Clasificación de actividades en datos de articulaciones 3D por modelos de aprendizaje de máquina. Proyecto Final UG

Frias Cortez Rafael Alejandro. Ingeniería de Datos e Inteligencia Artificial

Resumen – En este proyecto se presenta un estudio de clasificación de actividades humanas básicas utilizando datos tridimensionales de articulaciones obtenidos mediante sensores Kinect. Se analizaron 25 coordenadas (x, y, z) de las articulaciones del cuerpo durante diez actividades cotidianas, procesando datos con diferentes duraciones y frecuencias de captura (FPS). Para cada actividad, se calcularon ocho distancias referenciadas desde la articulación central de la espina dorsal, y se extrajeron características estadísticas (promedio, varianza, mínimo y máximo) de dichas distancias y sus velocidades. Se entrenaron y evaluaron diversos clasificadores supervisados, agrupados en cuatro bloques: árboles de decisión y sus variantes, diferentes configuraciones de KNN, modelos basados en reglas de Bayes y análisis discriminante lineal, y máquinas de vectores de soporte (SVM) con diferentes núcleos. El rendimiento de los clasificadores se evaluó mediante métricas estándar (exactitud, precisión, recall, F1-score), matrices de confusión, y curvas ROC con el área bajo la curva (AUC).

I. INTRODUCCIÓN

El reconocimiento de actividades humanas usando datos de movimiento es cada vez más importante en áreas como la salud, seguridad y tecnología. Gracias a sensores como Kinect, es posible capturar en 3D las posiciones de las articulaciones del cuerpo mientras una persona realiza distintas acciones.

En este proyecto, se trabaja con datos de 25 articulaciones para clasificar diez actividades básicas, usando dos vistas del Kinect. Para facilitar el análisis, se transforma las coordenadas en ocho distancias clave desde la columna vertebral y se calcula características estadísticas como el promedio y la variación de esas distancias y sus velocidades.

Se prueba varios métodos de clasificación, como árboles de decisión, KNN, modelos bayesianos y máquinas de vectores de soporte, evaluando cuál funciona mejor con estos datos. También se analiza qué pasa si se reduce el número de actividades a clasificar para mejorar el rendimiento.

El objetivo es entender mejor cómo identificar movimientos humanos a partir de datos esqueléticos y proponer mejoras para futuros sistemas de reconocimiento.

II. METODOS

Para cada muestra de actividad, se calculó un conjunto de características a partir de **8 distancias** entre articulaciones del cuerpo humano (codo, muñeca, rodilla, talones. Estas distancias fueron referenciadas al punto central del cuerpo (joint 0). Se calcularon las **velocidades** entre frames.

Para cada distancia y velocidad, se extrajeron las siguientes estadísticas:

- Promedio:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- Varianza:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

- Minimo y maximo:

min(x), max(x)

Clasificadores Utilizados [1]

Bloque 1: Árboles y métodos basados en votación y vecinos

- **Árboles de decisión:** Construyen una serie de reglas para dividir los datos y clasificarlos paso a paso.
- **Random Forest:** Combina varios árboles de decisión para mejorar la precisión y evitar errores.
- **AdaBoost:** Une varios modelos simples (llamados “débiles”), donde cada modelo nuevo corrige los errores del anterior. Al final, todos votan para decidir la clase final.

Bloque 2: Variantes de K-Nearest Neighbors (KNN)

- **KNN Fine**: Usa pocos vecinos cercanos (entre 1 y 3) para decidir la clase, dando mucha importancia a los ejemplos más parecidos.
- **KNN Minkowski:** Mide la distancia entre puntos usando la distancia de Minkowski, que puede ser como contar pasos en línea recta o cuadras, dependiendo del parámetro elegido.
- **KNN Weighted:** Da más peso a los vecinos más cercanos al momento de votar, para que tengan mayor influencia en la decisión.
- **KNN Medium:** Usa la mediana de las distancias para decidir la clase, lo que ayuda a reducir el efecto de valores atípicos.
- **KNN Coarse**: Es una versión de KNN que usa un número alto de vecinos (entre 10 y 100) para decidir la clase, lo que hace la clasificación más general y menos sensible a ruido.
- KNN Cosine: Variante de KNN que no mide la distancia directa, sino el ángulo entre dos vectores (similitud de dirección).

Bloque 3: Modelos probabilísticos y análisis discriminante

- **Naive Bayes Gaussian:** Modelo basado en probabilidades que asume que las características tienen una distribución normal (gaussiana).
- **Naive Bayes Kernel:** Variante que usa funciones kernel para estimar mejor la distribución de las características.
- **LDA (Linear Discriminant Analysis):** Busca un eje o plano que separe lo mejor posible las clases proyectando los datos para que queden bien diferenciados. Funciona bien cuando cada clase está agrupada en una "nube".

Bloque 4: Máquinas de vectores de soporte (SVM) con diferentes núcleos

- **Linear SVM:** Busca una línea (o plano) que separa las clases con una frontera recta.
- **Quadratic SVM:** Usa un kernel cuadrático para permitir una frontera curva de separación.
- **Cubic SVM**: Usa un kernel cúbico, con una curva más compleja para separar las clases.
- **Fifth SVM:** Utiliza un kernel polinomial de grado 5, que permite separar las clases con fronteras aún más flexibles y complejas.

