## **INFORME IEEE ADAPTADO AL PROYECTO DE DETECCIÓN DE ACTIVIDADES HUMANAS**

Samuel Alvarez Alban - A00394750  
Gerson Hurtado Borja - A00394995  
Rafaela Ruiz- A00395368

### **Resumen**

Este proyecto implementa un sistema de detección de actividades humanas en tiempo real utilizando inteligencia artificial. Se procesaron videos propios capturados desde diferentes perspectivas para identificar actividades como caminar hacia la cámara, caminar de espaldas, girar, sentarse y levantarse. Usando MediaPipe, se extrajeron los puntos articulares clave, a partir de los cuales se calcularon características biomecánicas como ángulos articulares y distancias. Los datos fueron normalizados y utilizados para entrenar un modelo supervisado (Random Forest), el cual clasifica la actividad realizada. La solución permite inferencia en vivo desde cámara, mostrando la actividad detectada sobre la imagen.

### **Abstract**

This project implements a real-time human activity recognition system using artificial intelligence. Custom videos were captured from different angles to identify activities such as walking towards the camera, walking away, turning, sitting, and standing. MediaPipe was used to extract joint landmarks, and biomechanical features like joint angles and distances were calculated. The data was normalized and used to train a supervised learning model (Random Forest), which classifies the observed activity. The solution supports real-time camera inference with activity prediction displayed live.

## **I. INTRODUCCIÓN**

El reconocimiento automático de actividades humanas tiene múltiples aplicaciones en campos como la medicina, la rehabilitación física, la vigilancia y el análisis deportivo. Este proyecto se centra en la clasificación de cinco actividades clave mediante el análisis de video en tiempo real y la utilización de modelos de aprendizaje supervisado. A través de la extracción de puntos articulares y el procesamiento de características biomecánicas derivadas, se logra una representación eficiente del movimiento humano que alimenta un modelo capaz de identificar la actividad actual de un individuo.

## **II. MARCO TEÓRICO**

### **A. Detección de Articulaciones**

MediaPipe permite detectar 33 puntos del cuerpo humano (landmarks) en cada cuadro de video, proporcionando coordenadas (x, y, z) y un valor de visibilidad para cada punto.

### **B. Normalización**

Se normalizaron los datos utilizando StandardScaler para eliminar la dependencia de tamaño corporal, posición o distancia a la cámara, garantizando datos comparables entre individuos.

### **C. Características Biomecánicas**

Se calcularon ángulos articulares (ej. rodilla, cadera, codo) y distancias relativas (hombros, caderas) para representar mejor la postura y el movimiento, eliminando la dependencia de coordenadas absolutas.

### **D. Modelos Supervisados**

Se entrenó un modelo RandomForestClassifier por su robustez ante ruido y capacidad de interpretar relaciones no lineales. Se evaluó su desempeño con precisión, recall y F1-score.

## **III. METODOLOGÍA**

### **A. Recolección de Datos**

Se grabaron videos propios realizando las cinco actividades en distintos entornos, con diferentes usuarios y velocidades. Los videos se organizaron por clase en carpetas separadas para facilitar su procesamiento.

