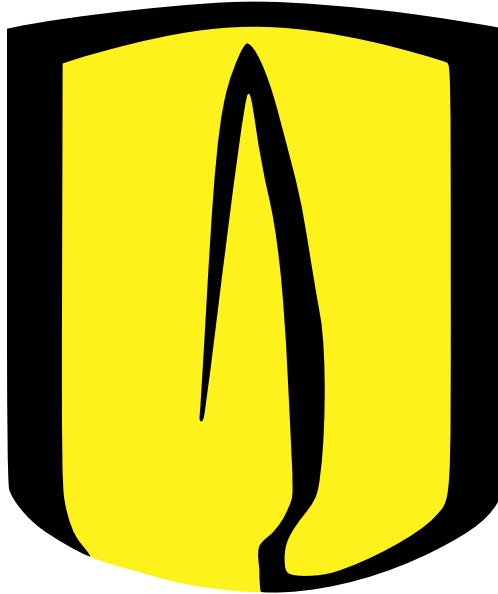


UNIVERSIDAD DE LOS ANDES
DEPARTAMENTO DE INGENIERIA DE SISTEMAS Y COMPUTACIÓN



PROYECTO 1: ANALITICA DE TEXTOS ETAPA 2

GRUPO 28

Juan Diego Rodríguez Barragán – 202221822

Juan Pablo Reyes Romero - 202310852

Santiago Pineda Quintero - 202023262

2025-20

Contenido

| | |
|--|----|
| 1. Sección 1: Aumentación de datos y reentrenamiento del modelo. | 3 |
| 2. Sección 2: Proceso de automatización del proceso de preparación de datos, construcción del modelo, persistencia del modelo y acceso por medio de API. | 8 |
| 3. Sección 3: Desarrollo de la aplicación y justificación..... | 9 |
| 3.1. Usuario, rol y conexión con el proceso | 9 |
| 3.2 Desarrollo de la aplicación web | 9 |
| 3.3 | 10 |
| 4. Sección 4: Resultados. | 10 |
| 5. Sección 5: Trabajo en equipo..... | 10 |
| 6. Referencias..... | 12 |

1. Sección 1: Aumentación de datos y reentrenamiento del modelo.

El objetivo principal de esta sección era incrementar la cantidad de ejemplos disponibles de las clases minoritarias del conjunto de datos. Esto se llevó a cabo con el fin de balancer las clases y mejorar de ese modo el desempeño del modelo de aprendizaje.

Como se estipulo desde un inicio, se empleó la técnica de aumentaciión de datos basada en prompting con modelos de lenguaje largos (LLMs), precisamente el que fue implementado fue el de openAI, GPT-4o-mini.

En el entorno se llevó a cabo la importación de librerías complementarias y que ayudaba en la configuración de parámetro globales como la semilla (SEED), siendo la responsable de garantizar la reproducibilidad, el modelo (openAI), y la relacion del balance objetivo, cuyo propósito era buscar que la clase minoritaria tuviera al menos el 80% de los ejemplos de la clase mayoritaria.

Teniendo esto en consideración, a continuación, se hará presentación de una serie de imágenes probatorias del proceso que se llevó a cabo. Ello haciendo uso del modelo ejecutado en la entrega pasada, y desarrollando métricas relevantes que son fundamentales para el apartado de comparación:

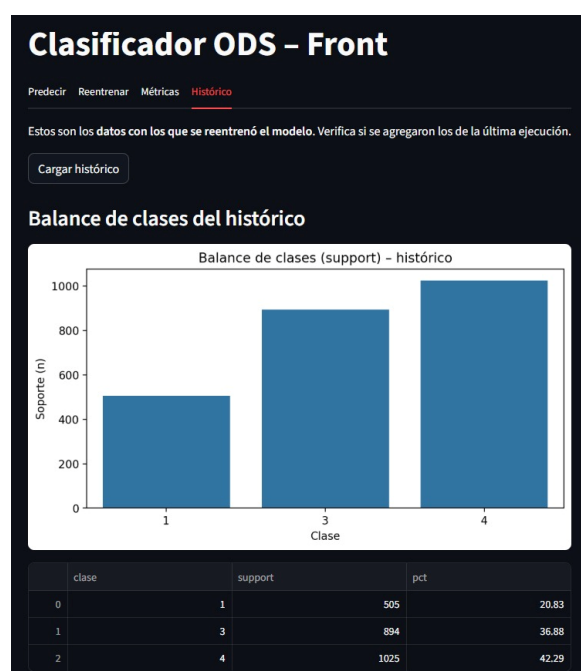


Imagen 1 – Balanceo de clases antes de aumentación

Como se aprecia en la imagen anterior, las clases se encontraban bastante desbalanceadas considerando la incidencia que tiene este hecho dentro del modelo. Por esa razón, se puede concluir que, en el proceso de entrenamiento previo, claramente era posible encontrarse con sesgos. Razón por la cuál era necesaria la aumentación.



Como se aprecia anteriormente, hubo una clara diferencia en cuanto al balanceo de las clases. Ello es fundamental considerando la labor de reentrenamiento que se tenía que llevar a cabo. Ofreciendo así una baja probabilidad de sesgo y una alta robustez del modelo desarrollado.



Imagen 4 - Visualización Front del Proyecto

Se desarrollo un front mediante el cuál la tarea de comparación y de visualización de los elementos que había que tener en consideración para esta etapa, fueran desplegados. En el mismo se ofrece el apartado de predicción, de reentrenamiento, de métricas, y de carga de datos a nivel histórico.

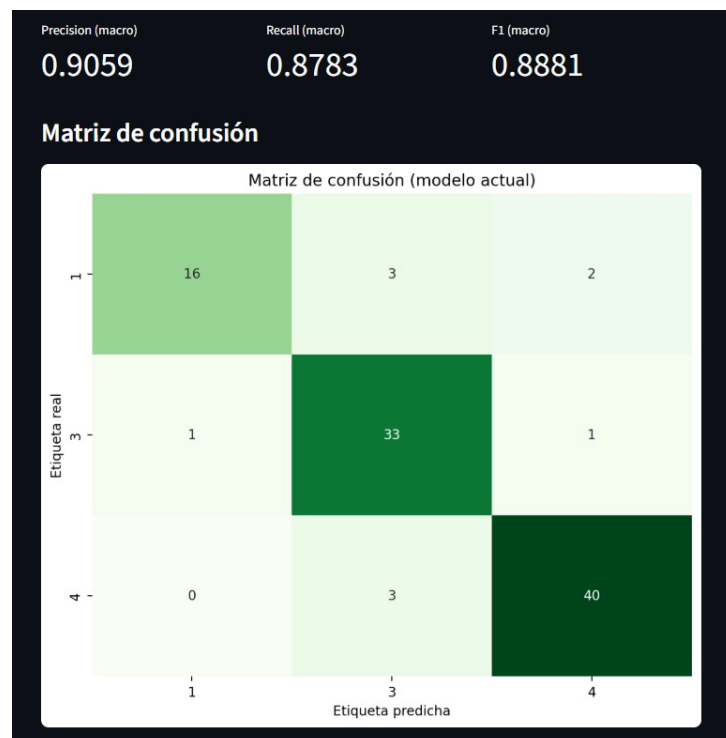


Imagen 5 – Matriz de confusión modelo actual

Ahora bien, como se puede observar en la imagen posterior, luego de haber ingresado el modelo, se pueden extraer las métricas importantes que determinan la calidad de la labor del modelo. Inicialmente, se observa un 0.9059 a nivel de precisión, dato bastante bueno ya que indica que el 90,59% de las predicciones positivas fueron correctas. Por otro lado, el recall fue de 0.8783, igualmente positivo, ya que el modelo fue capaz de recuperar un 87,83% de los casos positivos en el dataset. Finalmente, el F1 fue de 88,81%, lo cual es sustancial, ya que refleja que existe un equilibrio sólido en lo que consta las métricas anteriores.