Evaluación

Para cada clasificador se generaron:

- **Matriz de confusión**
  Muestra los aciertos y errores por clase.
- **Métricas de desempeño**
  - **Exactitud (Accuracy)**
  - **Precisión**
  - **Recall (Sensibilidad)**
  - **F1-Score**

Estas métricas permitieron comparar objetivamente el rendimiento de los clasificadores y seleccionar el más adecuado según la actividad a identificar.

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

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

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

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

Imagen 1. Fórmulas de las métricas de desempeño.

Una **matriz de confusión** es una herramienta para evaluar el rendimiento de un clasificador, especialmente cuando estamos tratando con múltiples clases. Para una **matriz de confusión de 3x3**, cada fila representa una **clase real**, y cada columna representa una **clase predicha**. La idea principal de las matrices es comparar las **predicciones** del modelo con las **etiquetas reales** para cada clase.

En una **matriz de confusión 3x3**, como hay tres clases, tendrás tres filas y tres columnas, cada una correspondiente a las tres clases:

	Predicha 0	**Predicha 1**	**Predicha 2**
Real 0	TP_00	FP_01	FP_02
Real 1	FN_10	TP_11	FP_12
Real 2	FN_20	FN_21	TP_22

Tabla 1. Explicación de matrices de confusión.

Elementos de la matriz de confusión

- **TP (True Positive)**: Son los casos en los que **el modelo predijo correctamente** la clase. En la matriz, esto se encuentra en la diagonal principal,
  - En **TP_00** (donde el valor real y el predicho es 0), representa el número de veces que el modelo predijo correctamente la clase 0.
- **FP (False Positive)**: Son los casos en los que el modelo **predijo incorrectamente** una clase, asignando una etiqueta errónea cuando la clase real era diferente.
  - En **FP_01** (donde la clase real era 0, pero el modelo predijo 1), es el número de veces que la clase 0 fue confundida con la clase 1.
- **FN (False Negative)**: Son los casos en los que el modelo **no logró predecir correctamente** una clase real, predijo una clase incorrecta.
  - En **FN_10** (donde la clase real era 1, pero el modelo predijo 0), representa cuántas veces el modelo no predijo correctamente la clase 1.
- **TN (True Negative)**: No aparece explícitamente en la matriz, pero se puede calcular como **todos los otros valores que no están en la fila y columna de la clase en cuestión**. Los verdaderos negativos son todas las instancias que **no pertenecen a la clase actual**, y fueron correctamente predichas como no pertenecientes a esa clase.

III. RESULTADOS

Para comenzar con la práctica, se realizó la lectura de los archivos .txt correspondientes a cada acción realizada por cada persona. Estos archivos contenían las posiciones de los **joints** (articulaciones) capturados en cada frame.
La lectura se organizó utilizando rutas hacia las carpetas correspondientes y se guio mediante un archivo .csv que indicaba los **fragmentos relevantes** (frame de inicio y de fin) para cada acción. Se extrajeron los puntos necesarios de cada secuencia de manera estructurada.
Se implementó una visualización en formato **2D** del esqueleto humano a partir de los puntos extraídos (Imagen 1), con el fin de observar el movimiento realizado en cada acción. Para construir estas visualizaciones, se definieron conexiones entre los joints clave, tales como:

- Columna ↔ Cabeza
- Hombros ↔ Codos ↔ Muñecas
- Cadera ↔ Rodillas ↔ Talones

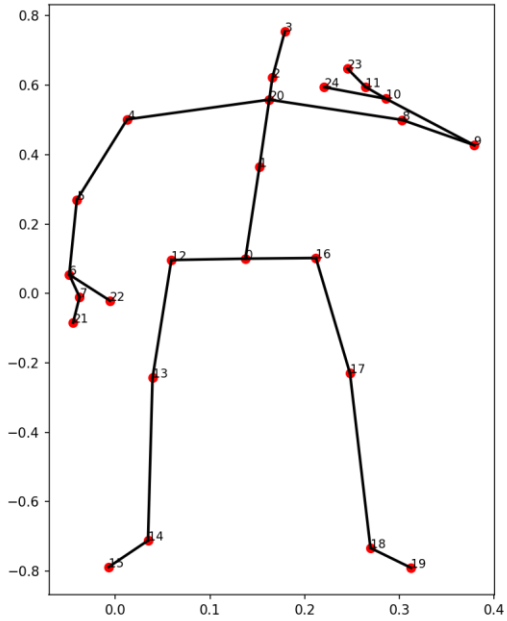

Imagen 1. Visualizacion 2D conexiones Kinect.

Estas uniones se aplicaron para ambos lados del cuerpo (izquierdo y derecho), permitiendo representar de forma clara la postura y el movimiento del sujeto en cada actividad.
Antes de calcular características como distancias y velocidades, se aplicó un **suavizado por promedio móvil** a las posiciones de las articulaciones (joints) para reducir el ruido en los datos capturados por el sensor.

