

Informe Integral sobre la Implementación de Sistemas de Diagnóstico Dermatológico Automatizado mediante Arquitecturas YOLOv8

1. Introducción y Contextualización del Aprendizaje Profundo en Dermatoscopia

La intersección entre la inteligencia artificial y la medicina diagnóstica ha experimentado una transformación radical en la última década, impulsada fundamentalmente por el advenimiento del Aprendizaje Profundo (*Deep Learning*). En el ámbito de la dermatología, la capacidad de diferenciar entre lesiones cutáneas benignas y malignas es una tarea crítica que tradicionalmente ha dependido de la experiencia visual y subjetiva del clínico. El proyecto que nos ocupa, centrado en la clasificación de lesiones del conjunto de datos ISIC 2019 utilizando la arquitectura YOLOv8, no es meramente un ejercicio académico, sino una simulación de un flujo de trabajo real en ingeniería biomédica. Este informe desglosa de manera exhaustiva los fundamentos teóricos, la ingeniería de datos, la arquitectura de modelos y los protocolos de evaluación necesarios para completar con éxito la práctica asignada, asumiendo un enfoque desde los primeros principios para garantizar una comprensión profunda de cada componente.

1.1. La Evolución de la Visión por Computador: De la Detección a la Clasificación

Para comprender la elección de YOLO (*You Only Look Once*) para esta tarea, es imperativo contextualizar su posición en la historia de la visión por computador. Tradicionalmente, YOLO ha sido el estándar de oro para la **detección de objetos** en tiempo real. La detección implica dos subtareas simultáneas: la localización (determinar las coordenadas o *bounding box* de un objeto en una imagen) y la clasificación (determinar qué es ese objeto).¹ Sin embargo, la tarea asignada en esta práctica es estrictamente de **clasificación de imágenes**.

La clasificación de imágenes es el proceso fundamental mediante el cual una red neuronal asigna una etiqueta única a una imagen completa basándose en su contenido visual predominante, sin necesidad de localizar espacialmente la lesión dentro del encuadre.³ Históricamente, modelos como ResNet o EfficientNet dominaban este espacio. No obstante, la versión 8 de YOLO (YOLOv8), desarrollada por Ultralytics, introdujo una arquitectura unificada

que soporta nativamente la clasificación (identificada con el sufijo -cls), aprovechando la potencia de su "backbone" o columna vertebral de extracción de características para tareas que no requieren localización.⁴ Esto representa un cambio de paradigma: utilizar un motor diseñado para la velocidad y la complejidad espacial (detección) y reorientarlo para la precisión categórica (clasificación).

1.2. El Desafío Clínico: El Conjunto de Datos ISIC 2019

El conjunto de datos proporcionado proviene del *International Skin Imaging Collaboration* (ISIC), la referencia mundial en imagen dermatológica. Entender la naturaleza de los datos es el primer paso para cualquier ingeniero de ML. El dataset ISIC 2019 contiene 25,331 imágenes dermoscópicas que abarcan nueve categorías diagnósticas distintas.⁵

La complejidad de este dataset radica en su **desbalanceo de clases**. En la práctica clínica real, y reflejado en estos datos, las lesiones benignas como los nevus melanocíticos (lunares comunes) son abrumadoramente más frecuentes que patologías graves como el melanoma o el carcinoma de células basales.⁷ Si se entrenara un modelo con la distribución natural de los datos, la red neuronal desarrollaría un sesgo hacia la clase mayoritaria, aprendiendo a clasificar todo como "nevus" para maximizar su precisión estadística, fallando catastróficamente en la detección de cáncer, que es el objetivo médico prioritario. Por esta razón, la práctica impone una restricción metodológica estricta: la creación de subconjuntos balanceados (100 imágenes para entrenamiento y 10 para prueba por clase)¹, forzando al modelo a aprender características distintivas de cada patología por igual, independientemente de su prevalencia en la población.

2. Ingeniería de Datos y Preparación del Entorno (setup.ipynb)

El archivo setup.ipynb constituye la base fundamental del proyecto. En el aprendizaje automático, la calidad y estructura de los datos dictan el techo de rendimiento del modelo; un modelo sofisticado alimentado con datos mal estructurados o etiquetas erróneas es inútil. Esta fase implica la descarga, descompresión, procesamiento de metadatos y reestructuración física de los archivos.