Clasificador ODS – Front

Predecir **Reentrenar** Métricas Histórico


Elige cómo enviar datos

☐ Editor manual ☒ Subir CSV/Excel ☐ Subir JSON

Archivo CSV/XLSX/XLS con columnas textos y labels

Drag and drop file here
Limit 200MB per file • CSV, XLSX, XLS

Browse files

 **datos_proyecto_aumentados.csv** 1.8MB ×

Vista previa:

| | textos | labels |
|------|--|--------|
| 1857 | Con la excepción de las Islas Cook, Nauru y Tuvalu, la mayoría de los países insulares del Pacífico | 1 |
| 1858 | Por lo tanto, es de suma importancia determinar los efectos que cada uno de estos factores ejerce | 1 |
| 1859 | Sin embargo, los estándares de acreditación son amplios y no especifican lo que se necesita para | 4 |
| 1860 | La pobreza es ahora un fenómeno predominantemente rural, y los indígenas del norte experimentan | 3 |
| 1861 | Para ciertas sustancias o en ciertos contextos, puede asumir un patrón patológico que debe abordarse | 3 |
| 1862 | Los únicos países que exigen un máster para enseñar preescolar son Inglaterra, Francia, Islandia | 4 |
| 1863 | A menudo, esto será todo lo que necesite el paciente, pero en el caso de que sea necesario de | 30 |
| 1864 | En total, en 2016, se notificaron 30 273 casos de sarampión en los 21 países. A nivel mundial, | 3 |
| 1865 | Por esta razón y con el fin de incorporar medidas de ingreso al mismo tiempo, el presente estudio | 1 |
| 1866 | Cubre 27 estados miembros de la UE, así como Noruega e Islandia. La mayoría de los 14 ítems | 1 |

Imagen 6 – Vista de apartado de Reentrenamiento con Datos aumentados

Como se observa anteriormente, fue incluido el apartado mediante el cual se logró constatar la opción de reentrenamiento del modelo, esta vez viéndose los datos que fueron aumentados con prompting. Ello fue sustancial en el proceso considerando las instrucciones que fueron dadas desde un inicio.