Se implementa una función suavizar_joints, que recorre todos los frames de la secuencia y reemplaza la posición de cada frame por el **promedio de sus vecinos cercanos**.
Esto ayuda a suavizar pequeños errores o fluctuaciones que puedan haber ocurrido en la captura del movimiento.
Funcionamiento:

- Para cada frame, se toma una **ventana** de tamaño definido (3 y 5 frames).
- Se promedian las posiciones de los joints dentro de esa ventana.
- El frame actual se reemplaza por ese promedio.
- El resultado es una secuencia más estable y continua, ideal para análisis de movimiento.

Ventajas:

- Reduce el **ruido de alta frecuencia** (saltos bruscos o errores de medición).
- Mejora la **precisión de características derivadas**, como velocidades y distancias.
- Es un método no invasivo que **preserva la forma general del movimiento**.

Ya con el calculo de las distancias y las velocidades de dichas distancias se hizo el cálculo de las características para posteriormente juntar todas las características de las acciones de cada persona, para poder hacer uso de los clasificadores.

Preparación de Datos para el Entrenamiento

Una vez obtenidas las características de cada muestra de actividad, se procedió a preparar los datos para el entrenamiento de los clasificadores. Este proceso incluyó los siguientes pasos:

1. **Carga del Conjunto de Datos**
   Se lee el archivo CSV caracteristicas_completo_filtro.csv, el cual contiene todas las muestras extraídas, con sus respectivas etiquetas (actividad realizada) y datos complementarios (persona y repetición).
   - **Dimensiones del CSV: (432, 67)**
2. **Selección de Variables**
   Se eliminaron las columnas no numéricas (persona, repeticion, actividad) para obtener solo las características cuantitativas que se utilizarán como entrada para los modelos. La columna actividad se conservó como etiqueta (y).
3. **Codificación de Etiquetas**
   Dado que los clasificadores trabajan con valores numéricos, las etiquetas categóricas correspondientes a las actividades fueron transformadas a valores enteros utilizando **Label Encoding**.
4. **División de Datos**
   Se dividió el conjunto total en dos subconjuntos:
   - **70% para entrenamiento**
   - **30% para prueba (test)**
   - **Total de muestras: 432**
   - **Datos de entrenamiento: 302**
   - **Datos de prueba: 130**

Esta división se realizó con la función train_test_split de sklearn, utilizando una **estratificación** basada en las etiquetas para garantizar que todas las clases estén representadas proporcionalmente en ambos subconjuntos.

5. Se hizo la primera prueba en los clasificadores con una suavidad en los datos de 3 ventanas, obtuvo mejores desempeños en unos clasificadores
6. En la segunda prueba se hizo con un suavizado móvil de 5 ventanas y dio el mejor desempeño en Random Forest
7. Se hizo clasificaron los modelos con las 10 actividades (sentarse, pararse, aplaudir, saludar, martillar, dibujar, agarrar teléfono, tomar vaso de agua y beber, mirar la hora, caminar) con las 10 actividades dieron un desempeño decente.

Eliminación de actividades.

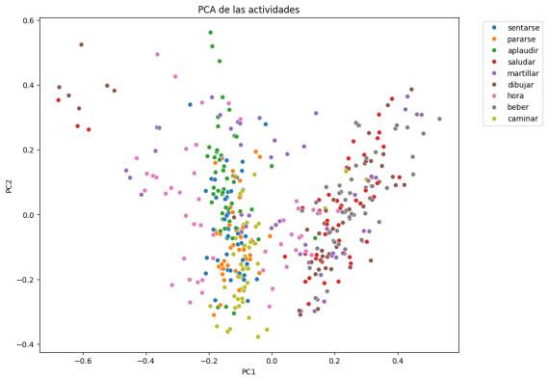

Imagen 2. Soplamiento PCA

Con este método se observa que algunas actividades están demasiado juntas o con una distancia muy cercana, se hizo el calculo de cuales actividades tenían distancias mas cercas para hacer pruebas de eliminación de ellas.

Pararse – sentarse	0.03
Dibujar – saludar	0.05
Hora – sentarse	0.05
Dibujar – martillar	0.08
Hora - pararse	0.08

Tabla 2. Distancias en el soplamiento.

Se hicieron las pruebas y si dieron mejores resultados eliminando ambas actividades, se hizo prueba con 3 ventanas que es el que obtuvo más desempeño a comparación de con 5 ventas.

Matriz de correlación.

Se realizo una matriz de correlación para una segunda prueba para eliminación de actividades, para ver otro método de como eran similares las actividades, las actividades teléfono y beber obtuvieron una correlación de 0.96.

