Resumen técnico del proyecto

Asistente coreográfico inteligente

1) Datos

- **Fuente**: keypoints corporales extraídos de vídeo (YOLOv8-Pose / MediaPipe).
- **Unidad de análisis**: ventana temporal (p. ej., 5 s con hop 2.5 s) por vídeo.
- **Features** (numéricas agregadas por ventana):
  `amplitud_x`, `amplitud_y`, `amplitud_z`, `velocidad_media`, `simetria`, `nivel_alto`, `nivel_bajo`, `nivel_rango`, `variedad_direcciones`.
  Metadatos: `filename`, `filepath`, `frames`, `joints=17`, `dims=3`.
- **Preprocesado**:
  - Interpolación/forward-backfill de NaNs por articulación y eje.
  - Normalización (imputación + escalado) antes del modelo.
  - Generación de ventanas con `start_f/end_f` y `start_s/end_s`.
- **Etiquetas (multietiqueta)**:
  `amplitud_baja`, `variedad_baja`, `mucha_simetria`, `poca_simetria`, `fluidez_baja`, `poco_rango_niveles`, `exceso_rango_niveles`, `frase_corta`.
  Cada etiqueta se mapea a **sugerencias coreográficas** (texto) para la app.

Nota: el dataset base está en `data/processed/aistpp/features_coreograficos.csv` y las ventanas/timelines se exportan a `reports/`.

Cuando empecé a estructurar el corpus para este asistente coreográfico, asumí algo sencillo pero exigente: **casi todo lo que importa está en el movimiento**. Y el movimiento, para poder analizarlo con rigor, hay que capturarlo bien. Por eso la **fuente primaria** de datos son *keypoints* corporales extraídos directamente de vídeo mediante dos backends complementarios —**YOLOv8-Pose** y **MediaPipe**—. No siempre es fácil: la iluminación, la ropa o la velocidad del giro pueden sabotear una estimación; aun así, he observado que combinar ambos enfoques reduce fallos gruesos y, sobre todo, estabiliza los casos "difíciles" (saltos, oclusiones, diagonales largas).

Unidad de análisis: ventanas temporales

En lugar de etiquetar fotograma a fotograma —ruidoso y, en la práctica, poco útil para la coreografía—, opté por trabajar con **ventanas temporales**. Cada vídeo se trocea en tramos de, por ejemplo, **5 segundos** con un *hop* de **2,5 segundos**. Esta granularidad tiene un sentido coreográfico: cinco segundos suelen contener una micro-frase; dos y medio permiten solapamiento suficiente para no "cortar" una intención a la mitad. Me pregunto si, en repertorios con tempi extremos, convendría adaptar dinámicamente esta longitud; es una línea futura interesante.

Features: lo que medimos y por qué

Sobre cada ventana calculo un conjunto de **características numéricas** que resumen el gesto sin perder su sustancia:

- **amplitud_x, amplitud_y, amplitud_z**: rangos espaciales por eje. X e Y anclan la expansión lateral y vertical; Z captura profundidad cuando hay 3D disponible.
- **velocidad_media**: norma media del desplazamiento entre fotogramas; una señal compacta de energía y fluidez.
- **simetria**: cuánto se parecen entre sí los pares izquierda/derecha (hombros, codos, muñecas, caderas, rodillas, tobillos). En piezas con contrapesos, esta métrica cuenta una historia.
- **nivel_alto, nivel_bajo, nivel_rango**: percentiles del centro de masas (aprox. caderas) y su diferencia; ayudan a saber si se está explorando el espacio vertical o se permanece en una meseta.
- **variedad_direcciones**: cambios promedio de dirección en la trayectoria; cuando cae, la frase tiende a "alinearse" y perder riqueza.

Acompaño estas señales con **metadatos** útiles para trazabilidad: `filename`, `filepath`, y medidas estructurales (`frames`, `joints=17`, `dims=3`). La constancia en `joints` y `dims` simplifica validaciones y previene sorpresas en el *pipeline*.

Preprocesado: hacer habitable el dato

El dato en crudo, confieso, es terco. Ocurre con frecuencia que un codo desaparezca tres fotogramas o que una muñeca se "salga" del plano. Para **suavizar estas faltas** aplico un esquema pragmático:

1. **Interpolación/forward-backfill de NaNs** por articulación y eje. Primero relleno desde el pasado inmediato, luego desde el futuro, y finalmente interpolo linealmente donde haga falta. En secuencias muy rotas, prefiero marcar la articulación como no fiable antes que inventar.
2. **Normalización** previa al modelo: **imputación** (SimpleImputer) y **escalado** (StandardScaler). Esto reduce el sesgo por escala y evita que una amplitud excepcional se coma el resto.
3. **Generación de ventanas** con índices en fotogramas (`start_f`, `end_f`) y su espejo en segundos (`start_s`, `end_s`). Me ha resultado útil para conciliar métricas técnicas con la vivencia musical de quien baila.

Esquema de etiquetado (multietiqueta)

El problema no es "¿qué clase es?", sino **"¿qué aspectos coexisten?"**. Un mismo tramo puede mostrar gran simetría y, a la vez, poca variedad direccional. Por eso el marco es **multietiqueta**, con el siguiente conjunto operativo:

- `amplitud_baja`, `variedad_baja`, `mucha_simetria`, `poca_simetria`, `fluidez_baja`, `poco_rango_niveles`, `exceso_rango_niveles`, `frase_corta`.

Cada etiqueta no es un juicio final, sino un **disparador de sugerencias**: textos breves y accionables que la aplicación muestra al intérprete o al docente (por ejemplo, *"introducir cambios de dirección y diagonales"* cuando

cae `variedad_direcciones`). He notado que esta capa textual, por simple que parezca, cambia la recepción del sistema: del diagnóstico seco al **acompañamiento pedagógico**.

Rutas y reproducibilidad

Para facilitar réplica y auditoría, el **dataset base** se conserva en:

```
/content/drive/MyDrive/asistente_coreografico/data/processed/aistpp/features_coreograficos.csv
```

y las **ventanas / timelines** se exportan a `reports/`. Puede parecer un detalle menor, pero cuidar estas rutas —y documentarlas— ahorra horas de dudas y versiones cruzadas.