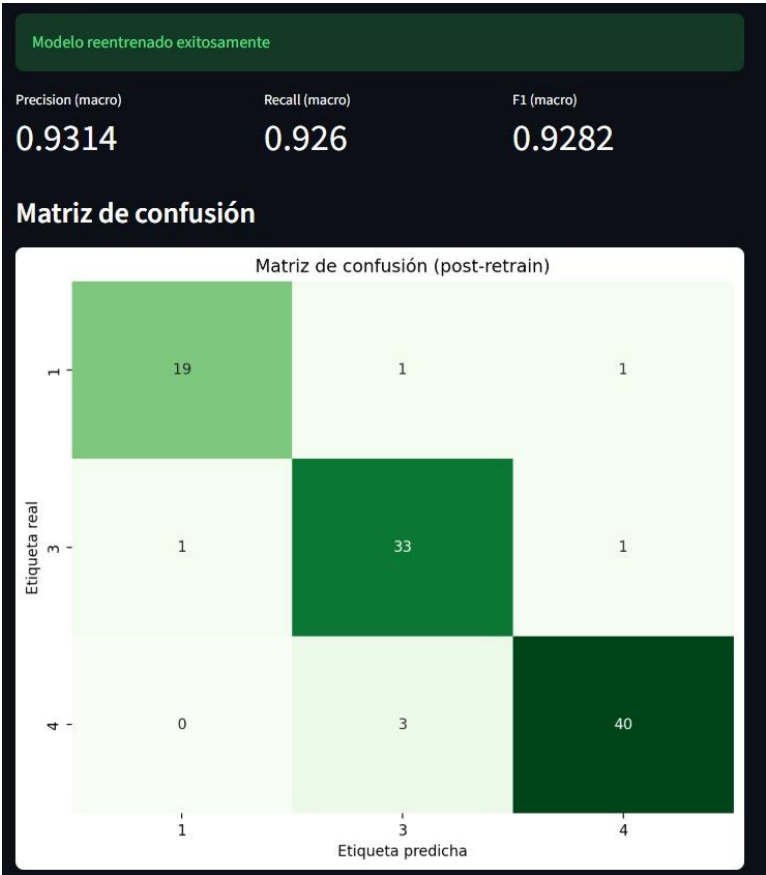


Imagen 7 – Matriz de confusión de modelo reentrenado

Teniendo en consideración lo estipulado en la imagen anterior, en el desarrollo de esta etapa se logró con éxito la labor de reentrenamiento, con la precisión de que fue completada la tarea con los nuevos datos de balance generados gracias al prompting empleando openAI. Ahora bien, con relación a la matriz posterior, es más que clara la mejoría en cuanto a las métricas. En primer lugar, se observa que la precisión fue del 93,14%, mejorando 2,55 puntos porcentuales. A través de dicha métrica se puede concluir que el modelo comete menos falsos positivos, valga la redundancia, mejorando la precisión. En segundo lugar, el recall ahora es de 92,6%, mejorando 4,77 puntos porcentuales, lo que implica que el modelo es capaz de detectar de mejor forma los casos reales. Finalmente, el F1 es de 92,82%, indicando después del proceso que incluso, el equilibrio entre precisión y recall mejoró sustancialmente.

| Métrica | Antes | Después | Δ absoluto | Δ relativo |
|-------------------|--------|---------|-------------------|-------------------|
| Precision (macro) | 0.9059 | 0.9314 | +0.0255 | +2.81% |
| Recall (macro) | 0.8783 | 0.9260 | +0.0477 | +5.43% |
| F1 (macro) | 0.8881 | 0.9282 | +0.0401 | +4.52% |

Imagen 8 – Tabla comparativa de métricas (Overall)

Concluyendo, en la anterior imagen se puede apreciar un resumen de las métricas obtenidas en el proceso, en donde evidentemente se presentaron mejoras en el “después”. Cabe destacar que a pesar de que había métricas apropiadas dada la tarea, al ser un conjunto un tanto desbalanceado, existía sesgo, en donde la clase mayoritaria dominaba en la tarea de aprendizaje.

Sobre las mejoras relativas, es posible decir que el incremento del recall fue el más alto, lo que sugiere que, al balancearse las clases, el modelo detecta más ejemplos reales que antes. Adicionalmente, aunque la precisión tuvo una mejora moderada, su aumento es bastante positivo, ya que, al crecer simultáneamente con el recall, se puede intuir un modelo más estable y robusto. Finalmente, la mejora del F1, evidencia que el modelo predice mejor, y además está en la capacidad de mantener un buen desempeño en todas las instancias.

2. Sección 2: Proceso de automatización del proceso de preparación de datos, construcción del modelo, persistencia del modelo y acceso por medio de API.

En esta parte dejamos todo automatizado como era requerido. El modelo que entrenamos antes, se guardó en .joblib y ahora se carga solo cuando se levanta la app, sin volver a entrenar ni correr nada manual. Además, el procesamiento del texto quedó dentro del pipeline, así que cuando llega un texto el propio modelo lo transforma y predice.

Sobre el reentrenamiento, consideramos tres formas. En primer lugar está el enfoque batch, que consiste en entrenar el modelo usando todos los datos disponibles de una sola vez. Si después aparecen datos nuevos, no se actualiza por partes, sino que hay que volver a entrenarlo completo desde el inicio usando el conjunto anterior más lo nuevo (*Aprendizaje Batch Y Online | Interactive Chaos*, s.f.). Es una forma más estable porque el modelo no va aprendiendo por pedazos sino con todo el contexto junto .