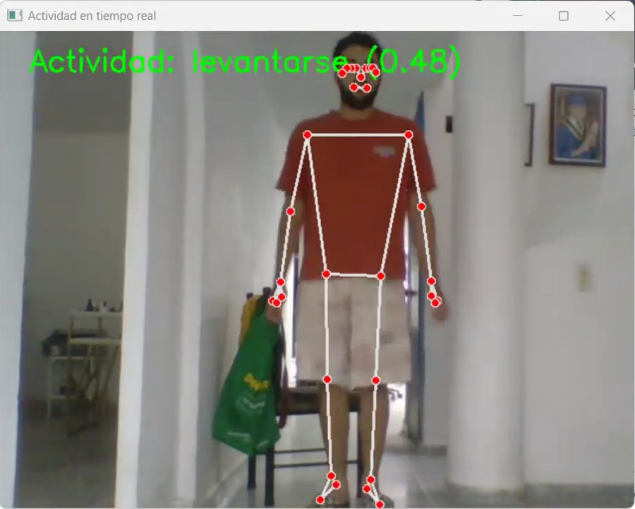
Texto

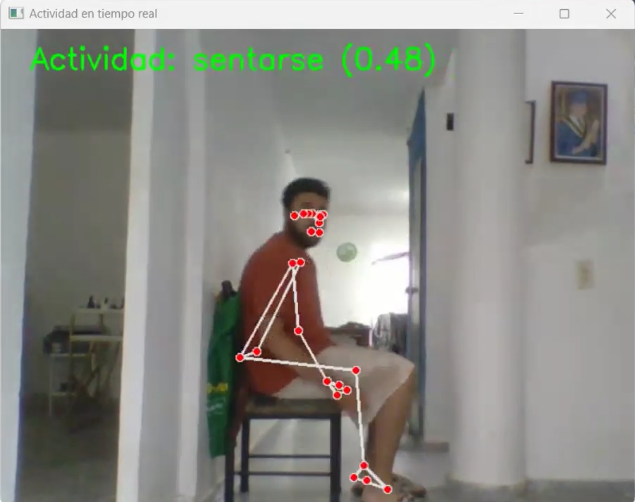
El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.esta sería la estructura del proyecto respectivamente y el formato de guardado para los videos es el siguiente:   
[nombreUser]\_[TipoMovimiento]\_[DireccionDeGr abacion]\_[#Video]

### **B. Extracción de Landmarks**

Se utilizaron las soluciones de MediaPipe Pose para extraer los 33 landmarks por cuadro. Los datos se almacenaron como archivos .csv.



**C. Generación de Características**

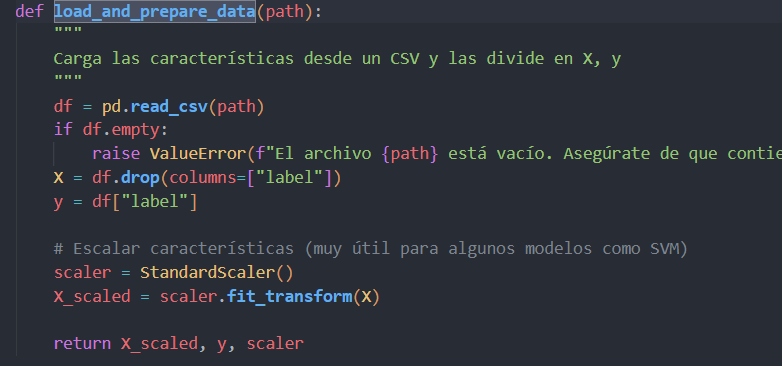
Se creó un módulo que calcula por cuadro:

* Ángulos articulares (cadera, rodilla, codo, hombro).
* Inclinación del tronco.
* Distancia entre hombros.
* Otras métricas derivadas de la posición relativa de los puntos clave.

Estas características fueron seleccionadas por su capacidad para discriminar entre diferentes tipos de movimientos y posturas. El cálculo de ángulos se realizó mediante funciones trigonométricas, asegurando la consistencia con las variables utilizadas durante el entrenamiento y la inferencia.

**Procesamiento de Datos**

Antes del entrenamiento, los datos fueron normalizados utilizando la técnica de estandarización (StandardScaler), lo que permite que todas las características tengan media cero y desviación estándar uno. Este paso es fundamental para evitar que variables con diferentes escalas afecten negativamente el desempeño del modelo.



El dataset se dividió en dos subconjuntos: entrenamiento (80%) y prueba (20%), utilizando la función [train\_test\_split](vscode-file://vscode-app/c:/Users/gdjhb/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) de scikit-learn con la opción [stratify](vscode-file://vscode-app/c:/Users/gdjhb/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html) para mantener la proporción de clases en ambos conjuntos. Esto garantiza una evaluación justa y representativa del modelo.

Texto

El contenido generado por IA puede ser incorrecto.

**Entrenamiento del modelo**

Para la clasificación de actividades, se seleccionó el algoritmo Random Forest (RandomForestClassifier), conocido por su robustez y capacidad para manejar datos tabulares con variables numéricas y categóricas.  
El modelo se entrenó utilizando el conjunto de entrenamiento, ajustando los parámetros principales (número de árboles, semilla aleatoria) para asegurar reproducibilidad y buen desempeño.

Durante el entrenamiento, se monitoreó el comportamiento del modelo para evitar sobreajuste y se validó que las métricas de desempeño fueran consistentes.