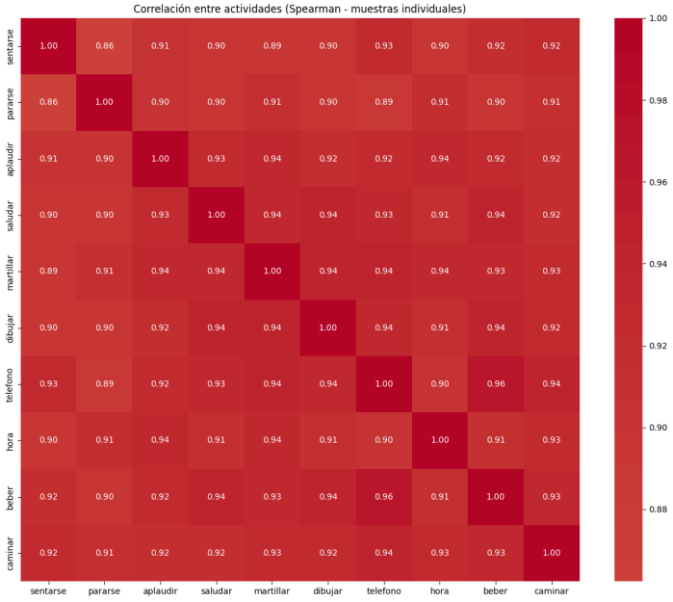

Imagen 3. Matriz de correlación.

Con la eliminación de la actividad mejoraron mucho los clasificadores con un 85% cuando en la primera prueba del suavizado móvil con 5 ventanas, con la segunda prueba con un suavizado de 3 ventanas mejoro a 86%.

Resultados de los modelos:

Clasificador	Accuracy	Precision	Recall	F1-Score
Arboles de decisión Random Forest	86%	87%	86%	84%
KNN Coseno	76%	77%	76%	76%
Naive Bayes LDA	80%	81%	80%	80%
SVM Cubico	42%	35%	42%	35%

Tabla 3. Resultados por bloques

RANDOM FOREST

Matriz de Confusión

La matriz de confusión muestra un desempeño sólido con alta precisión en la mayoría de las clases.

- Clases como aplaudir, beber, caminar, sentarse, y saludar fueron clasificadas casi perfectamente.
- El mayor problema se observa en la clase martillar, donde el modelo cometió varios errores de clasificación, confundiéndola con clases como dibujar y hora.
- Esto podría deberse a similitud entre patrones de movimiento o datos insuficientes.
- En general, el modelo muestra buen desempeño con pocas confusiones importantes.

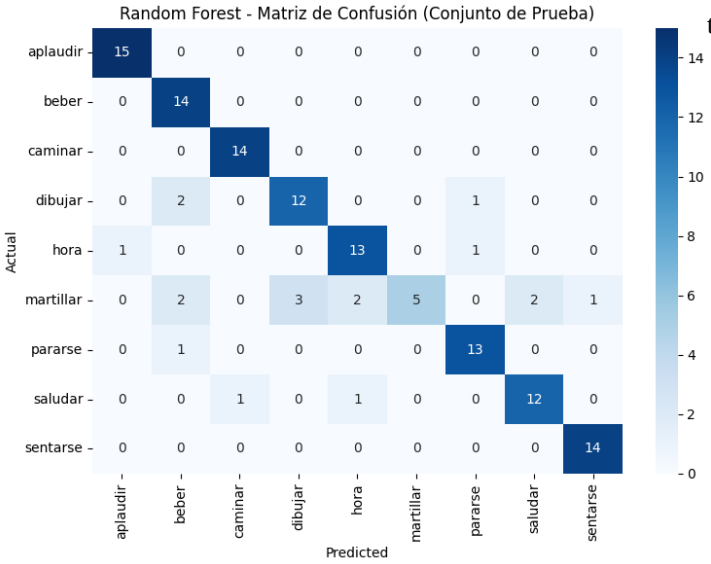

Imagen 4. Matriz de confusión de prueba

Curva de Aprendizaje

La curva de aprendizaje revela un comportamiento esperable:

- El score de entrenamiento se mantiene alto (por encima del 90%) incluso con pocas muestras, lo cual es típico en Random Forest debido a su capacidad para ajustarse bien a los datos.
- El score de validación va mejorando conforme aumenta el número de muestras, alcanzando una exactitud cercana al 76% con todo el conjunto.
- Aunque hay una ligera diferencia entre entrenamiento y validación, no hay un sobreajuste grave. Aumentar el conjunto de datos podría cerrar aún más esa brecha.

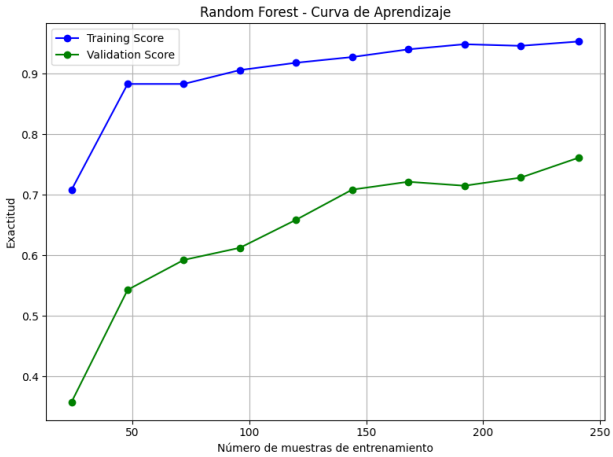

Imagen 5. Curva de aprendizaje

Curva ROC Promedio

La curva ROC muestra un rendimiento excelente en términos de capacidad discriminativa:

- Tanto el micro-promedio como el macro-promedio tienen un área bajo la curva (AUC) de 0.97, lo cual indica que el modelo distingue bien entre las clases.
- Esto sugiere que incluso en presencia de clases desbalanceadas, el clasificador mantiene un buen comportamiento general.