2.1. Adquisición y Exploración de Datos

El primer paso lógico es la obtención de los datos crudos. El conjunto de datos ISIC 2019 se divide en dos componentes principales: las imágenes (en formato JPEG dentro de un archivo ZIP) y las etiquetas o "Ground Truth" (en un archivo CSV). Las imágenes son la entrada (*inputs*) y el CSV contiene las salidas esperadas (*targets*).¹

Al descomprimir el archivo de imágenes, nos encontramos con una carpeta plana donde residen miles de archivos con nombres como ISIC_0000000.jpg, ISIC_0000001.jpg, etc. Sin embargo, el nombre del archivo no indica la enfermedad. Esa información reside exclusivamente en el archivo ISIC_2019_Training_GroundTruth.csv.⁵

Análisis del Formato *One-Hot Encoding*

Una de las barreras iniciales más comunes es la interpretación del archivo CSV. Al abrir este archivo, no encontramos una columna simple que diga "Clase: Melanoma". En su lugar, encontramos una matriz dispersa siguiendo el formato *One-Hot Encoding*.⁷

image	MEL	NV	BCC	AK	BKL	DF	VASC	SCC	UNK
ISIC_001	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ISIC_002	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

En este formato:

- Cada fila representa una imagen.
- Cada columna (excepto la primera) representa una posible enfermedad.
- Un valor de 1.0 indica la presencia de esa enfermedad, mientras que 0.0 indica su ausencia.
- **MEL:** Melanoma.
- **NV:** Nevus Melanocítico.
- **BCC:** Carcinoma de Células Basales.
- **AK:** Queratosis Actínica.
- **BKL:** Queratosis Benigna.
- **DF:** Dermatofibroma.
- **VASC:** Lesión Vascular.
- **SCC:** Carcinoma de Células Escamosas.
- **UNK:** Desconocido.

Para poder organizar las imágenes en carpetas (que es lo que YOLO necesita), es necesario transformar este formato matricial a una lista de etiquetas categóricas. Utilizando la librería pandas en Python, esta operación se realiza identificando qué columna tiene el valor máximo (el 1.0) para cada fila. La función `idxmax(axis=1)` es la herramienta estándar para esta transformación, colapsando las 9 columnas de enfermedades en una sola columna llamada

"clase" o "label".¹⁰

2.2. Estrategia de Muestreo Estratificado

El enunciado exige extraer específicamente **100 imágenes para entrenamiento** y **10 imágenes para test** de cada una de las clases.¹ Esto no es trivial debido a la variabilidad en la cantidad de imágenes originales.

- **¿Por qué no seleccionar las primeras 100?** Si seleccionamos las primeras 100 imágenes que aparecen en el CSV ordenado alfabéticamente, corremos el riesgo de introducir sesgos temporales o de procedimiento (por ejemplo, si las imágenes de un hospital específico se cargaron primero).
- **La necesidad de aleatoriedad:** Se debe aplicar un muestreo aleatorio (*random sampling*) para garantizar que las imágenes seleccionadas sean representativas de la variabilidad intra-clase.
- **Reproducibilidad (Seeds):** En ciencia, los experimentos deben ser reproducibles. Al usar funciones aleatorias, es imperativo establecer una "semilla" (*random_state* o *seed*), generalmente un número entero como 42. Esto asegura que cada vez que se ejecute el código `setup.ipynb`, se seleccionen exactamente las mismas imágenes, permitiendo que los compañeros o evaluadores repliquen los resultados idénticos.

El proceso algorítmico en `setup.ipynb` debe seguir esta lógica:

1. Cargar el CSV en un DataFrame de Pandas.
2. Convertir las columnas *One-Hot* a una columna categórica única.
3. Agrupar el DataFrame por la columna de clase.
4. Para cada grupo (clase), extraer una muestra aleatoria de 110 imágenes (o el máximo disponible si alguna clase tuviera menos, aunque en ISIC 2019 las clases principales tienen suficientes).
5. Dividir esas 110 imágenes: las primeras 100 para el conjunto de entrenamiento (train) y las 10 restantes para el conjunto de prueba (test).