Este capítulo de datos intenta equilibrar dos pulsos: el **rigor técnico** (que le pide a la señal ser estable, comparable, normalizada) y la **sensibilidad coreográfica** (que le pide al sistema no violar la intención de una frase ni aplanar su expresividad). No siempre es fácil; a veces el algoritmo "ve" simetría donde yo percibo diálogo entre lados. Pero precisamente ahí, en esa fricción, el proyecto crece: afinando *features*, ajustando ventanas, y escuchando lo que el cuerpo —a través de los keypoints— intenta decirnos.

---

2) Objetivo

Entrenar un **clasificador multietiqueta** que, dado el vector de *features* de una **ventana temporal** de un vídeo de danza, **prediga déficits o focos de mejora coreográficos** y emita **sugerencias interpretables** para la app. El sistema no pretende dictar una "verdad" única, sino **ofrecer feedback práctico por tramo** —breve, accionable y comprensible— que ayude a afinar decisiones en ensayo y docencia.

Justificación coreográfica.
La coreografía trabaja con capas simultáneas: **espacio** (amplitud, niveles, direcciones), **tiempo** (flujo, fraseo), **energía**(dinámicas) y **cuerpo** (simetrías/asimetrías, transferencia de peso). En la práctica, varias cuestiones coexisten y se influyen mutuamente: una frase puede mostrar **poca variedad direccional**, a la vez que **exceso de simetría** y **rango vertical limitado**. Por eso un enfoque de **clasificación multietiqueta** encaja de forma natural: no obligamos a "elegir una sola falta", sino que reflejamos la **concurrencia real** de aspectos a trabajar, tal como ocurre en una corrección de sala.

Además, evaluar por **ventanas temporales** (p. ej., 5s con solapamiento) es coherente con el **material coreográfico**: muchas intenciones se condensan en microfrases. El bailarín no necesita una corrección por frame, necesita una **pista clara en el momento oportuno**. Ventanear permite al sistema "escuchar" la continuidad del gesto y, al mismo tiempo, entregar **señales puntuales** que acompañen el ensayo sin abrumar.

Las *features* elegidas —**amplitud**, **niveles**, **variedad de direcciones**, **simetría**, **fluidez/velocidad**— no son caprichosas: dialogan con categorías que **enseñamos y medimos** a diario. Si la **amplitud** cae, a menudo la presencia escénica se estrecha; si el **nivel_rango** se aplana, se pierden contrastes verticales; si la **variedad_direcciones** desciende, el recorrido espacial se empobrece. He observado que cuando estas señales se presentan con **lenguaje claro** ("introducir diagonales", "equilibrar simetría con momentos de asimetría"), la retroalimentación deja de ser abstracta y se **vuelve entrenable**.

La **interpretabilidad** no es un lujo: es una condición de uso. Elegimos un modelo con salida legible (OVR con regresión logística y umbrales por etiqueta) para **vincular cada sugerencia a patrones de movimiento** reconocibles. Esto importa pedagógicamente (el docente necesita explicar el "por qué") y éticamente (evitamos cajas negras que normalicen un único canon estético). El sistema debe **ayudar**, no imponer un estilo. Por eso mantenemos umbrales ajustables por etiqueta y contemplamos calibración: cada compañía, cada pieza, puede "afinar" la sensibilidad del detector a su realidad.

En términos de **impacto práctico**, el objetivo persigue tres beneficios:

1. **Reducir la carga cognitiva** del ensayo, externalizando la detección de patrones repetidos (p. ej., frases que sistemáticamente pierden rango).
2. **Aumentar la consistencia** de la retroalimentación entre sesiones y docentes.
3. **Acelerar la iteración coreográfica**, ofreciendo micro-sugerencias mientras el material aún está vivo y puede transformarse.

Soy consciente de los límites: la danza es **contextual y estética**; no todo "déficit" es un error —a veces es una **elección**. Por eso la meta no es un *veredicto*, sino un **acompañamiento informado**. El clasificador señala dónde mirar; la última palabra la tiene la mirada artística.

En suma, el objetivo —predecir focos de mejora y traducirlos a sugerencias útiles por ventana— se justifica porque **respeta la naturaleza multicapas del hecho coreográfico**, se alinea con el **ritmo real del ensayo** y ofrece una **inteligencia práctica**: menos juicio abstracto, más pistas concretas para que el movimiento crezca.

---

3) Enfoque / Modelado

- **Arquitectura**: `OneVsRestClassifier(LogisticRegression)` dentro de un **Pipeline** con `SimpleImputer` + `StandardScaler`.
- **Entrenamiento**: `train_test_split` + validación (se observa uso de *cross-validation*).
- **Decisión**: umbrales por etiqueta desde `thresholds.json` (fallback a umbral global).
- **Salida**: etiquetas activas + probabilidades/scores (si el estimador expone `predict_proba` o se usa `decision_function` con sigmoide).

- **Export de artefactos**:
  `imuter.joblib`, `scaler.joblib`, `model_ovr_logreg.joblib`, `labels.json`, `thresholds.json`.
  Evaluación
  a `artifacts/eval_overall.json` y `artifacts/eval_per_label.csv` + gráficas.

1. Arquitectura elegida

Modelo base.
Opto por un **One-Vs-Rest (OVR) con Regresión Logística** encapsulado en un `Pipeline` junto con **`SimpleImputer`** (para ausentes) y **`StandardScaler`** (para homogeneizar escalas). La elección no es casual:

- **Multietiqueta real**: en coreografía, varias "faltas" o focos conviven. OVR permite un clasificador binario **por etiqueta** que decide de forma independiente y concurrente.
- **Interpretabilidad**: la logística expone **coeficientes** por feature y etiqueta. Puedo decir *qué señal empuja* hacia "variedad_baja" o "poca_simetria". En una práctica artística, explicar **por qué** el sistema sugiere algo no es un lujo, es un pacto ético.
- **Eficiencia y robustez**: entrenar y servir OVR+LogReg es **rápido** (útil en ensayo) y estable con pocos datos por clase; modelos más complejos rinden bien, pero suelen exigir más calibración y computo.

Pipeline completo para evitar fugas.
Incluir **imputación** y **escalado** dentro del `Pipeline` garantiza que los pasos de preprocesado se **ajusten solo con entrenamiento** y se **apliquen** después a validación y test sin "ver" el futuro. En multietiqueta, las fugas (leakage) degradan la confianza: un centímetro de trampa arruina un kilómetro de métrica.

