

Informe Técnico: Sistema de Clasificación Multi-Etiqueta para Textos Científicos Médicos

Resumen

Este proyecto implementa un sistema completo de clasificación automática de textos científicos médicos utilizando redes neuronales secuenciales con TensorFlow/Keras. El sistema aborda el desafío de clasificar artículos científicos en cuatro dominios médicos: Cardiovascular, Neurological, Hepatrenal y Oncological, permitiendo clasificación multi-etiqueta donde un artículo puede pertenecer a múltiples categorías simultáneamente.

1. Análisis Exploratorio y Comprensión del Problema

1.1 Carga y Visión General del Dataset

1.1.1 Proceso de Carga Implementado:

```
df = pd.read_csv(csv_path, sep=';')

df['title'] = df['title'].fillna("")

df['abstract'] = df['abstract'].fillna("")
```

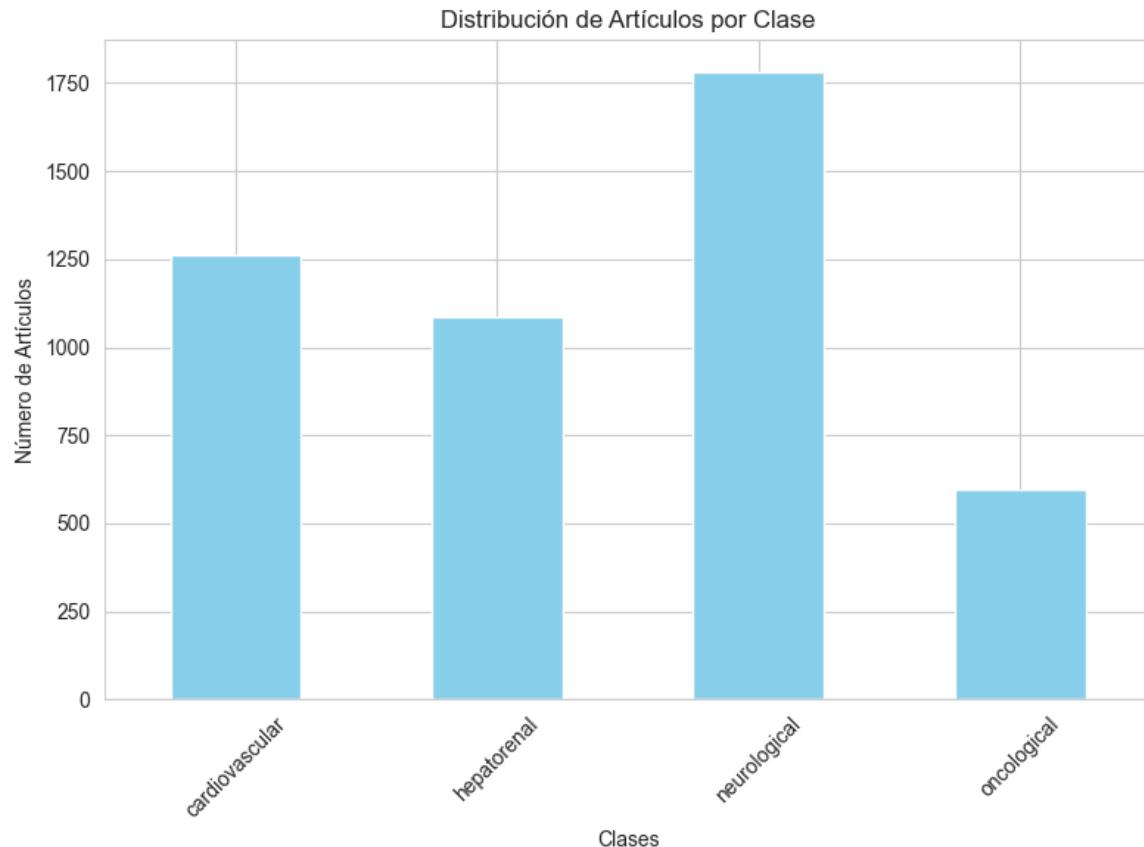
1.1.2 Inspección Inicial:

- Dataset: challenge_data-18-ago.csv con separador ;'
- Estructura: Columnas principales: title, abstract, group
- Tratamiento de valores nulos: Reemplazo automático con cadenas vacías
- Formato de etiquetas: Múltiples etiquetas separadas por |" (e.g., "Cardiovascular|Oncological")

