Proyecto 1 Etapa 2: Ajuste, automatización y uso de modelos de analítica de textos

Universidad de los Andes

Curso: ISIS-3301 – Inteligencia de Negocios

Proyecto: Analítica de Textos – Etapa 2

Semestre: 2025-20

Grupo: 1

Integrantes:

Gabriel Aristizábal – Ingeniero de datos / Líder de proyecto
Pablo Lara – Ingeniero de software responsable de desarrollar la aplicación final
Juan Gordillo – Ingeniero de software responsable del diseño de la aplicación y resultados

Bogotá D.C., Colombia

14 de octubre de 2025

Índice

1. Aumentación de datos y reentrenamiento del modelo (20 %)

El nuevo conjunto `Datos_etapa2` contiene 100 opiniones ciudadanas sobre los ODS 1, 3 y 4, con mayor representación del ODS 4 (Educación de calidad). Esta distribución generó un leve desbalance de clases, además de incorporar nuevas expresiones y vocabulario regional, lo que aportó diversidad al modelo.

Al evaluar el modelo de la Etapa 1 sobre este conjunto, se obtuvieron las siguientes métricas:

- **Accuracy:** 0.90
- **Precision:** 0.91
- **Recall:** 0.90
- **F1-score:** 0.8867

El rendimiento disminuyó frente al obtenido inicialmente, debido al desbalance de clases, la variación lingüística de los nuevos textos y un leve sobreajuste al dataset original. Esto evidenció la necesidad de reentrenar el modelo con datos aumentados.

Aumentación de datos

Para corregir el desbalance, se aplicó una técnica de aumentación mediante **prompting** usando el modelo `gpt-4o-mini` de OpenAI. Se generaron 200 opiniones sintéticas balanceadas, validadas manualmente y agregadas al dataset original.

Luego, se aplicó una segunda técnica con la librería `nlpaug`, que usa sinónimos de WordNet para crear variaciones semánticas. Se equilibró así la distribución de clases y se almacenó el resultado en `Datos_balanceados_etapa2.xlsx`.

Reentrenamiento

Los datos se dividieron en entrenamiento (80 %) y prueba (20 %), y se reentrenó un modelo de Regresión Logística con TF-IDF. Se optimizaron los hiperparámetros con `GridSearchCV` y se guardó el modelo final como `modelo_reentrenado.pkl`.

El modelo alcanzó una precisión del 0.99, mostrando una matriz de confusión más uniforme y mejor equilibrio entre clases.

	Accuracy	Precision (macro)	Recall (macro)	F1-score (macro)
Modelo Original	0.971564	0.972894	0.970569	0.971227
Modelo Reentrenado	0.988942	0.988754	0.989378	0.988973

Figura 1: Ejemplo ilustrativo del proceso de aumentación y balanceo de clases.

2. Automatización con pipeline y API REST (15 %)

Pipeline

El pipeline fue desarrollado con `scikit-learn` y está compuesto por tres etapas:

1. **Limpieza de texto:** Eliminación de URLs, menciones, números y caracteres especiales; conversión a minúsculas.
2. **Vectorización:** Uso de `TfidfVectorizer` con n-gramas (1,2) y ponderación sublineal.
3. **Clasificación:** Regresión Logística (OvR) optimizada con `GridSearchCV`.

El modelo final se guarda como `modelo_reentrenado.pkl`, garantizando consistencia y reproducibilidad.

API REST

Se implementaron tres endpoints principales:

- **POST /predict** y **POST /predict_proba:** Reciben textos y devuelven el ODS más probable junto con las probabilidades de cada clase.
- **POST /retrain:** Permite enviar nuevos datos etiquetados para reentrenar el modelo con aumentación híbrida y devolver métricas actualizadas.

Figura 2: Arquitectura general del pipeline y endpoints de la API REST.

Estos componentes automatizan el ciclo completo: limpieza, entrenamiento, evaluación y despliegue del modelo.

