**Informe Proyecto 1 Etapa 2**

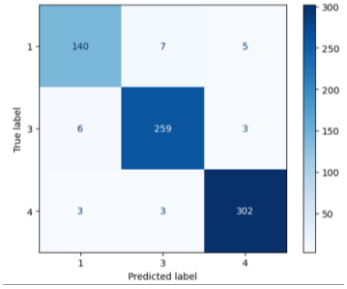
**Integrantes:** Sofia Vásquez - 202123910, Isabella Caputi – 202122075, Mario Velásquez - 202020502

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

### 1.1 Prueba Rendimiento modelo nuevos datos

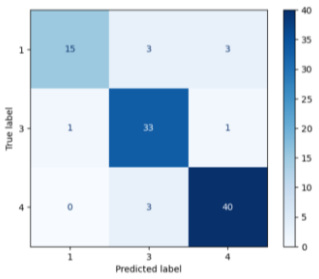
En la Etapa 1, el modelo de Random Forest entrenado mostró un rendimiento sobresaliente sobre el conjunto de datos original. En comparación a los otros dos modelos utilizados, este se escogio como modelo final para la tarea especificada. La exactitud alcanzó un 0.96, lo que significa que la gran mayoría de las predicciones coincidieron con las etiquetas reales. De manera complementaria, el F1-macro también fue de 0.96, lo que refleja un equilibrio sólido entre precisión y recall en todas las clases, garantizando un comportamiento uniforme del modelo. Al analizar métrica por métrica, la precisión indicó que las predicciones positivas en su mayoría fueron correctas. Por otro lado, el recall evidenció la capacidad del modelo para recuperar casi todos los ejemplos verdaderos de cada clase. Por último, el F1-score integra estas dos medidas, mostrando un balance óptimo en el desempeño.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.94 | 0.92 | 0.93 | 152 |
| 3 | 0.96 | 0.97 | 0.96 | 268 |
| 4 | 0.97 | 0.98 | 0.98 | 308 |
| accuracy | | | 0.96 | 728 |
| macro avg | 0.96 | 0.96 | 0.96 | 728 |
| weighted avg | 0.96 | 0.96 | 0.96 | 728 |




En la validación con el nuevo conjunto de datos proporcionado para la Etapa 2 del proyecto (Datos_etapa 2), las métricas se mantuvieron en un nivel positivo, aunque con una ligera reducción, alcanzando una exactitud del 0.89 y un F1-macro de 0.87. Esta diferencia es esperada al enfrentarse a un conjunto externo, ya que el modelo encuentra expresiones, vocabulario y estructuras distintas que no formaban parte del entrenamiento. Al analizar las métricas, la precisión se mantuvo alta en todas las clases, lo que significa que los aciertos del modelo siguen siendo confiables. El recall, aunque mostró una ligera disminución en la clase minoritaria (ODS 1), evidenció que el modelo aún logra identificar la mayoría de los casos relevantes. El F1-score, como métrica combinada, confirma que el balance entre precisión y recall sigue siendo adecuado.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.94 | 0.71 | 0.81 | 21 |
| 3 | 0.85 | 0.94 | 0.89 | 35 |
| 4 | 0.91 | 0.93 | 0.92 | 43 |
| accuracy | | | 0.89 | 99 |
| macro avg | 0.90 | 0.86 | 0.87 | 99 |
| weighted avg | 0.89 | 0.89 | 0.89 | 99 |

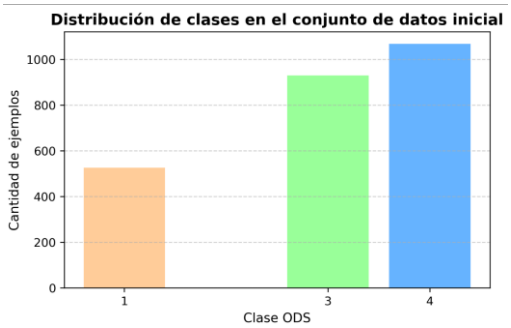



En conjunto, estos resultados permiten afirmar que el modelo no solo funciona correctamente en datos conocidos, sino que también mantiene un desempeño robusto en datos externos. La ligera

reducción en las métricas es natural y no representa una debilidad significativa, sino la oportunidad de fortalecer el modelo mediante técnicas de reentrenamiento como se realizará a continuación, con el fin de consolidar su capacidad de generalización y garantizar resultados consistentes en diferentes contextos de aplicación.