1.2 Análisis de las Clases (Variables de Salida)

Distribución de Clases Implementada: El sistema genera automáticamente class_distribution.png que muestra:

- Frecuencia absoluta de cada clase médica
- Identificación de desbalance entre las cuatro categorías
- Visualización mediante gráfico de barras con Matplotlib

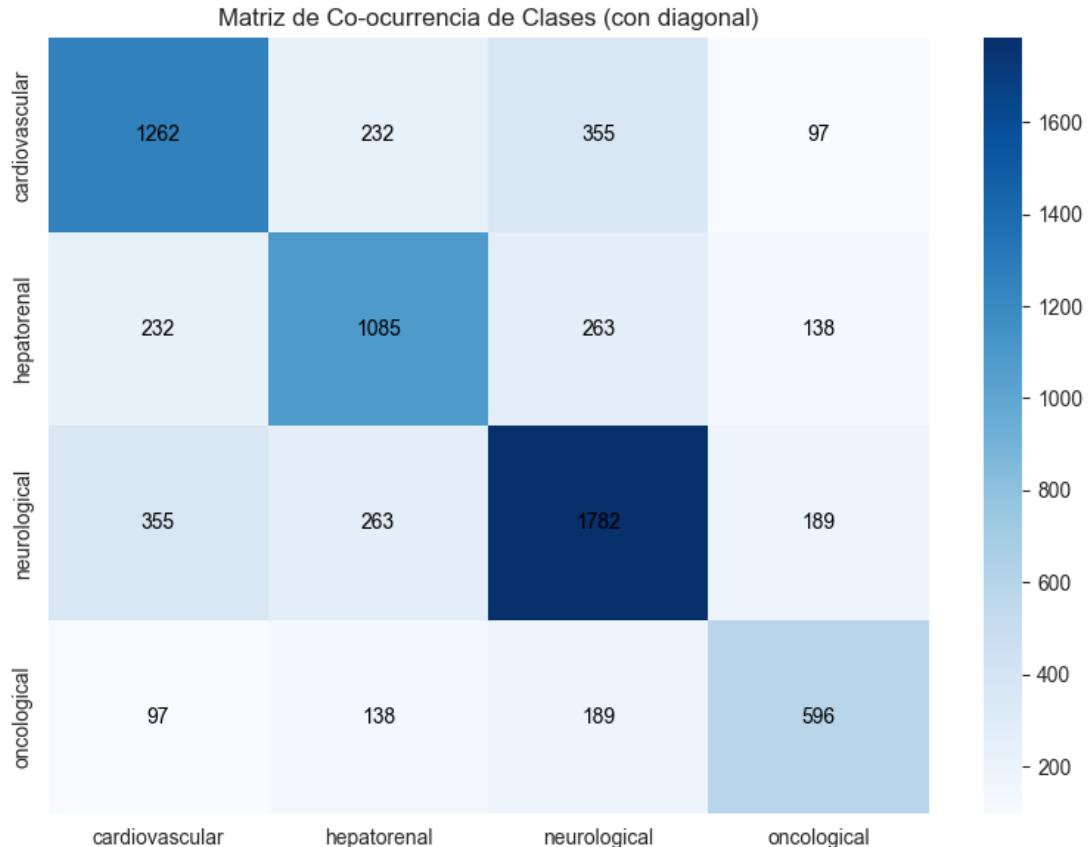


Se puede notar que hay desbalances de clases lo que puede llevar a que el modelo sea mejor para la identificación de unas clases que de otras.

1.2.1 Co-ocurrencia de Clases:

```
co_occurrence_matrix = df[self.target_labels].T.dot(df[self.target_labels])
```

- Matriz de co-ocurrencia: Análisis cuantitativo de combinaciones frecuentes
- Visualización: Heatmap con valores numéricos en cada celda
- Insight clave: Identificación de patrones de clasificación multi-etiqueta