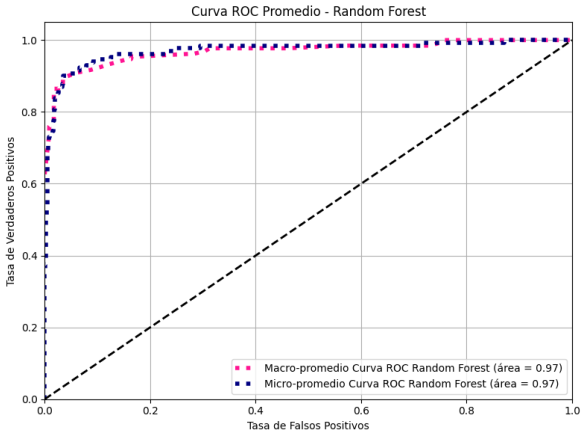

Imagen 6. Curva ROC

- **Accuracy – 86%:** El modelo acierta en la mayoría de los casos.
- **Precisión – 87%:** Cuando predice una clase, suele acertar (pocos falsos positivos).
- **Recall – 86%:** Detecta correctamente la mayoría de los casos reales (pocos falsos negativos).
- **F1-Score – 84%:** Buen equilibrio entre precisión y recall.

KNN Coseno

Matriz de Confusión

La matriz de confusión evidencia el desempeño del modelo sobre las diferentes clases.

- Clases como **"aplaudir"**, **"caminar"**, **"hora"** y **"saludar"** fueron reconocidas con alta precisión (mayoría de aciertos en la diagonal).
- Las clases **"dibujar"**, **"sentarse"** y **"pararse"** presentan más confusión. Por ejemplo:
  - "dibujar" se confunde con "hora" y "martillar".
  - "sentarse" es confundida frecuentemente con "pararse".
- Esto indica que ciertas clases tienen características similares en el espacio reducido por PCA o no están bien separadas bajo la métrica del coseno.

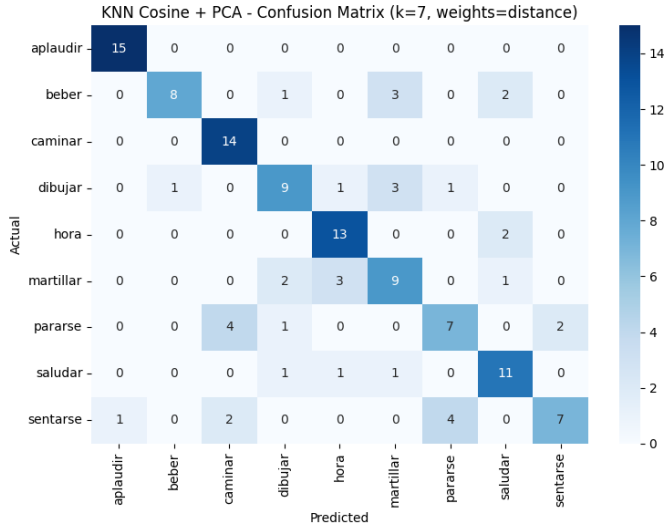

Imagen 7. Matriz de confusión de prueba

Curva de Aprendizaje

La curva muestra cómo se comporta el modelo al aumentar la cantidad de datos de entrenamiento:

- **Puntaje de entrenamiento:** cerca de 1.0 (100%), lo que es típico en KNN porque memoriza los datos.
- **Puntaje de validación:** empieza bajo (~0.4) pero mejora de manera constante hasta ~0.75.
- Existe un **gap entre entrenamiento y validación**, lo que indica **sobreajuste (overfitting)**.
- La tendencia ascendente de la curva de validación sugiere que **más datos podrían mejorar aún más el rendimiento**.
- El modelo se beneficia del crecimiento del conjunto de entrenamiento.

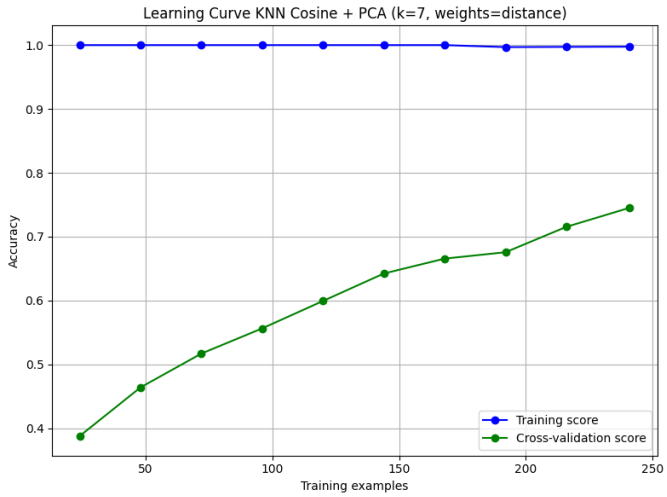

Imagen 8. Curva de aprendizaje

Curva ROC Promedio

El área bajo la curva (AUC) es de **0.87**, lo cual es **muy bueno para un clasificador multiclase**, el clasificador tiene una **alta capacidad de discriminación entre clases**.