2.3. Estructura de Directorios para YOLOv8 Classify

A diferencia de los modelos de detección que requieren archivos de texto con coordenadas, los modelos de clasificación de YOLOv8 infieren la etiqueta de la imagen basándose en la estructura de directorios en la que se encuentra el archivo. Este es un punto crítico: si la estructura de carpetas es incorrecta, el entrenamiento fallará o producirá resultados sin sentido.¹²

El script `setup.ipynb` debe crear físicamente la siguiente jerarquía en el disco:

Nivel 1	Nivel 2	Nivel 3 (Clases)	Contenido
dataset/	train/	MEL/	100 imágenes .jpg de melanomas
		NV/	100 imágenes .jpg de nevus
		BCC/	100 imágenes .jpg de carcinomas basales
		...	(resto de las clases)
	test/	MEL/	10 imágenes .jpg de melanomas
		NV/	10 imágenes .jpg de nevus
		...	(resto de las clases)

Es crucial que el código verifique la existencia de estas carpetas antes de copiar los archivos y, preferiblemente, limpie cualquier ejecución anterior para evitar mezclar datos. La librería `shutil` de Python (específicamente `shutil.copy`) es la herramienta estándar para mover los archivos desde la carpeta de descarga descomprimida a esta nueva estructura organizada.¹⁴

Además, es posible que algunas imágenes en el CSV no existan en el ZIP o viceversa (aunque raro en datasets curados como ISIC, es una buena práctica defensiva verificar la existencia del archivo antes de copiarlo). También se debe asegurar que los nombres de archivo coincidan exactamente, añadiendo la extensión `.jpg` si el CSV solo contiene el ID base (ej: `ISIC_0000000` -> `ISIC_0000000.jpg`).

3. Entrenamiento de Modelos y Arquitectura YOLO (train.ipynb)

Una vez que los datos están "limpios" y organizados, entramos en la fase de aprendizaje. El archivo train.ipynb es donde se define, instancia y entrena la red neuronal. El objetivo es crear tres modelos distintos basados en las variantes de tamaño de YOLOv8: Nano (n), Small (s) y Medium (m).¹

3.1. Arquitectura Interna de YOLOv8-Cls

YOLOv8 es una Red Neuronal Convolucional (CNN). Para "bajar a tierra" este concepto: imagine que la red es una serie de filtros o tamices.

1. **Capas Iniciales (Low-level features):** Las primeras capas de la red detectan patrones muy simples: líneas verticales, horizontales, curvas, puntos de color.
2. **Capas Intermedias:** Combinan las líneas y curvas para detectar formas más complejas: círculos, texturas rugosas, patrones irregulares típicos de lesiones.
3. **Capas Profundas (High-level features):** Combinan las formas para identificar "conceptos": una estructura asimétrica con bordes irregulares (típico de melanoma) vs. una estructura redonda y uniforme (nevus).
4. **Clasificador (Head):** Al final, la red toma toda esta información abstracta y emite 9 números: la probabilidad de que la imagen pertenezca a cada una de las 9 clases.

La diferencia entre las versiones n, s y m radica en la **profundidad** (cuántas capas tienen) y el **ancho** (cuántos filtros tiene cada capa).⁴

- **YOLOv8n (Nano):** ~1.5 millones de parámetros. Es extremadamente rápida y ligera. Aprende rápido pero tiene menos "capacidad cerebral" para entender matices muy sutiles.
- **YOLOv8s (Small):** ~5 millones de parámetros. Un punto medio.
- **YOLOv8m (Medium):** ~15 millones de parámetros. Tiene mucha más capacidad para aprender patrones complejos, pero requiere más memoria GPU y, crucialmente, necesita más datos para no "memorizar" (sobreajustar) el conjunto de entrenamiento.

3.2. Transfer Learning: La Clave del Éxito

El enunciado instruye usar modelos **pre-entrenados** (yolov8n-cls.pt, etc.).¹ Esto es fundamental. Entrenar una red desde cero (*from scratch*) con inicialización aleatoria requeriría decenas de miles de imágenes y mucho tiempo para que la red aprenda siquiera a detectar un borde.

Al usar un modelo pre-entrenado (generalmente en ImageNet, un dataset con millones de

imágenes de la vida cotidiana), partimos de una red que *ya sabe ver*. Ya sabe qué es una textura, una forma y un contraste. Lo que hacemos en train.ipynb es **Transfer Learning** (Aprendizaje por Transferencia): tomamos ese conocimiento general y lo refinamos (*fine-tuning*) para la tarea específica de dermatología. Es análogo a enseñar a un médico generalista a ser dermatólogo: ya sabe medicina (pre-entrenado), solo necesita especializarse en la piel (dataset ISIC).