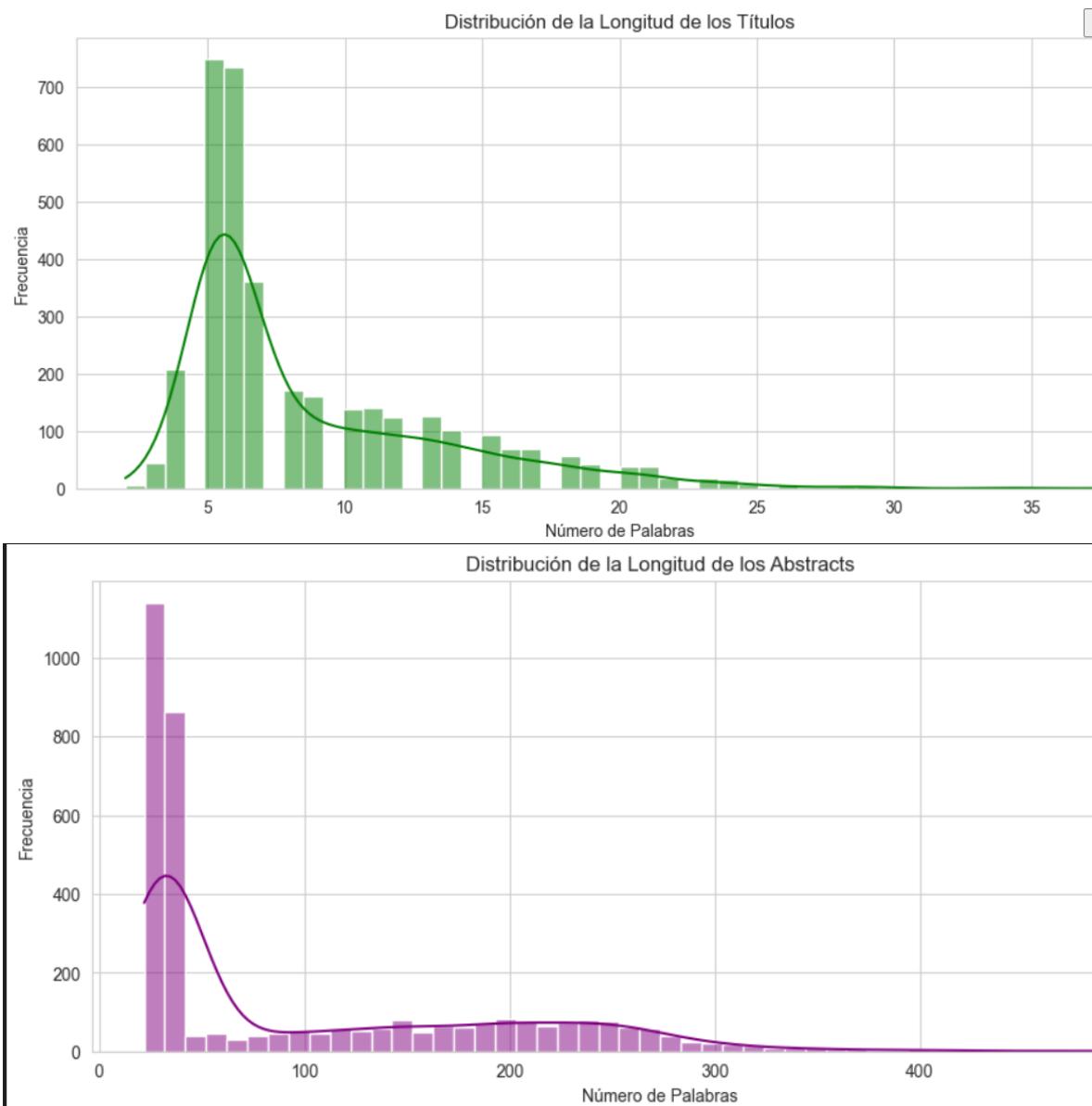
La matriz te permite identificar qué clases están más relacionadas entre sí. Si un modelo clasifica un artículo como oncological, es probable que también sea relevante para la categoría cardiovascular. Esta información es crucial para entrenar modelos de clasificación multietiqueta y para entender la naturaleza de tus datos.