---

2. Entrenamiento y validación

Partición.
Uso un `train_test_split` inicial para aislar test y, **dentro del entrenamiento**, una **validación** (con *cross-validation* observada en el cuaderno). Idealmente, la CV debería ser **estratificada por vídeo**: no mezclar ventanas del mismo vídeo entre train/val evita que el modelo aprenda "tics" del material en lugar de generalizar la noción de *fluidez* o *variedad*.

Desbalance y regularización.
Las etiquetas no aparecen con igual frecuencia. Dos decisiones prácticas:

- **`class_weight="balanced"`** (o ponderaciones por etiqueta) para no "olvidar" clases raras.
- **Regularización L2** en la logística y ajuste del hiperparámetro `C` (o en su defecto explorar un barrido simple). No siempre persigo el F1 global más alto; prefiero

un **balance** razonable entre clases que evite sugerencias sistemáticamente sesgadas.

Búsqueda ligera.
Cuando el tiempo lo permite, una CV con búsqueda acotada sobre `C`, tipo de solver y, en ocasiones, selección de variables (p. ej., descartar features redundantes que no aportan). Este ajuste fino **mejora la estabilidad** sin romper la simplicidad del flujo.

---

3. Regla de decisión y umbralado

Probabilidades y scores.
Si el estimador expone `predict_proba`, utilizo directamente esas probabilidades. Cuando no (o en algunos envoltorios OVR), empleo `decision_function` y **aplico una sigmoide** para obtener un score [0,1]; no es perfecta, pero en la práctica da una señal útil.

Umbrales por etiqueta.
La activación final de cada etiqueta no se deja al azar: leo **`thresholds.json`** (si existe) para aplicar un **umbral específico por clase**; en ausencia, uso un **umbral global** (p. ej., 0.5). Esta pequeña decisión tiene grandes consecuencias coreográficas: hay etiquetas que, por naturaleza, deben ser **más cautas** (evitar falsos positivos que generan ruido pedagógico) y otras que conviene **sensibilizar** para no perder oportunidades de mejora.

Ejemplo de `thresholds.json` (ilustrativo):

```
{
  "variedad_baja": 0.45,
  "poca_simetria": 0.55,
  "fluidez_baja": 0.50
}
```

Alineación con la práctica.
Los umbrales no son "verdades" sino *acuerdos*. En una compañía que trabaja construcciones simétricas, bajar el umbral de `mucha_simetria` es razonable; en una pieza muy minimalista, quizá suba el de `amplitud_baja` para no confundir una elección estética con una carencia.

---

4. Salida y explicabilidad

Qué devuelve el modelo.
Para cada ventana:

- **Etiquetas activas** (concurren sin exclusión: es multietiqueta),
- **Probabilidades/scores** por etiqueta,
- **Sugerencias interpretables** mapeadas 1-a-1 desde la etiqueta (textos cortos, accionables).

Trazabilidad del porqué.
La logística permite inspeccionar **coeficientes** por etiqueta:
si `variedad_direcciones` pesa negativo en `variedad_baja`, puedo mostrarlo. Esta capa de explicabilidad **construye confianza** y guía al docente: no solo *qué* mejorar, sino *qué señal lo indica*.

---

5. Export y operación (MLOps mínimo viable)

Para reproducibilidad y despliegue en la app:

- **Artefactos del preprocesado y modelo**
  - `imputer.joblib`, `scaler.joblib`, `model_ovr_logreg.joblib`
  - `labels.json` (orden de las etiquetas)
  - `thresholds.json` (umbrales por etiqueta, opcional)
- **Evaluación**
  - `artifacts/eval_overall.json` (métricas globales: subset accuracy, hamming, F1 micro/macro/weighted, Jaccard…)
  - `artifacts/eval_per_label.csv` (por etiqueta: precision, recall, F1, support, y si hay, ROC-AUC y AP)
  - **Gráficas**: barras de F1 por etiqueta, peores-N en precisión/recall, ROC/PR micro y, cuando toca, histogramas de AUC/AP.
- **Versionado y control**
  - Registrar versiones de librerías (p. ej., `versions.json`) evita incompatibilidades.
  - Congelar rutas (`BASE/artifacts`, `data/processed/...`) y nombres estabiliza el flujo entre Colab y la app.

---

6. Por qué este enfoque es pertinente en danza

- **Pluralidad simultánea**: la coreografía se juega en capas; el **OVR** respeta esa simultaneidad sin forzar una etiqueta única.
- **Tiempo útil**: trabajar por **ventanas** encaja con la estructura de micro-frases; la app entrega **feedback en contexto**, no microrruido por frame.
- **Lenguaje compartido**: amplitud, niveles, variedad, simetría, fluidez… son categorías **que el estudio ya usa**. El modelo habla ese idioma y lo traduce en **sugerencias accionables**.
- **Transparencia**: el binomio logística + umbral por etiqueta ofrece **control** y **explicabilidad**; en escena, imponer una estética es un riesgo ético. Aquí, el sistema **acompaña**.

---

7. Limitaciones y salvaguardas

- **Calibración**: cuando uso `decision_function + sigmoide`, reviso la **calibración** (posible con `CalibratedClassifierCV`) para que un 0.7 *signifique* algo comparable entre etiquetas.
- **Desbalance**: monitorizo métricas por etiqueta; si F1 micro crece pero F1 macro cae, ajuste de umbrales y pesos de clase.
- **Validación por vídeo**: insistir en splits por vídeo para no inflar resultados.
- **Evolución controlada**: si migro a modelos no lineales (LightGBM/TCN/Transformers ligeros), preservo la **capa de explicación** y recalibro umbrales con cuidado.

El **OVR con Regresión Logística**, montado en un **Pipeline** simple y honesto, me da tres cosas que valoro en un entorno artístico-técnico: **concurrencia de señales**, **explicabilidad inmediata** y **despliegue ágil**. Con umbrales por etiqueta, probabilidades (o scores bien calibrados) y un conjunto de artefactos reproducibles, el sistema no solo acierta *más*; **acierta mejor**, es decir, ofrece **pistas claras y justas** para que la coreografía pueda crecer donde de verdad lo necesita.