Texto

El contenido generado por IA puede ser incorrecto.

**Evaluación y visualización de resultados**

El modelo entrenado fue evaluado sobre el conjunto de prueba. Se calcularon y visualizaron las siguientes métricas:

* Reporte de clasificación: Precisión, recall y F1-score para cada clase, permitiendo identificar el desempeño específico en cada tipo de actividad.
* Matriz de confusión: Representada como un heatmap, facilita la identificación de errores comunes y clases que tienden a confundirse.
* Gráfica de F1-score por clase: Permite comparar visualmente el rendimiento del modelo en cada actividad.
* Importancia de características: Se graficó la relevancia de cada variable en las decisiones del modelo, ayudando a
* interpretar el comportamiento del clasificador.

Todas las gráficas se generaron automáticamente y se almacenaron en la carpeta “[results](vscode-file://vscode-app/c:/Users/gdjhb/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)” para su posterior análisis y documentación.

**Almacenamiento y reutilización del modelo**

Una vez validado el desempeño del modelo, tanto el clasificador entrenado como el objeto de escalado (scaler) fueron guardados en archivos .pkl utilizando la librería joblib. Esto permite reutilizar el modelo sin necesidad de reentrenar, facilitando su integración en aplicaciones de inferencia en tiempo real.

**Implementación de la Inferencia en Tiempo Real**

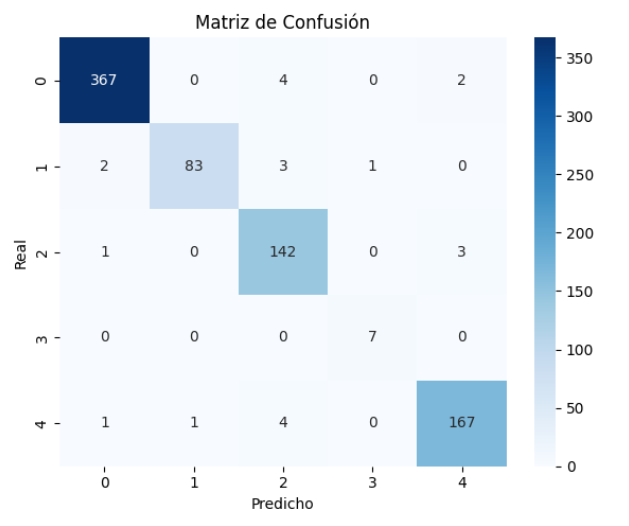
Se desarrolló una interfaz gráfica de usuario (GUI) utilizando Tkinter, que permite al usuario iniciar y detener la detección de actividades en tiempo real a través de la webcam.  
El flujo de inferencia es el siguiente:

* Captura de video en tiempo real.
* Extracción de landmarks con MediaPipe.
* Cálculo de las mismas características utilizadas en el entrenamiento.
* Normalización de las características con el scaler previamente guardado.
* Predicción de la actividad mediante el modelo entrenado.
* Visualización de la actividad detectada y su probabilidad en la interfaz y sobre el video.

Este sistema permite una experiencia interactiva y demuestra la aplicabilidad práctica del modelo desarrollado.

**Resultados**

A. **Matriz de Confusión**



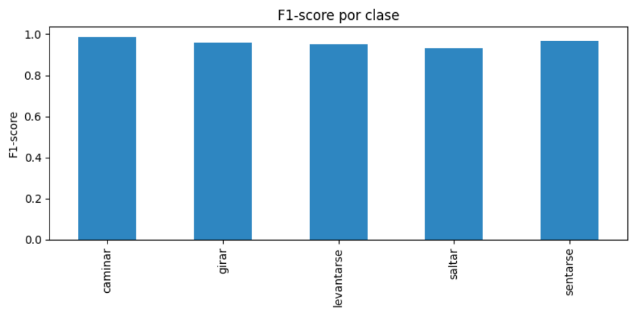
La siguiente matriz de confusión muestra el desempeño del modelo en los datos de prueba. En ella se representan las predicciones del modelo (eje horizontal) frente a las clases reales (eje vertical), donde cada celda indica el número de ejemplos clasificados en esa combinación.