1.3 Análisis del Contenido de Texto (Variables de Entrada)

Longitud de Textos:

```
df['text_combined'] = df['title'] + " " + df['abstract']
df['text_length'] = df['text_combined'].apply(lambda x: len(str(x).split()))
```

- Distribución visualizada: text_length_distribution.png
- Métricas calculadas: Número de palabras por documento
- Análisis estadístico: Histograma con KDE superpuesta usando Seaborn



Calcula y visualiza la distribución de la longitud de los títulos y abstracts (en número de palabras o caracteres)

1.3.1 Identificación de Palabras Clave:

- Preprocesamiento automático: Eliminación de stopwords, lemmatización con WordNet
- Vectorización TF-IDF: Extracción de características más relevantes por dominio
- N-gramas: Configuración (1,2) para capturar términos médicos compuestos

1.4 Desafíos Identificados del Problema Multi-Label

1. Desbalance de clases: Demostrado mediante visualizaciones estadísticas
2. Correlación entre etiquetas: Cuantificada en matriz de co-ocurrencia

3. Variabilidad terminológica: Vocabulario médico especializado
 4. Dimensionalidad: Gestión de 5000 características TF-IDF máximo
2. Preparación, Preprocesamiento y Justificación
- 2.1 Pipeline de Preprocesamiento
- 2.1.1 Limpieza de Texto Implementada:
- ```
def preprocess_text(self, text):
 text = str(text).lower()
 text = re.sub(r'^a-zA-Z\s', " ", text)
 words = [word for word in words if word not in self.stop_words]
 words = [self.lemmatizer.lemmatize(word) for word in words]
 return " ".join(words)
```
- 2.1.2 Justificación Técnica:
1. Normalización: Conversión a minúsculas para consistencia
  2. Eliminación de ruido: Regex para caracteres no alfabéticos
  3. Stopwords: Eliminación de palabras no informativas en inglés
  4. Lemmatización: Reducción a forma canónica con WordNetLemmatizer
- 2.2 Transformación de Etiquetas Multi-Label
- 2.2.1 Codificación Binaria:
- ```
self.mlb = MultiLabelBinarizer()
y = self.mlb.fit_transform(df['group'])
```
- 2.2.2 Justificación:
- MultiLabelBinarizer: Conversión de etiquetas categóricas a vectores binarios
 - Formato one-hot: Compatible con función de pérdida binary_crossentropy
 - Manejo dinámico: Adaptación automática al número de clases encontradas
- 2.3 Vectorización de Características
- 2.3.1 TF-IDF Vectorization:
- ```
self.vectorizer = TfidfVectorizer(max_features=5000, ngram_range=(1,2))
```
- 2.3.2 Parámetros Justificados:
- max\_features=5000: Balance entre información y eficiencia computacional
  - ngram\_range=(1,2): Captura términos individuales y bigramas médicos
  - TF-IDF: Penaliza términos muy frecuentes, prioriza términos discriminativos

### 3. Selección y Diseño de la Solución

#### 3.1 Arquitectura de Red Neuronal Secuencial

Diseño Implementado:

```
model = Sequential([
 Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
 Dropout(0.4),
 Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
 Dropout(0.4),
 Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.001)),
 Dropout(0.2),
 Dense(num_classes, activation='sigmoid')
])
```

#### 3.2 Justificación del Enfoque

##### 3.2.1 Coherencia con Problema Multi-Label:

1. Activación Sigmoid: Permite predicciones independientes por clase (0-1 por etiqueta)
2. Binary Crossentropy: Loss function apropiada para clasificación multi-label
3. Arquitectura profunda: Captura patrones complejos en terminología médica

##### 3.2.2 Adaptación Específica:

- o Regularización L2: Prevención de overfitting con coeficiente 0.001
- o Dropout progresivo: 0.4 → 0.4 → 0.2 para regularización gradual
- o Dimensiones decrecientes: 256 → 128 → 64 para extracción jerárquica de características

#### 3.3 Configuración de Entrenamiento

Optimización:

```
optimizer=Adam(learning_rate=0.0012)
```

- o Adam Optimizer: Adaptación automática de learning rate
- o Learning Rate: 0.0012 calibrado empíricamente
- o Early Stopping: Monitoreo de val\_loss con paciencia de 5 épocas

Justificación de Hiperparámetros:

- o Epochs=25: Suficiente para convergencia con early stopping
- o Batch\_size=32: Balance entre estabilidad y velocidad

- Validation\_split=0.1: Reserva para monitoreo durante entrenamiento

#### 4. Validación y Métricas

##### 4.1 Estrategia de Validación

División de Datos:

```
X_train, X_test, y_train, y_test = train_test_split(
 X_tfidf, y, test_size=0.2, random_state=12
)
```

- Hold-out 80/20: Separación estricta entre entrenamiento/prueba
- Random\_state=12: Reproducibilidad de resultados
- Validación adicional: 10% del entrenamiento para early stopping