---

4) Evaluación

- **Métricas globales**: *subset accuracy* (exact match), *hamming loss*, *F1 micro/macro/weighted*, *precision/recall micro*, *Jaccard* (samples/macro).
- **Por etiqueta**: `precision`, `recall`, `f1`, `support`, y si hay probabilidades: `ROC-AUC` y `Average Precision`.
- **Visualizaciones**:
  - Barras de **F1 por etiqueta** (orden ascendente para priorizar peor desempeño).
  - Barras de **precisión**/**recall** de las peores N etiquetas.
  - **ROC/PR micro** y **distribución de AUC/AP** por etiqueta (si hay `predict_proba`).
  - **Matrices de confusión** de las etiquetas más débiles.

Patrones habituales en multietiqueta de este dominio:

- **F1 micro > F1 macro** → indicio de **desbalance** (clases frecuentes dominan el micro).
- Etiquetas como `variedad_baja` o `fluidez_baja` suelen **solaparse** con métricas cinemáticas globales y son sensibles al **ventaneo** y a la suavización de la pose.
- `mucha_simetria`/`poca_simetria` dependen de la calidad de pares L/R; la **completitud** de keypoints afecta directamente.

Métricas globales (qué miro y por qué)

Cuando hablo de “cómo va” el sistema no me basta con un solo número. Este proyecto es multietiqueta y, además, coreográfico; varias cosas pueden ser ciertas a la vez. Por eso uso un **conjunto** de métricas globales que se complementan:

- **Subset accuracy (exact match).**
  Mide la proporción de ventanas en las que el conjunto de etiquetas predicho coincide **exactamente** con el conjunto real. Es muy estricta: si fallas una sola etiqueta, esa ventana cuenta como error. Útil para saber cuántas veces "lo clavamos" en términos prácticos; injusta cuando el oro tiene muchas etiquetas.
- **Hamming loss.**
  La fracción de etiquetas mal clasificadas respecto al total posible. Es más granular: penaliza menos una salida casi correcta. Si baja, el sistema tiende a equivocarse poco en cada etiqueta.
- **F1 micro / macro / weighted.**
  - **Micro** agrega TP/FP/FN sobre todas las etiquetas y luego calcula F1: prioriza lo frecuente.
  - **Macro** promedia el F1 de cada etiqueta por igual: revela si nos "olvidamos" de las minoritarias.
  - **Weighted** promedia ponderando por soporte (nº de ejemplos por etiqueta): término medio, sensible al desbalance.
    En danza, casi siempre reviso **micro y macro juntos**; si divergen mucho, algo se esconde en la distribución.
- **Precisión/Recall micro.**
  Versión micro de P/R. Buen termómetro para saber si tendemos a **sobre-etiquetar** (precision cae) o a **ser tímidos**(recall cae).
- **Jaccard (samples / macro).**
  El índice Jaccard compara conjuntos (intersección / unión).
  - **Samples** promedia Jaccard por ventana: mide "cuánto coinciden" nuestros conjuntos con el oro.
  - **Macro** promedia por etiqueta: parecido al macro-F1, pero con una intuición de solapamiento de conjuntos.

Ninguna métrica es "la verdad". Las miro como **panel**: exact match para "casos perfectos", Hamming para error medio, y el trío F1 micro/macro/weighted + Jaccard para el equilibrio global.

Métricas por etiqueta (dónde duele)

Una vez vista la foto general, piso tierra: **por etiqueta**.

- **Precision, Recall, F1, Support.**
  Definidos sobre el binario de cada etiqueta: TP/FP/FN/TN.
  - **Precision**: de lo que marco, ¿cuánto es correcto?
  - **Recall**: de lo que debía marcar, ¿cuánto encontré?
  - **F1**: equilibrio entre ambos.
  - **Support**: cuántos ejemplos tiene esa etiqueta en test (si es bajo, desconfío de conclusiones tajantes).
- **Si hay probabilidades: ROC-AUC y Average Precision (AP).**
  - **ROC-AUC** evalúa la capacidad de separar clases a lo largo de umbrales (robusta cuando no hay gran desbalance).
  - **AP (área bajo la curva precisión–recall)** es más informativa con **clases raras**: premia que, cuando disparamos, acertemos.

Para este dominio, **AP por etiqueta** suele contar una historia más honesta que AUC.

Estas métricas por etiqueta me dicen **qué etiquetas están débiles** y si el problema es de **exceso de falsos positivos (precision)** o de **faltas no detectadas (recall)**.
Ajusto **umbrales** etiqueta a etiqueta en consecuencia.

Visualizaciones (para decidir rápido)

- **Barras de F1 por etiqueta (orden ascendente).**
  Abajo del todo están las que requieren **prioridad de mejora**. Es mi lista de tareas.
- **Barras de precisión/recall de las peores N etiquetas.**
  Si la precisión es baja y el recall alto: **umbral demasiado bajo** o señal poco específica (muchos FP).
  Si la precisión es alta y el recall bajo: **umbral demasiado alto** o señal débil (muchos FN).
  Esta pareja me guía el **tuning de umbrales** y, a veces, la **revisión de features**.
- **ROC/PR micro y distribución de AUC/AP por etiqueta (si hay `predict_proba`).**
  - **ROC/PR micro** dan una lectura agregada del sistema.
  - **Histogramas de AUC/AP** por etiqueta revelan si hay una cola de etiquetas con separabilidad pobre: ahí hay que intervenir.
- **Matrices de confusión de las etiquetas más débiles.**
  Aunque sean binarias, ver **TN/FP/FN/TP** por etiqueta (especialmente en las de peor F1) muestra patrones: ¿la confundimos con otra? ¿Aparece siempre en el mismo tipo de ventana?

Estas figuras **ahorran discusiones**: convierten intuiciones en evidencia y explican *por qué* mover un umbral o cambiar una *feature*.

Patrones habituales en multietiqueta de este dominio (y cómo los leo)

- **F1 micro > F1 macro.**
  Señal de **desbalance**: las clases frecuentes dominan la foto.
  Qué hago: revisar **calibración** y **umbrales por etiqueta**,
  considerar `class_weight="balanced"` y, si procede, optimizar el umbral para cada clase maximizando F1/AP en validación. También vigilo el **support** de cada etiqueta: las minoritarias necesitan cariño metodológico.
- **`variedad_baja`** / **`fluidez_baja`** tienden a **solaparse** con métricas cinemáticas globales y son sensibles al **ventaneo** y a la **suavización** de pose.
  Qué hago:
  1. Ajustar tamaño/hop de ventana (micro-ventanas subventaneadas dentro de la principal).

