Informe Final: Solución de IA para la Clasificación de Literatura Médica

Proyecto: Clasificación Multi-etiqueta de Artículos Médicos mediante Machine Learning.
Equipo: Los sabrosos del código **Fecha:** 25 de agosto de 2025

1. Análisis Exploratorio y Comprensión del Problema

El desafío consistió en desarrollar un sistema de Inteligencia Artificial para clasificar artículos médicos en uno o varios dominios (**problema de clasificación multi-etiqueta**) utilizando únicamente el **título** y el **resumen (abstract)** como datos de entrada. El objetivo principal es predecir la variable `group`, que contiene las categorías médicas a las que pertenece cada artículo.

El dataset de referencia (`challenge_data-18-ago.csv`) contiene aproximadamente 3,500 registros. Un análisis inicial reveló los siguientes puntos clave:

- **Naturaleza del Texto:** El `title` es conciso, mientras que el `abstract` es denso en terminología técnica, acrónimos y jerga médica especializada. La combinación de ambos campos (`title` + `abstract`) crea una "bolsa de palabras" robusta y rica en contexto, fundamental para el modelo.
- **Estructura de Etiquetas:** Las etiquetas en la columna `group` se presentan como una cadena de texto, con múltiples categorías separadas por un delimitador (`|`). Esto confirmó la necesidad de un enfoque multi-etiqueta, donde un solo artículo puede ser relevante para dominios como `cardiovascular` y `neurological` simultáneamente.
- **Retos Identificados:**
    - **Ambigüedad terminológica:** Palabras clave pueden aparecer en múltiples dominios, lo que requiere un modelo capaz de capturar matices contextuales.
    - **Desbalance de clases (potencial):** Algunas categorías médicas podrían ser significativamente más frecuentes que otras, afectando el rendimiento del modelo en las clases minoritarias.
    - **Alta dimensionalidad:** El vocabulario médico es extenso. Se necesita una técnica de extracción de características que gestione eficientemente la dimensionalidad sin perder información relevante.

Para este proyecto, se decidió trabajar con un subconjunto de cuatro categorías bien definidas: **cardiovascular, neurological, hepatorenal y oncological**, lo que permitió un

prototipado rápido, una mejor interpretabilidad de los resultados y la obtención de métricas de rendimiento sólidas.

2. Preparación, Preprocesamiento y Justificación

Nuestro pipeline de preprocesamiento fue diseñado para ser modular, reproducible y eficiente, abordando los retos identificados en el análisis exploratorio.

A. Preprocesamiento de Texto:

1. **Construcción del Corpus:** Se unificaron las columnas `title` y `abstract` en un único campo de texto. Esta decisión maximiza la cantidad de información contextual disponible para cada documento.
2. **Vectorización con TF-IDF:** Se eligió el método **Term Frequency-Inverse Document Frequency (TF-IDF)** para convertir el texto en vectores numéricos.
    - **Justificación:** TF-IDF es un estándar de la industria para la clasificación de texto. Es computacionalmente eficiente y efectivo para resaltar términos que son importantes para un documento específico dentro de una colección. A diferencia de los embeddings de LLMs, no requiere hardware especializado (GPU) ni dependencias costosas, alineándose con nuestro objetivo de una solución desplegable y robusta.
3. **Optimización de Características:**
    - **N-gramas:** Se configuró el vectorizador para capturar tanto unigramas como bigramas (`ngram_range=(1, 2)`). Esto permite al modelo reconocer términos compuestos clave como "blood pressure" o "renal failure".
    - **Stop Words:** Se utilizó una lista combinada de *stop words* en inglés junto con una lista personalizada para eliminar ruido y términos de dominio genéricos que no aportan valor predictivo.
    - **Dimensionalidad:** Se limitó el número de características a las 5,000 más relevantes (`max_features=5000`) para equilibrar el poder predictivo con el costo computacional y mitigar el riesgo de sobreajuste.

B. Preparación de Etiquetas (Multi-Label):

Para manejar la naturaleza multi-etiqueta, se utilizó el `MultiLabelBinarizer` de Scikit-learn. Este componente transforma la columna `group` (ej: "cardiovascular|oncological") en un formato de matriz binaria, donde cada columna representa una categoría y un `1` indica la presencia de esa etiqueta. Se fijó el orden de las clases para garantizar la consistencia entre el entrenamiento y la inferencia.