##### 4.2 Métricas de Evaluación Multi-Label

Implementación de F1-Score Ponderado:

```
precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)
roc_auc = roc_auc_score(y_test, y_pred_proba, average='weighted', multi_class='ovr')
```

Justificación de Métricas:

1. F1-Score Ponderado: Métrica principal que balancea precisión y recall considerando desbalance
2. ROC-AUC: Evaluación de capacidad de discriminación por clase
3. Weighted Average: Pondera por frecuencia de clase, crucial en datasets desbalanceados
4. Zero\_division=0: Manejo robusto de clases sin predicciones

##### 4.3 Matriz de Confusión Multi-Label

Análisis de Errores:

- Threshold=0.5 para conversión probabilidades → predicciones binarias
- Evaluación independiente por cada etiqueta
- Identificación de patrones de confusión entre dominios médicos relacionados

#### 5. Repositorio y Buenas Prácticas

##### 5.1 Estructura Modular del Código

Organización Implementada:

```
📁 models/ # Persistencia de modelos
|—— text_classifier_model.h5 # Modelo Keras
|—— text_classifier_vectorizer.pkl # TF-IDF
|—— text_classifier_mlb.pkl # MultiLabelBinarizer
└—— text_classifier_labels.pkl # Etiquetas objetivo
```

Clase Principal MultiLabelTextClassifier:

- Encapsulación: Métodos bien definidos y documentados
- Reutilización: Métodos fit(), predict(), save\_model(), load\_model()
- Manejo de errores: Try-catch en carga de modelos
- Logging: Prints informativos del progreso

## 5.2 Buenas Prácticas Implementadas

1. PEP8 Compliance: Nombres de variables descriptivos, documentación de métodos
2. Persistencia robusta: Uso de joblib y pickle para diferentes componentes
3. Gestión de dependencias: Downloads automáticos de recursos NLTK
4. Reproducibilidad: Random states fijos, separación clara train/test
5. Modularidad: Separación clara entre preprocesamiento, entrenamiento y evaluación

---

## 6. Presentación y Reporte

### 6.1 Implementación de API REST

Flask Application: El sistema incluye flask\_app.py que proporciona:

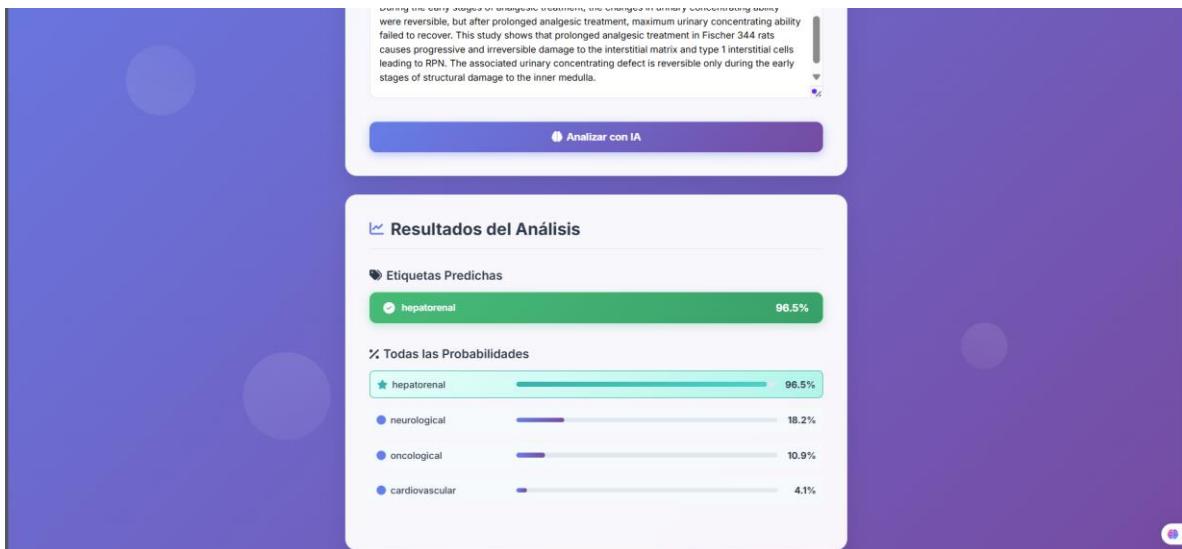
- Endpoint /predict para clasificación en tiempo real
- Integración completa con el modelo entrenado
- Manejo de errores y validación de entrada