2. Añadir **features temporales** (picos, autocorrelación, FFT ligera de la trayectoria del COM), no solo agregados estáticos.
3. Aplicar **smoothing** de keypoints (p. ej., Savitzky–Golay) para no confundir ruido con "fluidez".

- `mucha_simetria` / `poca_simetria` dependen de la calidad de pares **L/R**; la **completitud de keypoints** afecta directamente.
  Si se pierde una muñeca o un tobillo, la métrica se sesga.
  Qué hago:
  1. Pesar distancias por **confianza** del detector.
  2. Interpolar solo cuando hay **contexto suficiente**; si no, marcar esa ventana como de **menor fiabilidad** para esa etiqueta.
  3. Considerar **normalizaciones estructurales** (longitudes relativas al segmento) para reducir sesgo por escala.

Buenas prácticas que me han funcionado

1. **Panel, no número único.** Subset accuracy para la ambición, Hamming para el día a día, F1/Jaccard para el equilibrio.
2. **Per-Label Tuning.** Un `thresholds.json` vivo, afinado con validación, suele valer más que perseguir décimas en F1 global.
3. **PR por encima de ROC** cuando hay **clases raras**: la AP y la curva PR me han dado diagnósticos más honestos.
4. **Visual primero, hipótesis después.** Las barras ordenadas y las matrices de confusión evitan "arreglos a ciegas".
5. **Split por vídeo.** No mezclar ventanas del mismo vídeo entre train y test; si no, los números mienten.

Evaluar en multietiqueta, y más aún en coreografía, es **escuchar varias voces a la vez puesto que la coreografia es un conjunto en muchas ocasiones**. Las métricas globales me dicen si el conjunto suena afinado; las métricas por etiqueta me señalan qué instrumento desentona; las visualizaciones me muestran **cómo** y **cuándo** ocurre. A partir de ahí, ajusto umbrales, refino *features* y, si hace falta, reformulo el ventaneo. No busco una cifra perfecta, busco **feedback fiable** que ayude a que el movimiento crezca con sentido.

---

5) Experimentos

- **Baseline**: OVR-Logistic (lineal, interpretable) con *features* globales estáticas por ventana.
- **Umbralado**: global vs. por etiqueta (mejor calibración por clase).
- **Validación**: *train/test split* + *k-fold* observado; métricas y ranking de etiquetas por F1.

- **Ablation rápida** (implícita): importancia práctica de `amplitud_*`, `velocidad_media`, `nivel_rango` y `variedad_direcciones` para distintas etiquetas.

Justificación de los experimentos (enfoque, decisiones y validez)

1. Línea base: OVR–Regresión Logística con *features* estáticas por ventana

El cuaderno establece como **baseline** un clasificador **One-Vs-Rest (OVR)** con **Regresión Logística**, precedido por **imputación** y **escalado** dentro de un `Pipeline`. Esta elección se justifica por tres motivos:

1. **Correspondencia con el problema**: el escenario es **multietiqueta**; OVR descompone el problema en decisiones binarias independientes (una por etiqueta), lo que respeta la co-ocurrencia real de aspectos coreográficos en una misma ventana.
2. **Interpretabilidad**: al trabajar con datos estandarizados, los **coeficientes** de la logística permiten leer la contribución de cada *feature* por etiqueta (magnitud y signo). En un contexto pedagógico, esta trazabilidad es crítica: facilita explicar por qué el sistema sugiere "aumentar amplitud" o "introducir diagonales".
3. **Eficiencia y robustez**: frente a alternativas no lineales, la logística es ligera, rápida de entrenar/inferir y suficientemente competitiva con *features* agregadas (estáticas) por ventana. Esto garantiza **latencia baja** para la app y reduce el riesgo de sobreajuste con tamaños moderados de datos.

La decisión de usar **features globales por ventana** (p. ej., amplitud_x/y/z, velocidad_media, simetría, nivel_rango, variedad_direcciones) asume que la **micro-frase** coreográfica puede resumirse en estadísticas compactas. Este supuesto es pragmático: simplifica el modelado, estabiliza frente a ruido de pose y sienta un punto de comparación claro para extensiones temporales posteriores.

---

2. Regla de decisión: umbral global vs. umbral por etiqueta

Tras estimar **probabilidades/scores**, la activación final de etiquetas se controla mediante **umbrales**:

- **Global** (único valor, p. ej. 0.5): simple y reproducible, útil como punto de partida homogéneo.
- **Por etiqueta** (definidos en `thresholds.json`): **calibración específica** por clase, alineada con la semántica coreográfica y el desbalance. Etiquetas propensas a falsos positivos (p. ej., *variedad_baja* en frases minimalistas)

pueden requerir umbrales más altos; otras, más sensibles a omisiones (p. ej., *poco_rango_niveles*), funcionan mejor con umbrales más bajos.

En términos metodológicos, el **umbralar por etiqueta** reduce el **error operativo** en la app (consejos ruidosos o tardíos) y permite adaptar el sistema a **estéticas distintas** sin re-entrenar el clasificador. El cuaderno refleja esta lógica dejando el **fallback global** cuando no existe `thresholds.json`.

---

3. Validación: *train/test split* y *k-fold*

Para estimar la generalización:

- Se aísla un **conjunto de prueba** vía `train_test_split` para evitar sesgos de evaluación.
- Se observa validación con **k-fold**, que estabiliza la estimación de métricas frente a la variabilidad muestral.

En esta tarea, es recomendable (y se toma como criterio de calidad) **estratificar por vídeo** o, al menos, **agrupar por identidad de vídeo** en validación para evitar "fugas" (ventanas del mismo vídeo repartidas entre entrenamiento y validación). Esta salvaguarda es especialmente importante cuando las *features* agregadas capturan rasgos idiosincrásicos del material (tempo, iluminación, escenografía) que podrían inflar artificialmente las métricas.

Métricas reportadas y lectura conjunta.

- Globales: *subset accuracy*, *hamming loss*, **F1 micro/macro/weighted**, *precision/recall micro*, **Jaccard**(samples/macro).
- Por etiqueta: precisión, recall, F1, soporte y, si hay probabilidades, **ROC-AUC** y **Average Precision (AP)**.

