-76047-1_4
- Liang, W., Wang, F., Fan, A., Zhao, W., Yao, W., & Yang, P. (2023). **Extended application of inertial measurement units in biomechanics: from activity recognition to force estimation.** *Sensors, 23*(9), 4229. https://doi.org/10.3390/s23094229
- Warden, P., & Situnayake, D. (2019). **TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers.** O'Reilly Media.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). **Deep Learning.** MIT Press.
- Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., et al. (2019). **Model Cards for Model Reporting.** En *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*'19)* (pp. 220–229). ACM. https://doi.org/10.1145/3287560.3287596