3. Selección y Diseño de la Solución

Tras una fase de experimentación que incluyó modelos como Regresión Logística, SVM, Redes Neuronales y arquitecturas basadas en Transformers (BERT, BioBERT), se seleccionó un enfoque de **Machine Learning clásico** por su equilibrio entre rendimiento, interpretabilidad y facilidad de despliegue.

A. Modelo Seleccionado: XGBoost con One-Vs-Rest (OVR)

- **Modelo Base:** Se utilizó **XGBClassifier**, una implementación optimizada de *gradient boosting* reconocida por su alto rendimiento en datos tabulares y vectorizados.
    - **Justificación:** XGBoost maneja eficientemente la escasez de los datos TF-IDF, ofrece un control granular sobre el sobreajuste mediante regularización (L1/L2) y submuestreo, y proporciona métricas de importancia de características que facilitan la interpretabilidad.
- **Estrategia Multi-etiqueta:** Se envolvió el `XGBClassifier` en un `OneVsRestClassifier` (OVR).
    - **Justificación:** La estrategia OVR entrena un clasificador binario independiente para cada una de las cuatro categorías. Para una nueva predicción, cada clasificador emite una probabilidad, y el sistema asigna todas las etiquetas cuyo clasificador supera un umbral predefinido. Este enfoque es robusto, escalable y conceptualmente simple, adaptándose perfectamente al problema.

B. Configuración y Reproducibilidad:

Para facilitar la experimentación y asegurar la reproducibilidad, se definieron varias configuraciones de hiperparámetros (`default`, `fast_training`, `high_performance`, `regularized`), permitiendo ajustar el modelo para velocidad de prototipado o para maximizar el rendimiento final.

4. Validación y Métricas

Se implementó un riguroso pipeline de validación para asegurar que el modelo no solo tuviera un buen rendimiento, sino que también generalizó correctamente a datos no vistos.

A. Estrategia de Validación:

- **División Train/Test:** Los datos se dividieron en conjuntos de entrenamiento y prueba (80/20). Aunque se utilizó una estratificación estándar, se reconoce como mejora futura el uso de una estratificación iterativa específica para problemas multi-etiqueta.
- **Validación Cruzada (Cross-Validation):** Se realizó una validación cruzada de 5 pliegues (*5-fold CV*) para obtener una estimación más robusta del rendimiento del

modelo y asegurar que los resultados no dependieran de una única división de datos.

B. Métricas de Evaluación:

La métrica principal fue el **F1-Score Ponderado (Weighted F1-Score)**, ya que ofrece un balance entre precisión y recall, ajustado por la frecuencia de cada clase.

Resultados Obtenidos (en el conjunto de prueba):

Métrica	Valor	Interpretación
F1-Score (Ponderado)	**90.8%**	**Excelente rendimiento general, balanceando precisión y recall.**
Accuracy	80.79 %	El modelo predice correctamente el conjunto exacto de etiquetas en el 80.8% de los casos.
Precision (Macro)	97.02 %	Cuando el modelo predice una etiqueta, es correcto el 97% de las veces en promedio.
Recall (Macro)	85.62 %	El modelo identifica el 85.6% de todas las etiquetas relevantes en promedio.
ROC-AUC (Macro)	97.40 %	Sobresaliente capacidad del modelo para distinguir entre clases.

Resultados por Clase (F1-Score):

- Cardiovascular: **93.22%**
- Neurológico: **93.56%**
- Hepatorenal: **86.23%**
- Oncológico: **90.09%**

El rendimiento es consistentemente alto en todas las categorías, con una ligera caída en `Hepatorenal`, probablemente debido a una menor frecuencia o mayor ambigüedad en sus términos.