La combinación **F1 micro > F1 macro** se interpreta, con criterio estándar, como **indicio de desbalance** (las clases frecuentes dominan la foto). En consecuencia, el *tuning* de umbrales se realiza **por etiqueta**, y se revisan pesos/clasificación para mitigar este efecto.

---

4. "Ablation" implícita: utilidad práctica de familias de *features*

El cuaderno explora —de forma **implícita**— la **importancia práctica** de familias de *features* al observar:

- **Coeficientes por etiqueta** (tras estandarización): su magnitud orienta sobre qué señales empujan cada decisión.

- **Estabilidad de métricas** cuando se **remueven** (o se atenúan) subconjuntos: **amplitud_*** con *amplitud_baja*; **velocidad_media** con *fluidez_baja*; **nivel_rango** con *poco/exceso_rango_niveles*; **variedad_direcciones** con *variedad_baja*.

Esta ablation ligera, apoyada en análisis de coeficientes y en variaciones de F1 por etiqueta, aporta **evidencia causal**(aunque no definitiva) de la **pertinencia** de cada grupo de variables. En dominios con señales ruidosas (pose 2D con oclusiones), esta aproximación es proporcional: no introduce complejidad innecesaria y guía prioridades de ingeniería (p. ej., mejorar *smoothing* antes de añadir decenas de *features*).

---

5. Export y reproducibilidad

El flujo guarda artefactos y resultados en formato estandarizado:

- **Modelo y preprocesado**: `imputer.joblib`, `scaler.joblib`, `model_ovr_logreg.joblib`.
- **Metadatos operativos**: `labels.json` (orden de clases) y `thresholds.json` (umbrales por etiqueta).
- **Evaluación**: `artifacts/eval_overall.json`, `artifacts/eval_per_label.csv` y **gráficas** asociadas.

Este empaquetado permite:
(i) **reproducibilidad** (misma transformación en entrenamiento e inferencia),
(ii) **auditoría** (revisión posterior de decisiones y umbrales), y
(iii) **despliegue estable** en la app sin acoplar el código de entrenamiento.

---

6. Amenazas a la validez y salvaguardas

- **Desbalance y solapamiento semántico**: se aborda con métricas por etiqueta y *tuning* de umbrales; se recomienda `class_weight` y monitorizar **AP** por clase.
- **Calidad de la pose** (pares L/R y *fluidez*): se mitiga con interpolación prudente y, en trabajos futuros, **pesado por confianza** y *smoothing* (p. ej., Savitzky–Golay).
- **Sensibilidad al ventaneo**: la elección 5 s/2.5 s responde a micro-frases; se aconseja validar tamaños alternativos en repertorios con tempi extremos.
- **Fugas por vídeo**: usar splits agrupados por vídeo en la CV.

---

7. Conclusión operativa

El diseño experimental del cuaderno persigue un equilibrio entre **rigurosidad estadística**, **viabilidad en app** y **legibilidad pedagógica**:

- Una **línea base interpretable** (OVR-LogReg) que rinde bien con *features* agregadas.
- Una **decisión por umbrales** ajustable por etiqueta que traslada el modelo al **uso real** sin re-entrenar.
- Una **validación** con *train/test* + *k-fold* que evita optimismo indebido.
- Una **ablation** pragmática que ordena prioridades de señal (amplitud, velocidad_media, nivel_rango, variedad_direcciones) por etiqueta.

Este conjunto de elecciones produce un sistema **estable, explicable y accionable**, y sienta una base sólida para futuras extensiones temporales o no lineales sin perder control ni transparencia.

---

6) Conclusiones

1) Un pipeline lineal OVR es una base sólida, interpretable y operativa.
El enfoque **One-Vs-Rest con Regresión Logística**, precedido por **imputación** y **escalado**, ofrece un equilibrio raro de lograr: **buen rendimiento con pocas suposiciones, explicabilidad inmediata** (coeficientes legibles por etiqueta/feature) y **latencia muy baja** en inferencia. Para una aplicación que debe sugerir en tiempo casi real, esta combinación resulta **apta para producción**. La linealidad, lejos de ser un límite a estas escalas, funciona como **regularizador natural**, reduce sobreajuste y facilita la transferencia entre sesiones y repertorios.
Implicación práctica: mantener este pipeline como **baseline de referencia** y como **fallback robusto** cuando modelos más complejos no aporten mejoras claras o comprometan la interpretabilidad.

2) Rendimiento heterogéneo por etiqueta: lo temporal importa.
Las etiquetas que describen **dinámica y direccionalidad** (p. ej., `variedad_baja`, `fluidez_baja`) dependen de patrones **intravenatana** (cambios, picos, ritmos) que los agregados estáticos no capturan del todo. En cambio, las etiquetas **geométricas** (`amplitud_*`, `nivel_rango`) se benefician decisivamente de estadísticas sencillas.
Implicación práctica: introducir **features temporales** (picos/jerk, autocorrelación, FFT ligera del COM) o **modelos secuenciales ligeros** (TCN/GRU con ventanas subventaneadas) para las etiquetas dinámicas, manteniendo el flujo lineal para las geométricas. Esta **estrategia híbrida** maximiza rendimiento sin sacrificar velocidad ni claridad.

3) Umbralado por etiqueta: más utilidad, exige calibración.
El uso de **umbrales específicos por clase** reduce **falsos positivos** en etiquetas raras y ajusta la sensibilidad a las necesidades estéticas de cada compañía o pieza. No obstante, requiere una **calibración sistemática**: sin ella, el sistema puede sesgarse hacia la timidez (recall bajo) o la verbosidad (precision baja).

Implicación práctica: mantener un **`thresholds.json` versionado**, recalibrado por validación (idealmente optimizando F1/AP por etiqueta) y revisado con cada cambio de datos o features. Añadir **telemetría** en la app (qué sugerencias se consideran útiles) para **realimentar** la calibración.

4) La calidad de la pose es un cuello de botella directo en simetría y variedad.
NaNs, pérdidas de articulación y ruido afectan de manera desproporcionada las métricas que comparan **pares L/R** o calculan **cambios direccionales**. La **interpolación prudente** y el **smoothing** (p. ej., Savitzky–Golay) antes del cómputo de features no son detalles menores: cambian el signo de la decisión.
Implicación práctica: robustecer el preprocesado con (i) **pesado por confianza** del detector, (ii) reglas de **exclusión parcial** cuando una articulación no alcanza umbral mínimo de presencia y (iii) **validaciones automáticas** (porcentaje de keypoints válidos por ventana) que etiqueten ventanas de baja fiabilidad antes de alimentar el clasificador.