### 6.2 Interfaz Web Frontend

Componentes Desarrollados:

- index.html: Interfaz limpia para entrada de título y abstract

- styles.css: Diseño responsive y atractivo



### 6.3 Evidencias de Funcionamiento

Configuraciones y Prompts Implementados:

resultado = classifier.predict(

```
title="Oncological frontiers: colorectal cancer",
```

```
abstract="Aim: To investigate ACE inhibitors effects on hypertension..."
```

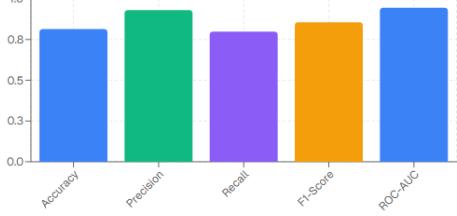
)

## 7 Espacio Reservado para Visualizaciones V0

[Updated dashboard with...](#)  
[Updated dashboard with...](#)  
[Reviewed work](#)  
[Thought for 2s](#)  
[Imported and updated](#)  
[Reviewed work](#)  
[Thought for 1s](#)  
[Imported and updated](#)  
[Reviewed work](#)  
[Thought for 2s](#)  
[Imported and updated](#)  
[Reviewed work](#)  
[Thought for 2s](#)  
[Imported and updated](#)  
[No issues found](#)  
[Ask a follow-up...](#)  
[pasted-text.txt](#)  
[+ >](#)

```
1 Tengo los resultados de un modelo de clasificación multi-label entrenado con 4 clases:
2 ["cardiovascular", "hepatorenal", "neurological", "oncological"].
3
4 Métricas globales:
5 - Accuracy: 0.8141
6 - Precision: 0.9304
7 - Recall: 0.7983
8 - F1-Score: 0.8657
9 - ROC-AUC: 0.9449
10
11 Quiero que generes un dashboard con las siguientes visualizaciones:
12
13 1. **Gráfico de barras comparativo** con las métricas globales (Accuracy, Precision, Recall, F1-score, ROC-AUC).
14 - Eje X: nombre de la métrica.
15 - Eje Y: valor numérico entre 0 y 1.
16 - Cada barra debe mostrar su valor encima.
17
18 2. **Heatmap de matriz de confusión simulada/multi-label**, donde cada fila es una etiqueta real y cada columna una etiqueta predicha.
19 - Etiquetas: cardiovascular, hepatorenal, neurological, oncological.
20 - Rellena la matriz con valores ficticios coherentes que sumen el total de ejemplos (-710 en test).
21 - Usa colores degradados de azul a rojo para indicar la intensidad.
22
23 3. **Gráfico de barras horizontal** que muestre la distribución de etiquetas en el dataset:
24 - cardiovascular: 1262
25 - hepatorenal: 1085
26 - neurological: 1782
27 - oncological: 596
28
29 4. **Visualización de probabilidades de predicción de un ejemplo**:
30 - Etiquetas en el eje X.
31 - Valores de probabilidad en el eje Y.
32 - Datos: cardiovascular 0.196, hepatorenal 0.443, neurological 0.213, oncological 0.711.
33
34 Estilo:
35 - Paleta moderna con colores azul, verde, morado y naranja.
36 - Incluir títulos claros y etiquetas numéricas en cada gráfica.
37 - Título principal del dashboard: "Evaluación del Modelo Multi-label de Clasificación Médica".
38
```