### 1.2 Aumentación de datos y Reentrenamiento del modelo

En la siguiente figura, se observa la distribución de clases en el conjunto de datos original (Datos Proyecto), correspondiente a los Objetivos de Desarrollo Sostenible (ODS) 1, 3 y 4. Se observa que la clase ODS 1 (Fin de la pobreza) cuenta con aproximadamente 500 ejemplos, mientras que las clases ODS 3 (Salud y bienestar) y ODS 4 (Educación de calidad) presentan entre 900 y 1000 ejemplos respectivamente.

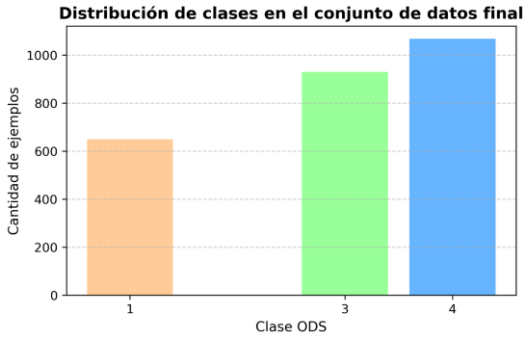



Esta diferencia evidencia un claro desbalance de clases, donde ODS 1 se encuentra subrepresentada. Este desequilibrio afecta el desempeño del modelo, pues tiende a aprender con mayor precisión las clases mayoritarias y a presentar un menor recall en la clase minoritaria, lo que implica que muchos textos realmente asociados a ODS 1 no son correctamente identificados. Para abordar el desbalance existente en el conjunto de entrenamiento, particularmente en la clase correspondiente al ODS 1 (Fin de la pobreza), se implementó una estrategia de aumentación de datos mediante *prompting* semántico. Para esto, se generaron ejemplos sintéticos adicionales que contribuyan a equilibrar el conjunto de entrenamiento y a mejorar la capacidad del modelo para reconocer patrones lingüísticos propios de esta categoría, favoreciendo así un rendimiento más equitativo y robusto entre todas las clases.

El procedimiento de *prompting* implementado consistió en la generación de nuevos textos sintéticos mediante el uso de la API de OpenAI, con el objetivo de aumentar la representación de la clase minoritaria correspondiente al ODS 1. A partir del conjunto original de entrenamiento, se identificaron los textos pertenecientes a dicha categoría y se seleccionó una muestra de ejemplos reales para servir como contexto en la construcción del *prompt*. Este *prompt* fue obtenido de la guía proporcionada para el proyecto, el cual cuenta con instrucciones específicas para solicitar la creación de opiniones ciudadanas breves, realistas y coherentes. Se utilizo para generar opiniones con problemáticas locales asociadas exclusivamente al ODS 1. El modelo de lenguaje GPT-4o-mini fue utilizado para generar aproximadamente 200 nuevos textos en formato JSON, los cuales incluían tanto el campo de texto como su etiqueta correspondiente. Sin embargo, realmente esta cantidad fue un estimado utilizado, ya que cuando se intentó generar una cantidad alta de datos (200) para mejorar el modelo, las métricas obtenidas no fueron tan buenas ya que se presentaba sobreajuste a la clase ODS 1. Por lo tanto, se tomó la decisión de adicionar entre 100-150 datos

artificiales, con el fin de que el modelo no obtuviera estos sesgos. Para terminar, una vez generados, los datos fueron sometidos a un proceso automatizado de limpieza y validación para eliminar duplicados, inconsistencias y fragmentos no pertinentes, asegurando así su calidad antes de ser integrados al conjunto de entrenamiento ampliado (Datos proyecto y Datos Etapa 2).

Posteriormente, este nuevo conjunto equilibrado fue empleado para reentrenar el modelo de clasificación basado en un Random Forest dentro del pipeline utilizado para el modelo original, el cual incluía el preprocesamiento de texto. Este proceso permitió mejorar la diversidad léxica y semántica del corpus, fortaleciendo la capacidad del modelo para identificar correctamente las opiniones relacionadas con el ODS 1 y reduciendo el sesgo hacia las clases mayoritarias (ODS 3 y ODS 4). En consecuencia, se logró un mejor equilibrio en las métricas de desempeño, particularmente en el recall y el F1-Score de la clase minoritaria. A continuación, se observan los resultados de esta estrategia implementada para una de las corridas hechas.