Luego, el aprendizaje incremental es cuando el modelo no se entrena todo de una vez sino que se va actualizando con los datos nuevos a medida que llegan. No se borra lo

que ya aprendió, solo ajusta lo que hace falta sin empezar desde cero (Awan, 2023). Esa forma sirve cuando los datos cambian con el tiempo o no se pueden guardar todos juntos.

Otra forma de reentrenar un modelo es usando una ventana de datos recientes que se va moviendo, lo que también llaman *sliding window* o *rolling window training*. En vez de usar todo el histórico, solo se entrena con los datos más nuevos y se descartan los viejos a medida que llegan más registros (Gulati, 2025). Esto sirve cuando la información cambia con el tiempo y no tiene sentido quedarse con datos desactualizados. Una ventaja es que el modelo se mantiene actualizado sin necesidad de cargar o procesar toda la data anterior. La desventaja es que si la ventana es muy corta puede olvidar patrones que todavía importan.

Al final decidimos implementar batch, usando el histórico más los datos nuevos y aplicando balanceo, dejando aparte un set de validación.

De esta forma, logramos dejar el endpoint /predict funcionando para recibir los textos y devolver lo que se pide: el mismo texto que se envió, la clasificación que da el modelo y la probabilidad. Además, si después se quieren meter datos nuevos para actualizar el modelo, se puede hacer por el endpoint de /retrain. Todo eso queda guardado en la carpeta /models para que el modelo se siga usando o se reemplace cuando se reentrene. Con esto, el proceso completo (preparar, usar y actualizar el modelo) se deja conectado por API.

3. Sección 3: Desarrollo de la aplicación y justificación.

3.1. Usuario, rol y conexión con el proceso

El desarrollo está pensado para el UNFPA y las entidades que trabajan con seguimiento de los Objetivos de Desarrollo Sostenible, especialmente los ODS 1, 3 y 4. En específico el usuario principal son los equipos técnicos que analizan información ciudadana vinculada a los ODS.

Ellos reciben información en formato de texto desde procesos de participación ciudadana, diagnósticos territoriales, consultas comunitarias o reportes institucionales, y necesitan clasificarla según los ODS sin tener que hacerlo manualmente.

Hoy esa tarea depende de personas expertas, consume tiempo, es poco eficiente y no escala cuando se trabaja con muchos textos. La aplicación sirve para automatizar esa parte, ya que toma opiniones o descripciones en lenguaje natural y las clasifica directamente según el modelo entrenado. Esto permite usar los datos más rápido para evaluación, planeación o seguimiento de políticas públicas alineadas con la Agenda 2030. También ayuda a estandarizar los criterios de clasificación y evita depender siempre de analistas especializados.

3.2 Desarrollo de la aplicación web

La aplicación funciona como una interfaz web que consume los dos endpoints hechos anteriormente: /predict para clasificar textos y /retrain para actualizar el modelo.

Desde la interfaz se pueden ingresar textos manualmente o cargar un archivo CSV. La respuesta muestra la clase ODS asignada y su probabilidad. Para usuarios técnicos también está disponible la opción de reentrenar el modelo con nuevos datos etiquetados, desde la misma interfaz. En la aplicación el backend como el frontend se desplegaron dentro de un contenedor Docker, lo que permite ejecutar todo junto.

Para probarlo sólo hay que levantar el contenedor con *docker compose up --build* y luego entrar a <http://localhost:8501> en el navegador. Ahí se encontrarán las opciones para predecir o reentrenar el modelo.

3.3

Recursos informáticos

El entrenamiento y reentrenamiento del modelo se hace en Python (scikit-learn), consideramos que con 4 a 8 GB de RAM y un CPU estándar es suficiente para entrenar, ejecutar y guardar el modelo. La aplicación corre en Docker, donde están juntos el backend Flask y el frontend web, y se expone por el puerto 8501 para acceder desde el navegador.

Integración o disponibilidad para el usuario

La aplicación está pensada para usarse dentro de procesos donde se analizan opiniones o textos vinculados a los ODS. Se puede acceder desde cualquier navegador y no requiere instalación local. Si una organización ya usa plataformas de participación o formularios, la API se puede integrar para procesar automáticamente los textos que se recojan ahí.