3.3. Configuración de Hiperparámetros de Entrenamiento

El código de entrenamiento en YOLOv8 es de alto nivel gracias a la librería ultralytics, pero requiere configurar parámetros críticos ¹²:

Python

```
from ultralytics import YOLO

# Cargar el modelo pre-entrenado
model = YOLO('yolov8n-cls.pt')

# Iniciar entrenamiento
results = model.train(
    data='path/to/dataset', # Ruta a la carpeta creada en setup
    epochs=50,              # Número de veces que la red ve todas las fotos
    imgsz=224,              # Resolución de entrada (estándar en clasificación)
    batch=16,               # Imágenes procesadas simultáneamente
    project='ISIC_Project', # Carpeta para guardar resultados
    name='yolov8n_run'      # Subcarpeta específica para este modelo
)
```

Análisis de los Argumentos:

- **data:** Debe apuntar a la carpeta raíz que contiene train y test. YOLO buscará automáticamente estas subcarpetas.¹²
- **epochs:** Un ciclo completo de mostrarle al modelo las 900 imágenes (100x9 clases). Si es muy bajo (ej. 5), el modelo no aprende suficiente (*subajuste*). Si es muy alto (ej. 200), el modelo empieza a memorizar el ruido (*sobreajuste*). Un valor entre 20 y 50 suele ser un buen punto de partida para este tamaño de dataset.
- **imgsz:** Las redes neuronales necesitan entradas de tamaño fijo. 224x224 píxeles es el estándar industrial para clasificación (como en ResNet o EfficientNet). YOLO redimensionará automáticamente las imágenes originales de ISIC (que son mucho más grandes) a este tamaño.
- **batch:** Determina cuántas imágenes ve la red antes de actualizar sus "pesos" matemáticos. Un batch más grande da estimaciones de error más estables pero

consume más memoria VRAM de la tarjeta gráfica.

3.4. Monitorización: Pérdida y Precisión

Durante el entrenamiento, el script mostrará y guardará métricas en tiempo real. Es vital entender qué significan para cumplir con el requisito de "analizar su comportamiento".¹

- **Loss (Función de Pérdida):** Representa el error del modelo. Es un número que queremos minimizar. En clasificación, se usa típicamente la *Cross-Entropy Loss*.
 - **Train Loss:** Error sobre las imágenes que el modelo está estudiando. Siempre debería bajar.
 - **Val Loss:** Error sobre las imágenes de validación (si se definen) o test. Es la prueba de fuego.
- **Accuracy (Precisión):** Porcentaje de aciertos. Queremos maximizarlo.

Interpretación de Curvas ¹⁷:

1. **Comportamiento Ideal:** Train Loss baja y Val Loss baja en paralelo.
2. **Sobreajuste (Overfitting):** Train Loss sigue bajando (el modelo memoriza), pero Val Loss empieza a subir (el modelo falla en datos nuevos). Esto es muy probable que ocurra con el modelo **Medium** (m) dado que tenemos pocas imágenes (100 por clase).
3. **Subajuste (Underfitting):** Ambas pérdidas se quedan altas y no bajan. El modelo no es capaz de encontrar patrones.

El estudiante debe implementar un bucle en Python que entrene secuencialmente los tres modelos (yolov8n-cls.pt, yolov8s-cls.pt, yolov8m-cls.pt) y asegurar que los resultados de cada uno se guarden en carpetas separadas para su posterior análisis.

4. Evaluación y Validación de Modelos (eval.ipynb)

El archivo eval.ipynb tiene como propósito realizar una auditoría final de los modelos entrenados. A diferencia de la validación durante el entrenamiento (que sirve para ajustar la red), esta evaluación utiliza el conjunto de test para simular cómo se comportaría el modelo en el mundo real con pacientes nuevos.

4.1. Generación de Predicciones

Para evaluar, cargamos los pesos del modelo ya entrenado (que se guardan automáticamente como best.pt en la carpeta de resultados) y ejecutamos el modo de validación o predicción.

Python

```
# Ejemplo conceptual
```

```
model = YOLO('runs/classify/ISIC_Project/yolov8n_run/weights/best.pt')
```

```
metrics = model.val() # Evalúa en el dataset de test configurado
```

4.2. Métricas Fundamentales en Medicina

En un problema de clasificación de gatitos, equivocarse tiene poco coste. En el diagnóstico de cáncer de piel, los errores son críticos. Por ello, la evaluación no debe limitarse a la "Exactitud" (*Accuracy*), sino profundizar en métricas desglosadas por clase.