---

1) Mantener OVR-LogReg como *baseline* y criterio de aceptación

Qué hacer

- Conserva el pipeline `Imputer → Scaler → OneVsRest(LogisticRegression)` como **modelo de referencia** y **fallback** de producción.
- Fija **semillas** y **versiona artefactos** (`imputer.joblib`, `scaler.joblib`, `model_ovr_logreg.joblib`, `labels.json`, `thresholds.json`, `versions.json`) en `artifacts/` con *hash* y fecha.
- Define una **puerta de aceptación**: ningún modelo candidato pasa a producción si **no supera** (o empata con margen) al baseline en métricas clave.

Criterios de aceptación (ejemplo)

- `F1_macro ≥ F1_macro_baseline − 0.01` **y** `F1_micro ≥ baseline`.
- `AP_median(labels) ≥ baseline` (si hay probabilidades).
- Latencia p95 de inferencia por ventana ≤ **30 ms** en CPU (o presupuesto acordado para la app).

Riesgos / mitigación

- *Drift de datos*: monitorizar distribución de *features*; reentrenar si KS-test/PSI > umbral.
- *Derivas silenciosas por librerías*: bloquea versiones en `requirements.txt` y registra en `versions.json`.

---

2) Hibridar: lineal para geométricas; temporales/secuenciales para dinámicas

Qué hacer

- **Separar etiquetas por "naturaleza"**:
  - **Geométricas**: `amplitud_*`, `nivel_*` → mantener **cabeza lineal** (OVR-LogReg).
  - **Dinámicas**: `variedad_baja`, `fluidez_baja` → añadir **señal temporal**:
    - Fase 1 (rápida): *features* temporales dentro de la ventana (picos, jerk, ACF, FFT ligera del COM).
    - Fase 2 (opcional): **TCN/GRU ligera** sobre subventanas (ej. 1 s) embebidas en la ventana madre.

Implementación mínima

- Genera por ventana, además de agregados estáticos:
  - `peak_density_COM`, `jerk_mean`, `acf_lag1`, `fft_dom_freq`, `fft_bandwidth`.
- Entrena **dos "cabezas"**:
  - `HeadGeom = OVR-LogReg(features_estáticas)`
  - `HeadDyn = (OVR-LogReg(features_estáticas + temporales)` **o** `TCN(features_framewise))`
- Combina salidas por etiqueta (router simple: si etiqueta ∈ dinámicas → `HeadDyn`, si no → `HeadGeom`).

Criterios de aceptación

- Mejora en `F1` y/o `AP` **por etiqueta dinámica** ≥ **+3–5 pts** sin degradar geométricas.
- Latencia total ≤ presupuesto (p. ej., **<60 ms**/ventana en CPU; si TCN, valida con *batching*).

Riesgos / mitigación

- *Sobreajuste temporal*: CV por vídeo y regularización; *early stopping* si deep.
- *Complejidad operativa*: comenzar por Fase 1 (features temporales) antes de TCN/GRU.

---

3) Calibrar umbrales por etiqueta con validación (AP/F1) y telemetría de uso

Qué hacer

- Optimiza **umbral por etiqueta** con **grid** sobre validación maximizando `F1` o `AP`.
- Guarda en `thresholds.json` y **versiona** (incluye fecha y set de validación).

- **Telemetría** en app: por etiqueta, registra cuando la sugerencia se **visualiza**, el usuario la **marca útil** o **la descarta**.

Pseudocódigo (calibración offline)

```
def tune_thresholds(y_true, y_score, labels, metric="f1"):
    th_map = {}
    for i, lbl in enumerate(labels):
        best, best_th = -1, 0.5
        for th in [x/100 for x in range(10, 91)]:  # 0.10→0.90
            yhat = (y_score[:, i] >= th).astype(int)
            if metric == "ap":
                from sklearn.metrics import average_precision_score
                m = average_precision_score(y_true[:, i], y_score[:,
i])
            else:
                from sklearn.metrics import f1_score
                m = f1_score(y_true[:, i], yhat, zero_division=0)
            if m > best:
                best, best_th = m, th
        th_map[lbl] = round(best_th, 2)
    return th_map
```

Ajuste con telemetría (online, suave)

- Define un **score de utilidad** por etiqueta: `u = (clicks_útiles − descartes) / vistas`.
- Cada **N** semanas, revisa: si `u` muy bajo y **FP** altos → sube umbral; si `u` alto y **FN** altos → bájalo ligeramente.
- Cambios **pequeños** (±0.02) y versionados.

Criterios de aceptación

- Reducción de **falsos positivos** en etiquetas raras (↑ precision, u estable/↑).
- No caída relevante de **recall** (>−2 pts) salvo decisión explícita.

---

4) Endurecer la etapa de pose: smoothing, imputación informada y QC por ventana

Qué hacer

- **Smoothing** de keypoints (antes de *features*):
  - **Savitzky–Golay** 1D por articulación/eje (ventana 7–11, orden 2–3) **o** EMA/Kalman simple.
- **Imputación informada por confianza**:
  - Si `conf < τ` en un frame, ignora ese punto en agregados; **no interpoles** secuencias largas sin apoyo.
- **Controles de calidad (QC) por ventana**:
  - `% frames con ≥K articulaciones válidas` ≥ **70%**.
  - `% frames con pares L/R completos` ≥ **60%**.

- Si no se cumple, marca ventana como **low_confidence** y:
  - (a) omite el cálculo de etiquetas sensibles (`mucha/poca_simetria`, `variedad_baja`), **o**
  - (b) emite sugerencias con **disclaimer** y no las persiste.

Criterios de aceptación

- Disminución de **ruido** en `mucha/poca_simetria` y `variedad_baja` (↑ precision, ↓ varianza de F1).
- Tasa de ventanas *low_confidence* < **10–15%** (según repertorio).

Riesgos / mitigación

- *Descartar demasiado*: ajusta umbrales de QC y revisa "por qué" cayó la confianza (iluminación, encuadre).