**Interpretación**:

* La clase 0 (caminar hacia la cámara) fue reconocida correctamente en 367 ocasiones, con errores menores hacia las clases 2 (girar) y 4 (levantarse).
* La clase 2 (girar) presenta una precisión elevada con 142 clasificaciones correctas.
* La clase 3 (sentarse) es la más afectada con solo 7 aciertos, posiblemente debido a la similitud postural con otras clases o falta de muestras.
* La clase 4(levantarse) tampoco se ve afectada por las demas clases dando un acierto de 167 al movimiento de levantarse
* El modelo muestra buenas capacidades generales de clasificación, con errores localizados entre actividades con posturas similares.

**Desempeño por Clase (F1-Score)**

Para evaluar la efectividad del modelo en cada tipo de actividad, se calcularon los **F1-score por clase**. La siguiente gráfica ilustra la puntuación alcanzada por cada categoría:



**Interpretación**:

* Las clases caminar, girar, levantarse y sentarse presentan **F1-scores superiores al 0.95**, indicando un balance adecuado entre precisión y recall.
* La clase saltar, aunque no se menciona explícitamente en el conjunto base de actividades, muestra también una buena puntuación. Si esta clase fue generada como experimento, debe aclararse su consistencia dentro del dataset.
* La consistencia en F1-score sugiere que el modelo **no está sesgado hacia ninguna clase en particular**, lo que es una señal positiva para su generalización.

**Conclusiones:**

El desarrollo de este sistema de reconocimiento de actividades humanas mediante aprendizaje automático ha demostrado la viabilidad y eficacia de combinar técnicas de visión por computadora (MediaPipe para extracción de características) con modelos de machine learning (Random Forest) para la clasificación de acciones en video. El proceso de normalización de datos con StandardScaler y la adecuada división de los conjuntos de entrenamiento y prueba permitieron obtener un modelo robusto y generalizable.

Los resultados obtenidos, reflejados en la matriz de confusión y los F1-score por clase, muestran que el modelo es capaz de distinguir correctamente la mayoría de las actividades, alcanzando valores de F1-score superiores a 0.95 en casi todas las clases. Esto indica un excelente equilibrio entre precisión y recall, y sugiere que el sistema puede ser utilizado en aplicaciones prácticas de monitoreo o interacción basada en gestos.

La visualización de la importancia de las características también permitió identificar qué variables son más relevantes para la toma de decisiones del modelo, lo que abre la puerta a futuras optimizaciones y a una mejor comprensión del comportamiento del sistema.

En términos de implementación, la integración del modelo entrenado en una interfaz gráfica de usuario (GUI) con inferencia en tiempo real demuestra la aplicabilidad del sistema en escenarios interactivos, permitiendo la detección y visualización inmediata de las actividades humanas capturadas por la webcam.

Problemáticas encontradas durante la implementación

Durante el desarrollo del proyecto se presentaron diversos desafíos, entre los que destacan:

Desbalance de clases: Algunas actividades, como "sentarse", estaban subrepresentadas en el conjunto de datos, lo que afectó negativamente el desempeño del modelo en esas clases. Esto se reflejó en un menor número de aciertos y F1-score para dichas actividades.

Similitud postural entre actividades: Se observaron errores de clasificación entre actividades con posturas similares, como "sentarse" y "levantarse". Esto sugiere la necesidad de extraer características más discriminativas o aumentar la cantidad de muestras para estas clases.

Calidad y variabilidad de los datos: La calidad de los datos capturados (iluminación, ángulo de la cámara, variabilidad entre sujetos) influyó en la precisión de los landmarks extraídos por MediaPipe, lo que en ocasiones generó ruido en las características.

Procesamiento en tiempo real: La integración de la inferencia en tiempo real requirió optimizar el flujo de procesamiento para evitar retrasos perceptibles en la GUI, especialmente al manejar video y predicciones simultáneamente.

Persistencia y reutilización del modelo: Fue necesario implementar mecanismos robustos para guardar y cargar tanto el modelo como el scaler, asegurando la consistencia entre el entrenamiento y la inferencia.

A pesar de estas problemáticas, el sistema logró cumplir con los objetivos planteados, y las lecciones aprendidas servirán para futuras mejoras, como el aumento y balanceo del dataset, la exploración de modelos más avanzados o la incorporación de técnicas de aumento de datos.