1. **Matriz de Confusión:** Es la herramienta de diagnóstico más potente para el propio modelo.¹⁹ Es una tabla cuadrada (9x9 en este caso) donde las filas representan la **Clase Real** y las columnas la **Clase Predicha**.
 - La **diagonal principal** muestra los aciertos (era Melanoma y predijo Melanoma).
 - Los elementos fuera de la diagonal son los errores.
 - *Análisis Crítico:* Es vital observar si el modelo confunde clases peligrosas (MEL, BCC) con benignas (NV, BKL). Confundir un Melanoma con un Nevus (Falso Negativo) es el peor error posible. Confundir un Nevus con Melanoma (Falso Positivo) implica una biopsia innecesaria, pero salva la vida del paciente.
2. **Top-1 Accuracy vs Top-5 Accuracy:**
 - **Top-1:** El modelo acierta si la clase con mayor probabilidad es la correcta.
 - **Top-5:** El modelo acierta si la clase correcta está entre las 5 con mayor probabilidad.
 - Para este problema médico, nos interesa principalmente la **Top-1**. Queremos un diagnóstico preciso, no una lista de "podría ser esto".
3. **Precision, Recall y F1-Score:**
 - **Precision (Valor Predictivo Positivo):** De todos los casos que el modelo llamó "Melanoma", ¿cuántos lo eran realmente?
 - **Recall (Sensibilidad):** De todos los Melanomas que existen en el dataset, ¿cuántos encontró el modelo? En medicina, maximizar el **Recall** suele ser prioritario para no dejar escapar casos patológicos.
 - **F1-Score:** Es una media armónica que balancea ambos. Es útil para comparar modelos globalmente.

El script eval.ipynb debe iterar por los tres modelos entrenados, cargar sus pesos, ejecutar la validación y almacenar estas métricas. Es recomendable guardar las matrices de confusión como imágenes o archivos CSV para incluirlos posteriormente en el informe LaTeX.

5. Análisis Comparativo y Visualización (results.ipynb)

Este cuaderno final actúa como el centro de síntesis. Aquí no se entrena ni se predice, sino que se recopilan los datos generados por los notebooks anteriores para crear las tablas y figuras que irán al informe final.

5.1. Comparación de Arquitecturas

El objetivo es responder a la pregunta: ¿Afecta el tamaño del modelo al rendimiento en este dataset específico?

Se deben generar tablas comparativas como la siguiente (ejemplo hipotético):

Modelo	Tamaño (Params)	Tiempo Entrenamiento (min)	Top-1 Accuracy (Test)	Recall (Melanoma)
YOLOv8n	1.5M	15	72%	68%
YOLOv8s	5.0M	25	75%	71%
YOLOv8m	15.0M	40	74%	65%

Insight de Segundo Orden: Es posible que descubra que el modelo m (Medium), a pesar de ser teóricamente más potente, obtenga resultados similares o peores que el s (Small). Esto se explica por la cantidad limitada de datos (100 imágenes por clase). Los modelos grandes son "hambrientos de datos"; sin suficientes ejemplos, tienden a sobreajustarse rápidamente, memorizando el ruido de entrenamiento en lugar de generalizar. El modelo n o s suele ser más eficiente en regímenes de datos escasos (*Low-Data Regimes*).

5.2. Visualización de Curvas de Aprendizaje

El código debe leer los archivos results.csv generados por YOLO en cada entrenamiento y graficar en una sola figura las curvas de pérdida de validación de los tres modelos.²¹

- Eje X: Épocas.
- Eje Y: Pérdida (Loss).
- Series: Nano, Small, Medium.
- Esto permitirá visualizar qué modelo converge más rápido y cuál muestra signos de inestabilidad.

6. Elaboración del Informe Técnico en LaTeX (Formato LNCS)

La práctica culmina con la comunicación de los resultados. Se requiere el uso de **LaTeX** y específicamente la plantilla **LNCS** (*Lecture Notes in Computer Science*) de Springer.²³ LaTeX es el estándar *de facto* para la publicación científica en campos técnicos debido a su gestión superior de fórmulas matemáticas, referencias cruzadas y consistencia tipográfica.