En primer lugar, observamos la nueva distribución de clases, que como se puede ver en la imagen anterior, ha incrementado significativamente la cantidad de datos para la clase ODS 1. En esta imagen, se observa que la clase ODS 1 paso a tener 650 datos, cuando antes tenía aproximadamente 500 datos. Por lo tanto, efectivamente se generaron suficientes datos para ajustar parcialmente la inequidad de la distribución de datos entre clases.

Por otro lado, a partir de los resultados presentados en las siguientes tablas, se observa una mejora significativa en el desempeño del modelo tras la aplicación de la estrategia de aumentación de datos mediante *prompting*. El modelo original alcanzó una exactitud de 0.899, una precisión macro promedio de 0.906, un recall de 0.870 y un F1-score de 0.882. En contraste, el modelo reentrenado, que incorporó los textos sintéticos generados, logró una exactitud de 0.961, una precisión macro de 0.962, un recall de 0.954 y un F1-score de 0.957.

| Métrica | Modelo original | Modelo reentrenado |
|---|---|---|
| Accuracy | 0.899 | 0.961 |
| Precision (macro) | 0.906 | 0.962 |
| Recall (macro) | 0.87 | 0.954 |
| F1-score (macro) | 0.882 | 0.957 |

Estas mejoras reflejan un incremento notable en la capacidad del modelo para generalizar y reconocer con mayor consistencia los patrones de las diferentes clases asociadas a los ODS. En particular, el avance más relevante se evidencia en la clase minoritaria (ODS 1), donde el F1-score aumentó de 0.811 a 0.936 y el recall paso de 0.714 a 0.903. Esto sugiere que hubo una reducción

sustancial de los errores de clasificación del modelo y una mejor representación de esta categoría en el conjunto de datos ampliado.

| Clase | Precisión original | Precisión reentrenad | Recall original | Recall reentrenado | F1 original | F1 reentrenado |
|---|---|---|---|---|---|---|
| 1.0 | 0.938 | 0.972 | 0.714 | 0.903 | 0.811 | 0.936 |
| 3.0 | 0.868 | 0.934 | 0.943 | 0.971 | 0.904 | 0.953 |
| 4.0 | 0.911 | 0.978 | 0.953 | 0.988 | 0.932 | 0.983 |

En términos generales, el modelo reentrenado demuestra un comportamiento más equilibrado y robusto, alcanzando una mejora homogénea en todas las métricas. La estrategia de aumentación mediante *prompting* contribuyó a diversificar el vocabulario y las estructuras sintácticas de los textos, lo cual permitió al modelo capturar de manera más efectiva las relaciones semánticas subyacentes. En consecuencia, el sistema final no solo exhibe un rendimiento cuantitativamente superior, sino también una mayor estabilidad en la clasificación de opiniones ciudadanas vinculadas con los ODS 1, 3 y 4.

**Sección 2. (15%)**

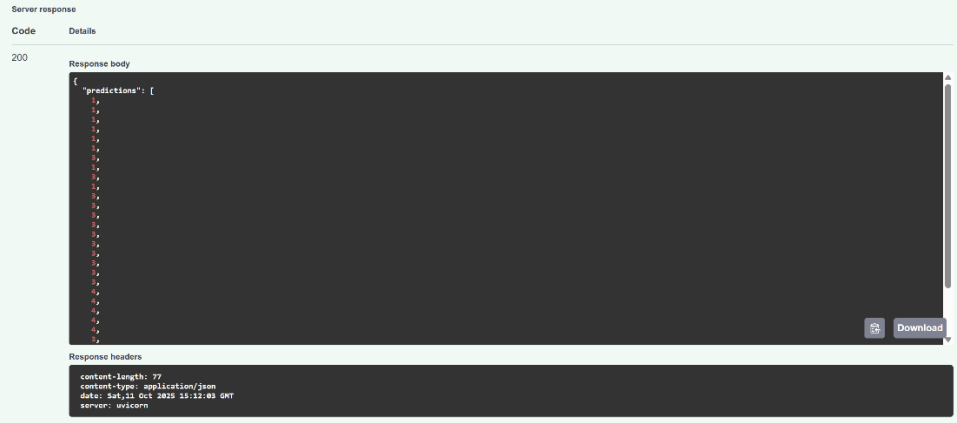
Proceso de automatización del proceso de preparación de datos, construcción del modelo, persistencia del modelo y acceso por medio de API: Descripción del proceso e implementación realizados por el ingeniero de datos, acompañado del código. Debe utilizar pipelines para este proceso y un Framework para desarrollo de la API. La API REST debe estar compuesta por dos “endpoints”:

- El primero debe recibir mediante el "body" una o más instancias de datos con la totalidad de las características requeridas para las que se desea realizar la predicción. Este “endpoint” debe devolver una lista con la misma cardinalidad de las instancias recibidas, y en el mismo orden, que contenga las predicciones realizadas por el modelo para cada instancia de datos recibida.