C. Matriz de Confusión:

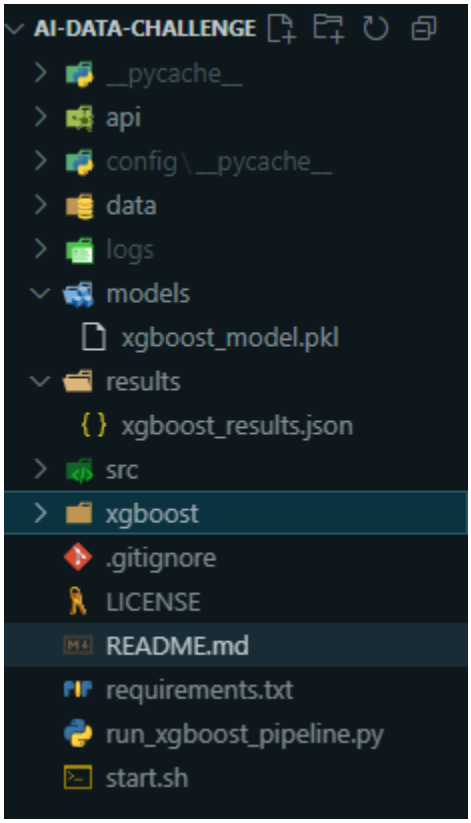
Se generaron matrices de confusión para cada clase bajo la estrategia One-Vs-Rest. Estas visualizaciones confirmaron la baja tasa de falsos positivos y falsos negativos en general, validando la alta precisión y recall del modelo.

5. Presentación y Reporte

La solución se entregó como un sistema integral y listo para producción, compuesto por los siguientes artefactos:

- **API de Inferencia (Flask):** Un backend ligero que expone un endpoint `/api/predict`. Recibe un texto y devuelve las categorías predichas junto con sus probabilidades y un nivel de confianza.
- **Dashboard Interactivo (Next.js):** Una interfaz de usuario web que consume la API, permitiendo a los usuarios ingresar texto y obtener clasificaciones en tiempo real. También muestra métricas de rendimiento del modelo.
- **Explicabilidad (SHAP):** Se integró opcionalmente la librería SHAP para analizar las contribuciones de términos específicos a las predicciones, aumentando la transparencia y confianza en el modelo.

- **Artefactos Versionados:** El modelo entrenado (`xgboost_model.pkl`), los resultados de la evaluación (`results.json`) y las visualizaciones (`.png`) se guardan sistemáticamente, garantizando la trazabilidad de los experimentos.



code.png

6. Repositorio y Buenas Prácticas

El proyecto se organizó en un repositorio de GitHub siguiendo las mejores prácticas de desarrollo de software y MLOps.

- **Estructura Modular:** El código se dividió en módulos lógicos (configuración, entrenamiento, evaluación, API), promoviendo la reutilización y el mantenimiento.
- **Orquestación con CLI:** Se creó un script de línea de comandos (`run_xgboost_pipeline.py`) que centraliza todas las tareas (entrenar, evaluar, analizar), facilitando la ejecución del pipeline completo.
- **Calidad de Código:** Se adhirió al estándar de estilo PEP8 para garantizar la legibilidad y consistencia del código Python.
- **Gestión de Dependencias:** Todas las librerías necesarias se documentaron en un archivo `requirements.txt`, asegurando la reproducibilidad del entorno.

Reflexiones y Mejoras Futuras:

La solución actual es robusta, eficiente y explicable. Sin embargo, se identifican las siguientes áreas de mejora:

1. **Optimización de Umbrales:** Implementar una calibración de umbrales por clase para optimizar el punto de corte de decisión y mejorar aún más el F1-Score.
2. **Balanceo de Clases:** Aplicar técnicas como `scale_pos_weight` en XGBoost si se detecta un desbalance de clases significativo.
3. **Enfoque Híbrido:** Explorar el uso de embeddings pre-entrenados en dominios médicos (ej. BioClinicalBERT) como características de entrada para XGBoost, comparando el costo-beneficio con el enfoque TF-IDF.

7. Uso de V0 para la visualización de resultados

Prompt: Crea un dashboard médico profesional para clasificación de literatura médica con XGBoost.