Riesgos de uso

En cuanto a los riesgos, lo principal es que el modelo puede equivocarse y alguien podría tomar decisiones basadas en una predicción incorrecta. También existe la posibilidad de que se carguen textos con información sensible sin ningún control, y si el modelo no se reentrena con nuevos datos, puede perder vigencia o empezar a clasificar mal con el tiempo. Desde el lado técnico, siempre está el riesgo de que el servicio se caiga.

4. Sección 4: Resultados.

Link al video: https://github.com/BI-202502-LosPowerBI/Proyecto1-Grupo28/blob/main/resultados/video_etapa2.mp4

5. Sección 5: Trabajo en equipo.

Rol de cada integrante:

Juan Diego fue el líder de proyecto. Se encargó de coordinar las reuniones, organizar la distribución de las tareas y revisar que todo se integrara bien. Además, trabajó en el desarrollo del segundo endpoint /retrain, junto con la interfaz de la app y los resultados.

Santiago asumió el rol de ingeniero de software. Desarrolló la aplicación web e implementó el primer endpoint /predict. También trabajo en la documentación y revisó cómo se iba a consumir el modelo desde la interfaz y el despliegue en docker.

Juan Pablo trabajó como ingeniero de datos. Se encargó de la parte de reentrenamiento. Preparó los datos necesarios y estructuró la lógica para actualizar el modelo desde el backend.

Tiempo requerido para el desarrollo de la entrega:

Juan Diego: 3 horas

Santiago: 3 horas

Juan Pablo: 3 horas

Distribución de puntos:

Juan Diego: 33.3 puntos

Santiago: 33.3 puntos

Juan Pablo: 33.3 puntos

Uso de IA

En esta parte del proyecto se usó ChatGPT como apoyo cuando no se conocía la forma exacta de implementar algo o cuando aparecían errores en el código. Sirvió especialmente para crear la base del frontend en Streamlit y para corregir detalles del docker-compose y el despliegue. Además, se usó para resolver dudas puntuales o de redacción del documento.

Reuniones de grupo – Etapa 2

Reunión de lanzamiento y planeación. Cerramos la forma de trabajo y los roles para esta etapa: Juan Diego como líder del proyecto y resultados, Santiago como ingeniero de software (front y endpoint /predict), y Juan Pablo como ingeniero de datos (reentrenamiento y Sección 1). Definimos el alcance: dejar el pipeline automatizado dentro del modelo, exponer la API en Flask, montar el front en Streamlit y desplegar todo con Docker Compose.

Reuniones de seguimiento. Hicimos seguimientos cortos para validar la integración front-backend y las pruebas con Postman y archivos CSV. Ajustamos el docker-compose (puertos, depends_on, healthcheck), confirmamos la carga del modelo .joblib y probamos /predict desde la interfaz. Cuando apareció el tema del healthcheck, lo resolvimos sin

curl. En paralelo revisamos las métricas post-reentrenamiento y fuimos cerrando la redacción (automatización, desarrollo de la app y riesgos).

Reunión de finalización. Verificamos los endpoints desde la web, reconstruimos la imagen, grabamos el video (flujo de predicción y reentrenamiento) y cerramos el documento. Dejamos un checklist simple de despliegue (docker compose up --build, front en <http://localhost:8501> y API en <http://localhost:8000/ping>)

6. Referencias

Aprendizaje batch y online | *Interactive Chaos*. (s.f.).

<https://interactivechaos.com/es/manual/tutorial-de-machine-learning/aprendizaje-batch-y-online>

Awan, A. A. (2023, Julio 29). *¿Qué es el aprendizaje incremental?* Datacamp.

<https://www.datacamp.com/blog/what-is-incremental-learning>

Gulati, J. (2025, Abril 17). *Detecting & Handling Data Drift in Production*. Machine Learning Mastery. <https://machinelearningmastery.com/detecting-handling-data-drift-in-production/>