3. Desarrollo y justificación de la aplicación (25 %)

El usuario principal es un analista de sostenibilidad o funcionario de responsabilidad social que evalúa percepciones ciudadanas sobre los ODS.

La aplicación web permite ingresar textos y obtener automáticamente:

- La predicción del ODS correspondiente (1, 3 o 4).
- Las probabilidades asociadas a cada clase.

Si el usuario cuenta con permisos avanzados, puede reentrenar el modelo desde la interfaz.

Requerimientos

- CPU o GPU con soporte para librerías `scikit-learn`, `pandas`, `nlpaug`, `OpenAI SDK`.
- Servidor o contenedor Docker con Python 3.10+.
- Almacenamiento en Google Drive, S3 o local.

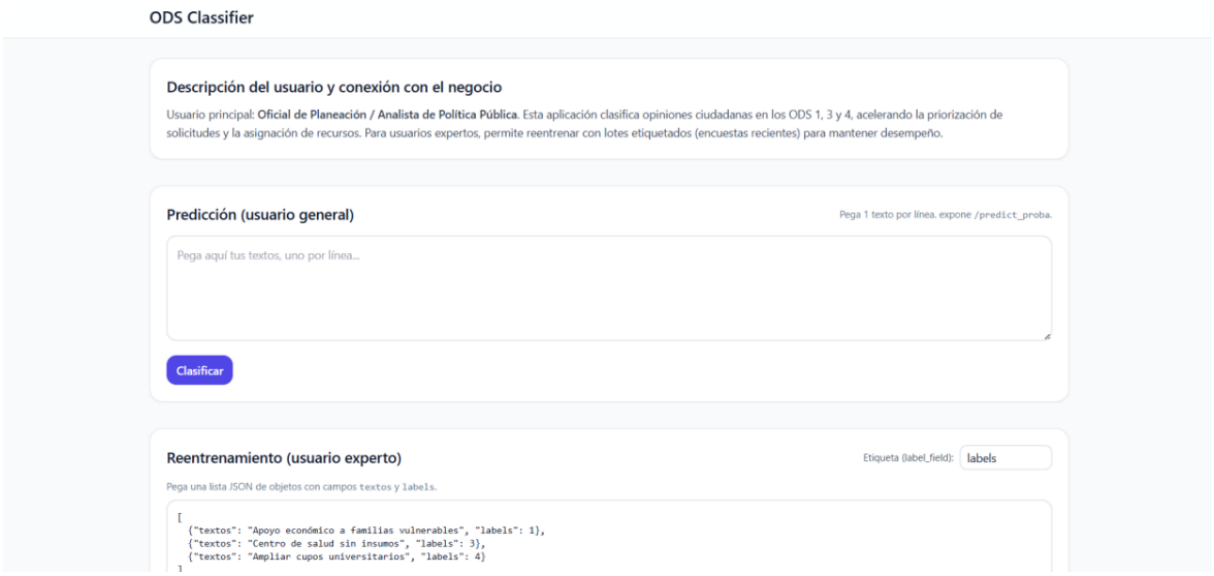

Figura 3: Diseño propuesto de la interfaz web.