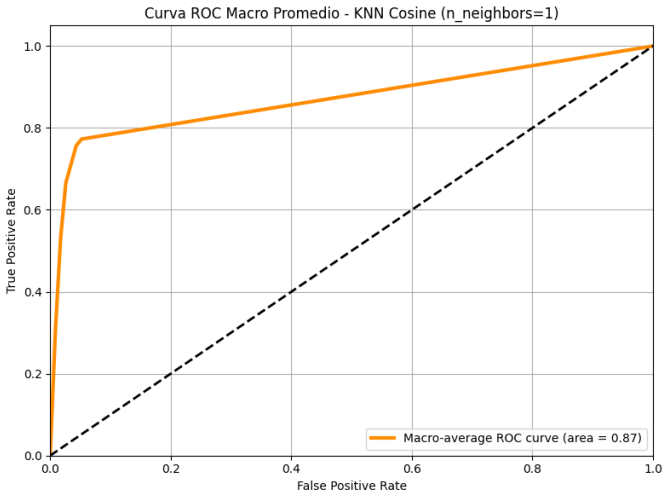

Imagen 9. Curva ROC

LDA

Matriz de Confusión

El modelo realiza en general una buena clasificación, con la mayoría de las predicciones correctas concentradas en la diagonal.

Hay algunas confusiones entre ciertas clases, por ejemplo:

- Clase 3 se confunde con 5 y 7.
- Clase 5 se confunde varias veces con clases 0, 3 y 7.
- Clase 8 se confunde con clase 2 y 7.

La cantidad de errores es baja, lo cual indica un rendimiento general sólido.

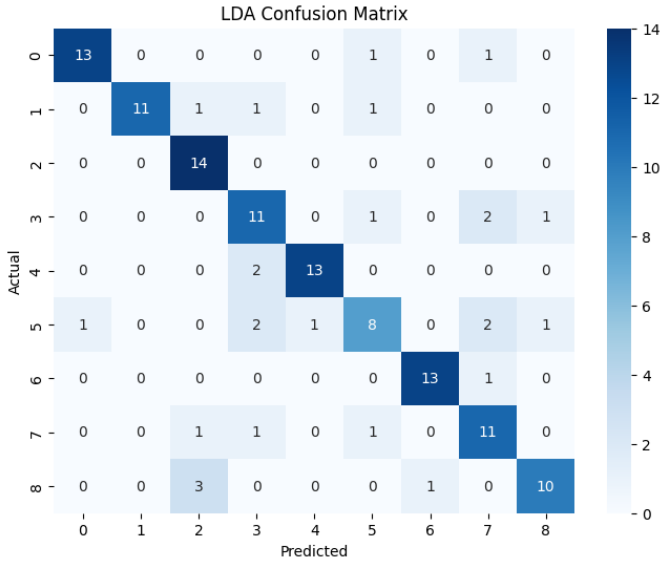

Imagen 10. Matriz de confusión de prueba

Curva de Aprendizaje

Entrenamiento: muy alto (cercano a 1), lo cual es consistente con un buen ajuste del modelo a los datos de entrenamiento.

Validación cruzada:

- Mejora progresiva conforme se incrementa la cantidad de datos de entrenamiento.
- Aumenta hasta ~0.73, lo que indica que el modelo aún está aprendiendo y podría beneficiarse de más datos.
- La brecha entre entrenamiento y validación sugiere cierta **varianza** (riesgo de sobreajuste), pero es más estable que en el primer caso.

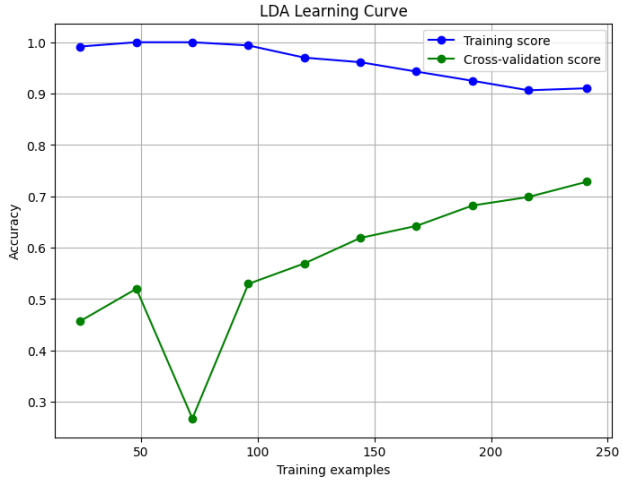

Imagen 11. Curva de aprendizaje

Curva ROC Promedio

AUC = 0.97, excelente resultado.

El modelo tiene alta capacidad de discriminación entre clases.

Curva pegada al eje superior izquierdo, indicando alta sensibilidad y especificidad global.