---

5) Monitorizar por etiqueta (no solo global)

Qué hacer

- Genera semanalmente `eval_per_label.csv` y compón un **panel** con:
  - **F1**/AP por etiqueta (línea temporal 4–8 semanas).
  - **Barras ordenadas** de peores-N (F1/precision/recall).
  - **Alertas**: dispara aviso si `F1` o `AP` de una etiqueta cae **> 5 pts** respecto a su media móvil de 3 semanas.
- Integra **confusión** por etiqueta débil (TN/FP/FN/TP) para diagnóstico.

Aceptación

- Alertas **accionables** (<5% falsos positivos).
- Tiempo de refresco del panel **< 10 min** post-entrenamiento.

Riesgos / mitigación

- *Señal inestable por poco soporte*: muestra **barras de error** o *shaded band* con IC bootstrap; etiqueta "bajo soporte".

---

6) Gobernanza: documentación y reproducibilidad

Qué hacer

- **Model card**: versión, datos, métricas por etiqueta, límites conocidos (p. ej., escenas oscuras).
- **Data card** del *features* CSV: esquema, normalizaciones, reglas de QC.
- **Checklist de entrega**:

1. Artefactos + thresholds versionados,
2. `eval_overall.json` + `eval_per_label.csv`,
3. Gráficas clave,
4. Panel actualizado.

---

7) Roadmap sugerido (3 sprints de 2 semanas)

- **Sprint 1**:
  Baseline congelado + panel per-label + calibración por etiqueta (offline) + QC de pose.
- **Sprint 2**:
  Features temporales (Fase 1) + re-calibración + telemetría en app + alertas automáticas.
- **Sprint 3**:
  Cabeza **TCN/GRU** opcional para dinámicas + pruebas A/B con latencia y utilidad percibida.

Resumen operativo

- Mantén **OVR-LogReg** como columna vertebral: rápido, estable, explicable.
- Añade **temporalidad donde duele** (dinámicas), sin penalizar latencia.
- **Umbraliza por etiqueta** con datos (AP/F1) y **ajusta con uso real** (telemetría).
- **Endurece la pose** antes de decidir: eslabón crítico para simetría y variedad.
- **Mira etiqueta por etiqueta**: así se corrige con precisión y se evita el autoengaño de los promedios.

Con estas prácticas, el sistema conserva su **vocación pedagógica** (sugerencias claras y justificadas) mientras gana **robustez** para contextos diversos, manteniendo lo esencial: **decir qué mejorar y por qué**, justo **cuando** hace falta.

---

7) Límites actuales

- *Features* mayormente **agregadas** (estáticas por ventana): pierden **dinámica intraventana** (transiciones, acentos rítmicos).
- **Probabilidades**: no siempre disponibles; cuando se derivan vía `decision_function + sigmoide`, la calibración puede ser pobre.
- **Desbalance** de etiquetas y **solapamiento semántico** entre clases.
- Dependencia de **calidad de keypoints** (iluminación, oclusiones, escala).

---

8) Mejoras propuestas

8.1 Features / Señales

- **Temporales**: *delta/jerk* de COM, *rolling stats* (medianas, IQR), **picos** (peak density), **duraciones** de estados, **ritmo**(autocorrelación), **FFT/mFCC** de trayectorias 2D del COM y de segmentos (frecuencia dominante, ancho de banda).
- **Estructurales**: proporciones normalizadas (longitudes por altura del sujeto), **ángulos** entre segmentos (0–180°) con robustez a faltantes, **asimetría** robusta (pares L/R pesados por confianza).
- **Contexto**: backends de pose (flag), *scene scale* estimada (para normalizar amplitudes).

8.2 Modelado

- **Modelos secuenciales** (LSTM/GRU/Temporal-CNN o Transformers ligeros) sobre secuencias de *features* frame-wise o ventanas deslizantes subven-taneadas.
- **Modelos no lineales** con calibración: Gradient Boosting (XGBoost/LightGBM) + **CalibratedClassifierCV** para probabilidades útiles.
- **Umbralado** con **optimización por etiqueta** (maximize F1/AP por clase en validación); guardar `thresholds.json` por split.
- **Cost-sensitive learning**: `class_weight`, *focal loss* (si deep), *oversampling* por etiqueta minoritaria.

8.3 Validación / Métricas

- **CV estratificada por vídeo** (no mezclar ventanas del mismo vídeo en train/test).
- Reporte de **AP por etiqueta** + **PR-curves** (más informativas en desbalance).
- **Top-k accuracy**/coverage en multietiqueta (relevante para UX: mostrar 2–3 sugerencias).
- **Ablation**: quitar familias de *features* (amplitud vs. simetría vs. niveles) para ver contribuciones.

8.4 Robustez de Pose

- **Smoothing** de keypoints (Savitzky–Golay / Kalman) antes de derivar *features*.
- **Interpolación adaptativa** por confianza del detector; marcar frames con confianza baja para excluirlos del cómputo de simetría/variedad.
- **Auto-fallback** de backends (MediaPipe ↔ YOLO) según escena.

8.5 Producto / App

- **Streaming** con *micro-ventanas* y **debounce** de sugerencias (persistir solo si la señal es estable N segundos).
- **Explicabilidad**: mostrar la **feature dominante** que activó cada etiqueta (coeficiente absoluto top-k en Logistic).
- **Telemetría**: registrar qué sugerencias el usuario marca como útiles para **re-entrenar** con *feedback*.

9) Próximos pasos operativos

1. **Refactor** de *feature factory* para generar features frame-wise + agregados multi-escala (0.5 s, 1 s, 2 s, 5 s).
2. **Benchmark** de 2–3 modelos: OVR-LogReg (baseline), LightGBM (no lineal), TCN (temporal) con **calibración** y **threshold tuning** por etiqueta.
3. **Validación por vídeo** y **reporte por etiqueta** (AP/F1) + *error analysis* guiado por `eval_per_label.csv`.
4. **Hardening** del pipeline de pose (smoothing + filtros por confianza) y de rutas en Colab/Drive.
5. **Entrega**: congelar artefactos (`.joblib`, `labels.json`, `thresholds.json`, `versions.json`), actualizar README y **scripts de evaluación/inferencia**.

El sistema actual —OVR con *features* globales— es **rápido, interpretable y útil**, pero para mejorar las etiquetas más dependientes de **dinámica** (fluidez/variedad) conviene **añadir señal temporal** y **calibrar probabilidades/umbrales** por etiqueta, manteniendo splits por vídeo y robusteciendo la **calidad de pose**.