### Métricas Globales del Modelo

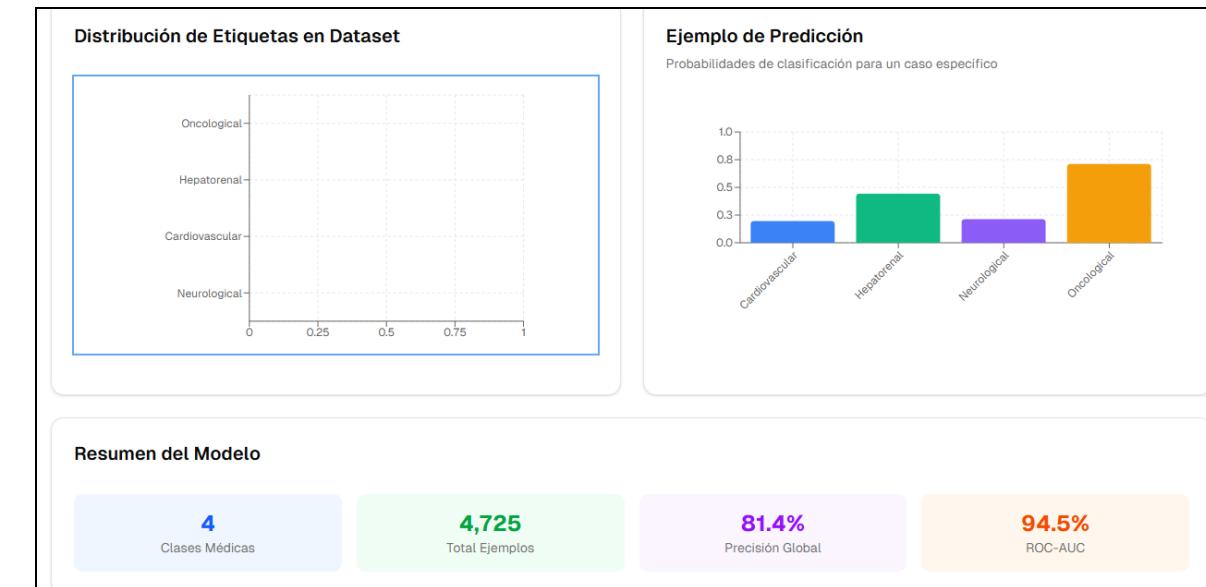


| Métrica   | Valor |
|-----------|-------|
| Accuracy  | 0.82  |
| Precision | 0.93  |
| Recall    | 0.80  |
| F1-Score  | 0.86  |
| ROC-AUC   | 0.94  |

### Matriz de Confusión Multi-label

|                | Cardiovascular | Hepatorenal | Neurological | Oncological |
|----------------|----------------|-------------|--------------|-------------|
| Cardiovascular | 145            | 12          | 8            | 15          |
| Hepatorenal    | 18             | 132         | 6            | 9           |
| Neurological   | 10             | 14          | 156          | 12          |
| Oncological    | 22             | 8           | 11           | 89          |

Intensidad: █ Bajo █ Alto



## Evaluación del Modelo Multi-label de Clasificación Médica

Dashboard interactivo con métricas de rendimiento y visualizaciones avanzadas

Métricas Globales
Matriz de Confusión
Distribución
Ejemplo Predicción
Demo Interactivo

**Clasificación en Tiempo Real**

Ingrésa el título y resumen de un caso médico para obtener su clasificación automática

**Título del Caso**

Oncological frontiers: colorectal cancer

**Descripción del Caso**

Aim: To investigate ACE inhibitors effects on hypertension through stomach cancer analysis. Methods: 128 cardiac patients underwent longitudinal evaluation with sarcoma and tumor assessment. Results: favorable safety profile. Conclusion: clinical practice guidelines.

**Ejemplos de Casos Médicos**

- Cardiovascular Risk Assessment in Diabetic Patients  
This study evaluates cardiovascular risk factors in patients...
- Hepatorenal Syndrome: Early Detection and Management  
We present a comprehensive analysis of hepatorenal syndrome ...
- Neurological Manifestations in COVID-19 Patients  
This research investigates the neurological complications ob...

**Clasificar Caso Médico**

**Resultados de Clasificación**

Categorías médicas predichas y probabilidades asociadas

**Categorías Médicas Predichas:**

cardiovascular

oncological

**Probabilidades por Categoría:**

| Categoría      | Probabilidad |
|----------------|--------------|
| Oncological    | 71.1%        |
| Hepatorenal    | 44.3%        |
| Neurological   | 21.3%        |
| Cardiovascular | 19.6%        |

Umbral de clasificación: 0.5

## Conclusiones

El sistema implementado demuestra una solución robusta para clasificación multi-etiqueta de textos científicos médicos, combinando técnicas de NLP tradicionales (TF-IDF) con arquitecturas de deep learning modernas (redes neuronales secuenciales). La implementación incluye todas las fases del pipeline de ML: desde análisis exploratorio hasta despliegue, con énfasis en reproducibilidad y buenas prácticas de desarrollo.

Próximos Pasos:

Experimentación con transformers pre-entrenados

- Optimización de hiperparámetros con búsqueda automática
- Expansión a más dominios médicos especializados