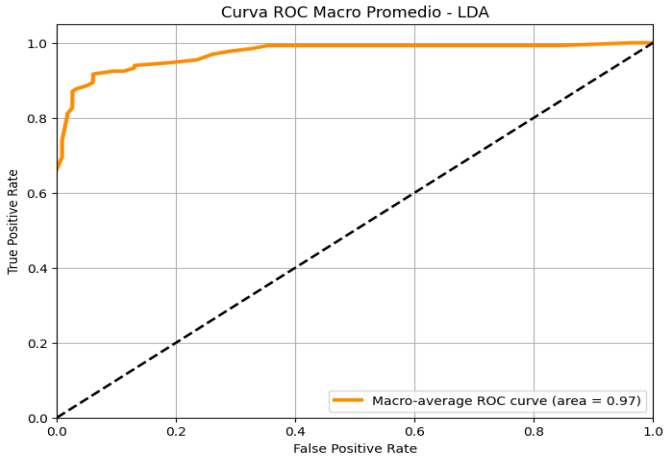

Imagen 12. Curva ROC

SVM Cubic

Matriz de Confusión

La diagonal muestra los aciertos para cada clase.

Hay **confusión significativa** entre algunas clases, como:

- Clase 4 se confunde con clase 6 (5 veces) y clase 8 (2 veces).
- Clase 5 tiene predicciones dispersas: clases 0, 1, 2, 4 y 7.

El modelo no está diferenciando bien entre ciertas clases, lo que podría deberse a características similares entre ellas o a un modelo poco expresivo con los datos actuales.

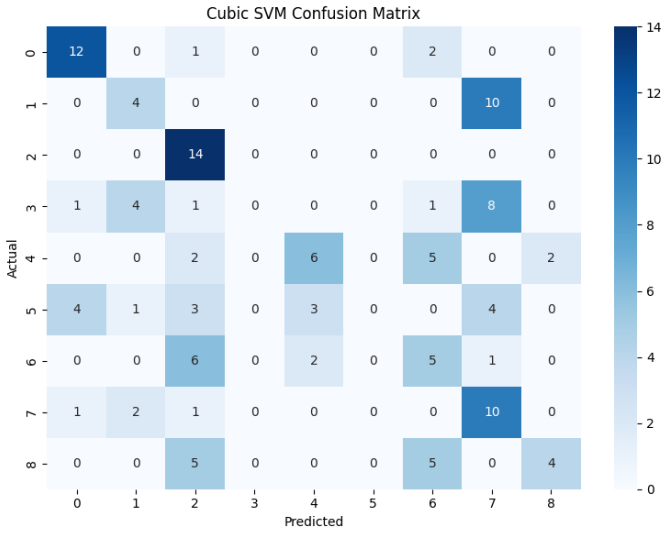

Imagen 13. Matriz de confusión de prueba

Curva de Aprendizaje

Tendencias:

- El **score de entrenamiento** crece hasta cerca de **0.50**, pero no alcanza un valor alto.
- El **score de validación cruzada** mejora con más datos y alcanza alrededor de **0.44**, pero aún es bajo.

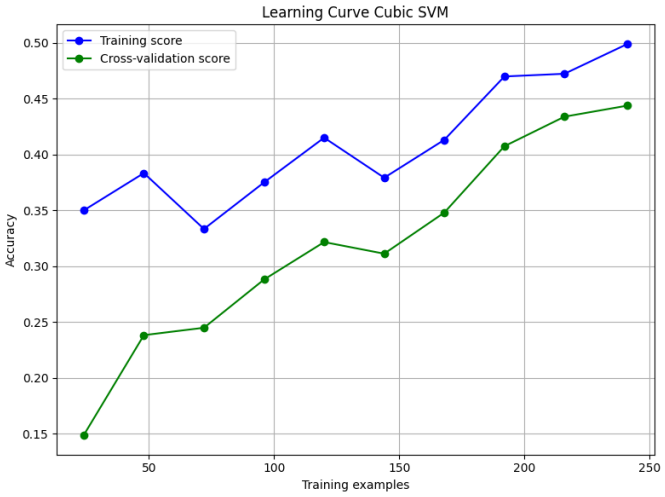

Imagen 14. Curva de aprendizaje

Curva ROC Promedio

- **AUC = 0.85**, lo cual es **bueno** y sugiere que el modelo tiene una **capacidad decente de clasificación global**.
- El AUC alto no siempre refleja precisión por clase, especialmente si las clases están desbalanceadas (lo cual podría ser el caso aquí).

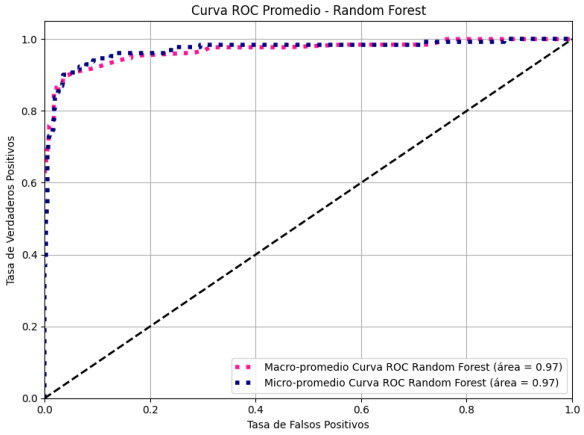

Imagen 15. Curva ROC

IV. DISCUSION

Tras realizar el procesamiento de datos, extracción de características y el suavizado por promedio móvil (con ventanas de tamaño 3 y 5), se entrenaron y evaluaron varios clasificadores para identificar actividades humanas a partir de datos de esqueleto capturados por Kinect. Los clasificadores utilizados fueron: Random Forest, KNN (coseno), LDA y SVM cúbico, cada uno con sus propios hiperparámetros, los cuales impactaron directamente en su rendimiento.