FUNCIONALIDADES PRINCIPALES:

- Dashboard para 4 categorías médicas: cardiovascular, neurológico, hepatorenal, oncológico

- Métricas de rendimiento con visualización interactiva

- Predictor de texto médico en tiempo real

- Matriz de confusión visual 4x4

- Gráficos de importancia de características

- Historia de entrenamiento con curvas

COMPONENTES:

1. Header con título "Medical AI Dashboard" y métricas principales en cards

2. Grid de métricas: Accuracy (85%), Precision (82%), Recall (79%), F1-Score (80%)

3. Predictor en tiempo real: textarea + botón "Predecir" + resultados con probabilidades por categoría

4. Matriz de confusión como heatmap interactivo

5. Bar chart horizontal con top 10 características más importantes

6. Line chart con curvas de loss/accuracy durante entrenamiento

7. Galería de ejemplos médicos con predicciones

DATOS Y API:

- API Flask en http://localhost:5000 con endpoints:

  * GET /api/health - estado del sistema

  * POST /api/predict - predicciones {"text": "texto médico"}

  * GET /api/statistics - métricas del modelo

  * GET /api/demo-examples - ejemplos para probar

- Datos estáticos en JSON disponibles si la API falla

DISEÑO:

- Tema médico profesional (azul #2563eb, blanco, grises sutiles)

- Layout responsive con CSS Grid/Flexbox

- Cards con sombras sutiles y bordes redondeados

- Animaciones suaves en hover y transiciones

- Tipografía clara y legible

- Iconos médicos apropiados

FUNCIONALIDADES UX:

- Loading states para predicciones

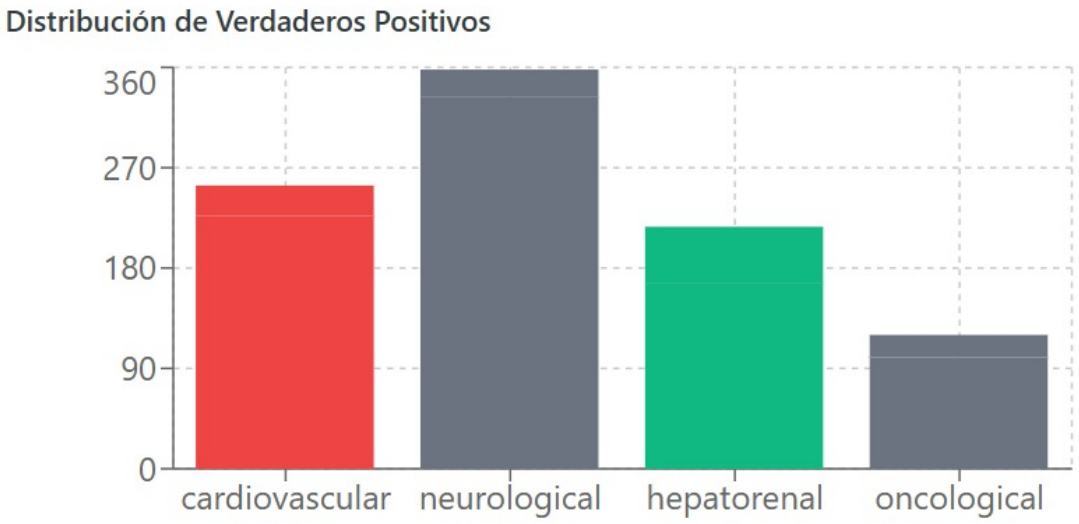
- Error handling graceful

- Tooltips informativos en gráficos

- Colores codificados por categoría médica

- Feedback visual para alta/baja confianza en predicciones

Integra con fetch() a la API Flask. Incluye manejo de errores y estados de carga.



Métricas por Categoría

Precisión y recall por especialidad médica

Cardiovascular

93.2% F1

97.4%	89.4%	254
Precisión	Recall	Muestras

Neurological

93.6% F1

93.8%	93.3%	358
Precisión	Recall	Muestras

Hepatorenal

86.2% F1

98.8%	76.5%	217
Precisión	Recall	Muestras

Oncological

90.1% F1

98.0%	83.3%	120
Precisión	Recall	Muestras

Matriz de Confusión

Verdaderos positivos por categoría

Cardiovascular

227

VP de 254

Neurological

334

VP de 358

Hepatorenal

166

VP de 217

Oncological

100

VP de 120

Resultados de Cross-Validation

Métricas de rendimiento en validación cruzada

Accuracy

Media: 74.86%

Desviación: ±1.73%

F1-Score

Media: 87.67%

Desviación: ±0.99%

Comparación de Métricas en Validación Cruzada