El proceso de automatización se implementó mediante **FastAPI** para exponer el modelo de clasificación de textos ODS como un servicio REST compuesto por dos endpoints:

1. **/predict**: recibe, mediante el cuerpo de la solicitud (body), una lista de instancias con la característica *“textos”*. El endpoint aplica el pipeline completo de preprocesamiento (tokenización, eliminación de stopwords, normalización y vectorización TF-IDF) y devuelve una lista de predicciones con la misma cardinalidad y orden que los textos recibidos.

Realizamos una prueba con los datos que estan en la carpeta Etapa 2\Pruebas\predic.json y nos dio el siguiente resultado:




Aca podemos ver como el modelo puede predecir correctamente a los ODS que pertencen los textos propuestos.

- El segundo “endpoint”, debe estar en capacidad de enviar una cantidad relevante de instancias de datos que servirán para realizar un proceso de re-entrenamiento del modelo, razón por la cual, además de enviar las características, también será requisito enviar la variable objetivo. Como respuesta, el “endpoint” debe devolver algunas métricas de desempeño como pueden ser el Precision, Recall, F1-score. Además, tenga presente que con el proceso de re-entrenamiento del modelo, el archivo binario deberá reemplazarse para ser tenido en cuenta como nueva versión la próxima vez que se solicite realizar una predicción mediante el primer “endpoint”. En este punto debe investigar y plantear tres definiciones distintas de re-entrenamiento. En cada una de ellas debe incluir una descripción del significado, acompañada de una ventaja o desventaja de esa propuesta. Al final debe concluir indicando la que implementó en esta etapa del proyecto. Los datos tanto de la solicitud como de la respuesta deben estar en formato JSON y deben respetar el esquema del CSV original proporcionado para este proyecto 1. Recuerde que el pipeline es el encargado de hacer todas las preparaciones requeridas previo a hacer los procesos de entrenamiento o predicción.

El segundo endpoint que implementamos:

2. **/retrain**: permite el reentrenamiento completo del modelo (Batch Full Refit). En este endpoint se reciben nuevas instancias en formato JSON con las variables *“textos”* y *“labels”*. Antes del entrenamiento, la API combina automáticamente estos datos con los históricos contenidos en los archivos Datos_proyecto.xlsx y Datos_etapa 2.xlsx. Posteriormente se eliminan duplicados, se realiza una división estratificada de entrenamiento-validación (70 / 30) y se reconstruye el pipeline para ajustar el modelo de clasificación.

Una vez finalizado el proceso, el nuevo modelo se persiste en el archivo pipeline_model.joblib, que reemplaza a la versión anterior, garantizando que las futuras predicciones de /predict usen el modelo más actualizado.

En el desarrollo del proyecto se investigaron y compararon tres enfoques distintos de reentrenamiento de modelos de aprendizaje automático, cada uno con características, ventajas y limitaciones particulares.

El primero es el reentrenamiento completo o *Batch Full Refit*, que consiste en volver a entrenar el modelo desde cero utilizando la totalidad de los datos disponibles (históricos y nuevos). Esta estrategia garantiza consistencia global y la incorporación íntegra de toda la información, aunque su principal desventaja es el mayor costo computacional y el tiempo de procesamiento requerido.

El segundo enfoque es el reentrenamiento incremental o *Online Learning*, en el que el modelo se actualiza progresivamente conforme llegan nuevos lotes de datos, ajustando sus parámetros sin necesidad de reentrenar desde el inicio. Su principal ventaja es la eficiencia en entornos donde los datos se generan de manera continua, aunque requiere modelos compatibles con aprendizaje parcial y un control cuidadoso de las tasas de actualización.

Finalmente, se analizó la estrategia de ventana deslizante o *Rolling Window*, que mantiene el modelo actualizado reentrenándolo periódicamente solo con los datos más recientes y descartando los antiguos, lo que permite adaptarse a cambios en los patrones de los datos (concept drift), pero puede ocasionar pérdida de conocimiento histórico si la ventana es demasiado corta.

En el caso de nuestro contexto, Se eligió el enfoque **Batch Full Refit** porque permite reconstruir el modelo de manera íntegra en cada ciclo de reentrenamiento, garantizando que las predicciones se basen en la totalidad de los datos disponibles (históricos y nuevos) y asegurando así mayor coherencia, estabilidad y control del desempeño del sistema.