- **Random Forest**

Este modelo fue el que mejor desempeño presentó en el conjunto de pruebas, alcanzando una exactitud del 86%, con una precisión del 87%, un recall del 86% y un F1-score del 84%. Se utilizó con los siguientes parámetros:

- n_estimators = 200: se usaron 200 árboles, lo que permite una votación más robusta y reduce la varianza del modelo.
- max_depth = 10: se limitó la profundidad máxima de los árboles para evitar sobreajuste y mejorar la capacidad de generalización.
- min_samples_split = 10: cada nodo se divide solo si hay al menos 10 muestras, ayudando a evitar divisiones innecesarias.
- min_samples_leaf = 4: asegura que cada hoja tenga al menos 4 muestras, lo que reduce el riesgo de ramas sobreajustadas.

- max_features = 'sqrt': en cada división se considera la raíz cuadrada del número total de características, lo que es común en Random Forest y mejora la diversidad entre los árboles.
- bootstrap = True: se activó el muestreo con reemplazo, lo cual es estándar para este algoritmo y mejora la robustez.

Estos parámetros fueron elegidos para lograr un equilibrio entre precisión y generalización, logrando evitar tanto el sobreajuste como la subutilización de los datos.

- **KNN (métrica del coseno)**

El clasificador KNN logró un rendimiento aceptable, con un 76% de accuracy y un AUC de 0.87. Se utilizó la métrica del coseno como medida de similitud. Esta métrica es útil cuando lo importante es la dirección del movimiento y no tanto la magnitud, lo cual es relevante en datos de esqueleto humano. No se requiere entrenamiento como tal, ya que es un algoritmo basado en vecinos.
La métrica de distancia coseno permite medir similitud angular, lo cual funcionó mejor que la distancia euclidiana en este caso.
El modelo mostró sobreajuste: la exactitud en entrenamiento fue muy alta (~100%) pero la de validación fue menor, indicando que memoriza los datos pero generaliza peor. Esto es característico de KNN cuando no se regula bien el número de vecinos o si hay ruido en los datos.

- **LDA (Linear Discriminant Analysis)**

El modelo LDA ofreció un rendimiento sólido con una exactitud del 80%. No se modificaron hiperparámetros, ya que LDA no requiere gran ajuste:

- Funciona asumiendo que las clases siguen una distribución normal y comparten una misma matriz de covarianza.
- Es adecuado para problemas linealmente separables y tiene una capacidad de generalización razonable si estas condiciones se cumplen.
- La curva ROC mostró un AUC de 0.97, lo que indica una excelente capacidad discriminativa. Este rendimiento es indicativo de que los datos, aunque no perfectamente lineales, sí tienen una estructura que LDA puede aprovechar.

- **SVM Cúbico (polinomial grado 3)**

El modelo SVM con un kernel polinómico de grado 3 (cúbico) tuvo el rendimiento más bajo, con una exactitud de apenas 42%. Se utilizaron los siguientes parámetros:

- kernel = 'poly': se seleccionó el núcleo polinomial para capturar relaciones no lineales.
- degree = 3: un polinomio de grado 3 permite modelar interacciones cúbicas entre las características.
- probability = True: se habilitó el cálculo de probabilidades para el análisis ROC, aunque esto aumenta el tiempo de entrenamiento.

A pesar de un AUC de 0.85, la clasificación fue deficiente para muchas clases. El modelo no logró aprender patrones representativos, posiblemente debido a:

- La complejidad del kernel cúbico sin un buen ajuste del parámetro C o gamma.
- La sensibilidad del SVM a la escala de los datos y la posible necesidad de normalización, que en este caso no fue aplicada.

Clasificador	**Accuracy**	**AUC**	**Comentario clave**
Random Forest	86%	0.97	Mejor desempeño general
KNN (coseno)	76%	0.87	Aceptable, pero propenso a sobreajuste
LDA	80%	0.97	Estable y buena discriminación
SVM (cúbico)	42%	0.85	Bajo desempeño, requiere mayor ajuste

V. CONCLUSION

El mejor modelo resultó ser **Random Forest**, debido a su **robustez ante el ruido**, su **buena generalización** y la capacidad de **manejar datos no lineales sin necesidad de escalar o transformar características**. Los resultados también reflejan que la **eliminación de actividades similares** y la **aplicación de un suavizado móvil** (ventana de 5 frames) fueron decisiones acertadas para mejorar el rendimiento del sistema de clasificación.
Este proceso me permitió **comprender la importancia de la ingeniería de características y la selección de modelos adecuados**, así como el impacto que tienen los ajustes de preprocesamiento y parámetros en el desempeño final de un sistema de aprendizaje automático.

Referencias

[1] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.