6.1. Estructura de la Plantilla LNCS

La plantilla LNCS tiene una estructura rígida que debe respetarse. El archivo principal .tex debe contener:

1. **Título y Autores:** Siguiendo el formato ciego o con nombre según indique la cátedra.
2. **Abstract (Resumen):** Un párrafo denso de 150-250 palabras. Debe resumir el problema (clasificación ISIC), el método (YOLOv8 con muestreo balanceado), los experimentos (comparativa n/s/m) y la conclusión principal. No lleva citas.
3. **1. Introduction (Introducción):**
 - Presentar el problema del cáncer de piel y la dificultad del diagnóstico visual.
 - Introducir las Redes Neuronales Convolucionales (CNN) y YOLO.
 - Definir el objetivo del trabajo.
4. **2. Dataset and Methodology (Conjunto de Datos y Metodología):**
 - Describir ISIC 2019: número de clases, desbalanceo original.
 - Explicar el preprocesamiento: detallar el algoritmo de muestreo estratificado (100 train / 10 test) implementado en setup.ipynb.
 - Describir la arquitectura YOLOv8 y la configuración de entrenamiento (épocas, optimizador, función de pérdida).
5. **3. Experiments and Results (Experimentos y Resultados):**
 - Aquí es donde se insertan las tablas y figuras generadas en results.ipynb.
 - Incluir la matriz de confusión del mejor modelo.
 - Comparar cuantitativamente los modelos n, s, y m.
6. **4. Discussion (Discusión):**
 - Interpretar los resultados. ¿Por qué el melanoma es más difícil de clasificar que otras lesiones? (Similitud visual con nevus).
 - Analizar el sobreajuste/subajuste basándose en las curvas de pérdida.
 - Discutir el *trade-off* (compromiso) entre la velocidad del modelo Nano y la precisión del modelo Small.
7. **5. Conclusions (Conclusiones):** Resumen final y líneas de trabajo futuro (ej. usar aumento de datos o *data augmentation* para mejorar resultados con pocas imágenes).
8. **References (Referencias):** Bibliografía formateada correctamente (BibTeX), citando el paper original de YOLO, el dataset ISIC y las librerías principales.

6.2. Recomendaciones de Sintaxis LaTeX

Para un estudiante nuevo en LaTeX, estos son los comandos esenciales para el informe:

- **Imágenes:**

Fragmento de código

```
\begin{figure}[ht]
```

```
\centering
```

```
\includegraphics[width=0.8\textwidth]{loss_curves.png}
```

```
\caption{Comparativa de curvas de pérdida de validación para YOLOv8n, s y m.}
```

```
\label{fig:loss}
```

```
\end{figure}
```

- **Tablas:** Utilizar entornos tabular dentro de table.
- **Citas:** Usar `\cite{autor2023}` en el texto y definir la entrada correspondiente en el archivo .bib.

7. Resumen de Pasos para la Ejecución Exitosa

Para sintetizar "todo lo que debe hacer desde lo más simple", siga esta hoja de ruta estricta:

1. **Fase 1 (Datos - setup.ipynb):**

- Descargue el ZIP y el CSV.
- Programe en Python con Pandas para leer el CSV.
- Use `idxmax()` para obtener las etiquetas de clase.
- Haga un muestreo aleatorio (`sample(n=110)`) por cada clase.
- Mueva los archivos a carpetas `dataset/train/CLASE` y `dataset/test/CLASE`. **Verifique manualmente que las carpetas contienen imágenes.**

2. **Fase 2 (Entrenamiento - train.ipynb):**

- Instale ultralytics (`pip install ultralytics`).
- Entrene el modelo nano (`yolo classify train model=yolov8n-cls.pt data=dataset...`).
- Repita para small y medium.
- Asegúrese de que el proceso termina y genera archivos en `runs/classify/`.

3. **Fase 3 (Evaluación - eval.ipynb):**

- Cargue los modelos entrenados.
- Ejecute la validación sobre el set de test.
- Extraiga la matriz de confusión y guárdela.

4. **Fase 4 (Informe - LaTeX):**

- Abra Overleaf o un editor LaTeX local con la plantilla LNCS.
- Redacte el contenido basándose en los datos obtenidos.
- Compile el PDF.

Siguiendo esta guía estructural y conceptual, se cubre no solo la ejecución técnica del código requerido, sino también la justificación científica detrás de cada decisión, garantizando un

entendimiento profundo ("bajar a tierra") de la materia.