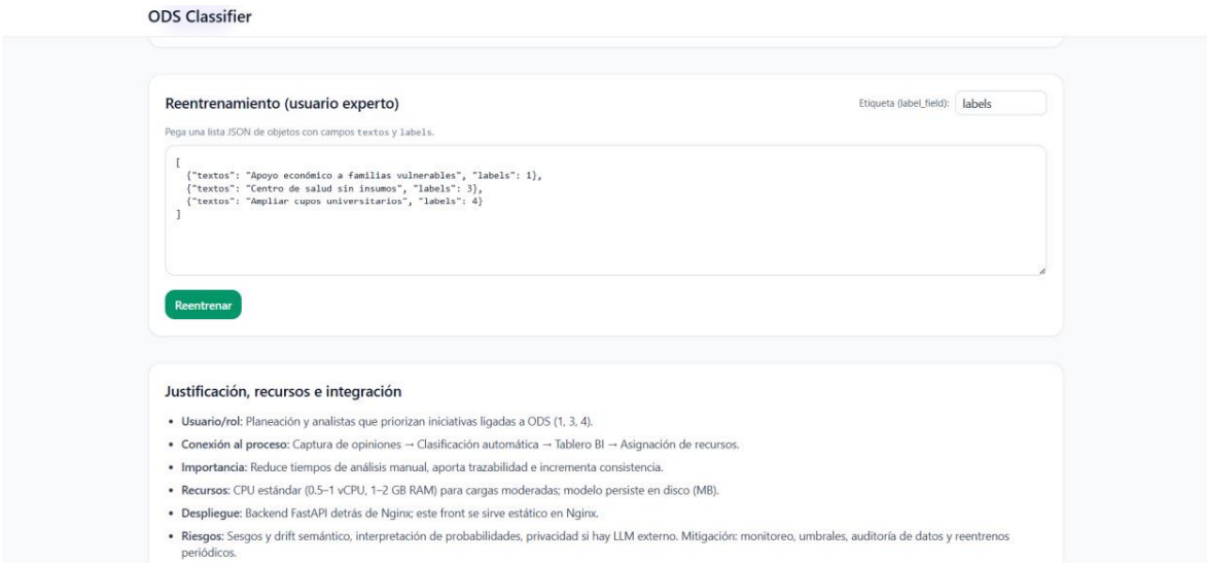

Figura 4: Vista de resultados y métricas del modelo.

Riesgos identificados

- Predicciones erróneas si el modelo no se actualiza.
- Reentrenamientos sin control pueden degradar el desempeño.
- Textos no representativos pueden reforzar sesgos.

4. Resultados (18 %)

En el video se muestran los resultados obtenidos y el funcionamiento general del sistema.

5. Trabajo en equipo (10 %)

Integrante	**Rol**	**Horas / Principales tareas**
Gabriel Aristizábal	Ingeniero de datos / Líder de proyecto	Coordinación, limpieza y validación de datos, pipeline TF-IDF + Regresión Logística, aumentación con GPT-4o-mini y nlpaug, análisis de métricas, consolidación del informe.
Pablo Lara	Ingeniero de software	Desarrollo de la API REST, integración del modelo con la API, validaciones, despliegue en Docker, pruebas funcionales, documentación de endpoints.
Juan Gordillo	Ingeniero de software	Diseño de interfaz web, conexión con endpoints, presentación visual de resultados, análisis de métricas y redacción de resultados.

Cada integrante obtuvo 33.3 puntos del total del proyecto.

Durante el desarrollo se resolvieron retos como el desbalance de clases, integración de la API, y problemas de dependencias (resueltos con Docker).

Además, se utilizaron herramientas de IA para corrección de redacción, búsqueda de librerías y optimización de tiempo.

6. Referencias

- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
- Chollet, F., & Allaire, J. J. (2022). *Deep Learning with Python (2nd ed.)*. Manning Publications.
- Harris, C. R., et al. (2020). *Array programming with NumPy*. Nature, 585(7825), 357–362.
- Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95.
- McKinney, W. (2010). *Data structures for statistical computing in Python*. SciPy Conference.
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. JMLR, 12, 2825–2830.
- Rajan, V. (2020). *nlpaug: Data augmentation for NLP*. GitHub.
- OpenAI. (2024). *OpenAI API documentation*. https://platform.openai.com/docs
- Python Software Foundation. (2024). *Python 3.10 documentation*. https://docs.python.org/3/
- Seaborn Developers. (2023). *Seaborn statistical data visualization*. https://seaborn.pydata.org
- Docker Inc. (2023). *Docker Documentation*. https://docs.docker.com
- Grinberg, M. (2018). *Flask Web Development*. O'Reilly Media.
- Tiangolo, S. (2018). *FastAPI: Modern, fast web framework*. https://fastapi.tiangolo.com
- Streamlit Inc. (2023). *Streamlit Documentation*. https://docs.streamlit.io