Realizamos la prueba con el archivo que se encuetra en la carpeta Etapa 2\Pruebas\retrain.json


En las pruebas finales, el reentrenamiento alcanzó un desempeño de validación macro promedio de Precision = 0.949, Recall = 0.942 y F1-Score = 0.945 sobre 766 instancias de validación, lo que evidencia la correcta integración del pipeline, la estabilidad del modelo y su capacidad de generalización frente a nuevos textos.

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

### 3.1 Usuario y conexión con el proceso de negocio.

El principal usuario de la aplicación es el analista de programas del UNFPA (rol de monitoreo y evaluación). Su responsabilidad es priorizar y hacer seguimiento a intervenciones alineadas con los ODS 1, 3 y 4 a partir de opiniones ciudadanas.

La aplicación se integra al proceso de negocio en tres puntos:

1. Ingesta y análisis: el analista carga opiniones (individuales o masivas) y obtiene la etiqueta ODS y su probabilidad por clase. Esto sirve como insumo para informes del analista de UNFPA.
2. Toma de decisiones: con las salidas agregadas de las opiniones ciudadanas es más fácil identificar cuales son las opiniones ciudadanas tras haber clasificado los textos dependiendo de a qué ODS se refieren. Esto para llevar estos insumos a paneles de expertos en políticas públicas que luego toman dichos textos (ya clasificados) como insumos en las políticas que diseñen.
3. Mejora continua: cuando el analista valida casos fronterizos, puede anotar ejemplos y enviarlos a reentrenamiento, cerrando el ciclo de aprendizaje del sistema y reduciendo el *drift* del modelo.

La disponibilidad de esta aplicación es crítica para el rol de analista de UNFPA porque reduce cargas de lectura manual, aporta trazabilidad cuantitativa (probabilidades y métricas) y automatiza los procesos de clasificación. Por su parte, la disponibilidad de la aplicación para el rol de experto en ML es crucial cuando se requiera hacer reéntrenos del modelo, por ejemplo, si llegan un número elevado de nuevos textos de un grupo de ciudadanos con un léxico poco conocido por el modelo.

### 3.2 Desarrollo de la aplicación e interacción.

Arquitectura de la aplicación:

- Backend: API en FastAPI/Uvicorn, con un *pipeline* de scikit-learn (vectorización TF-IDF y clasificador con predict_proba).
  - POST */predict*: recibe *instances[{textos}],* devuelve *predictions* y probabilities por ODS.
  - POST */retrain*: recibe *instances[{textos, labels}],* combina históricos y nuevos ejemplos, reentrena, persiste el modelo y devuelve métricas (*precision, recall* y F1 macro) y metadatos de versión.
  - Limpieza previa (HTML, normalización de etiquetas) y validación de tamaño/cantidad de textos.
- Persistencia: el modelo se guarda como binario (pipeline_model.joblib) en un volumen /data, con archivo de metadatos (timestamp/ruta) para auditoría y *rollback*.
- Frontend: interfaz web en Streamlit con dos pestañas:
  - Usuario: clasificación de un único texto o de un CSV. Se dispone visualización del ODS con su nombre completo y una gráfica de probabilidades con título, ejes y etiquetas legibles. Cuando se usa la opción de CSV se permite la exportación de los resultados.
  - Experto (retrain): carga de CSV/JSON etiquetado para reentrenar, visualización de métricas en tarjetas, tabla y gráfico, y se presenta la metadata de la versión del modelo activo.

Semántica de reentrenamiento implementada:

Se adoptó un esquema de reentrenamiento por reemplazo de versión: el pipeline se entrena desde cero con históricos + nuevos ejemplos validados, se evalúa con un *split* estratificado 70/30 y, si

pasa umbrales internos, se publica como nueva versión (persistencia en /data y actualización de metadatos). Esta estrategia facilita auditoría, comparación entre versiones y reversión.

### 3.3 Recursos informáticos y despliegue.

Para el despliegue de la aplicación web se requieren los siguientes recursos

- Entrenamiento/ejecución:
  - CPU: 1–2 vCPU son suficientes.
  - Memoria: 1–2 GB RAM (hasta 4 GB si se reentrena con lotes de textos nuevos grandes).
  - Almacenamiento: <1 GB para binario del modelo, NLTK y logs. Acá se persiste el modelo.
  - Tiempo: reentrenamientos en segundos o minutos con datasets de tamaño moderado.
- Despliegue:
  - API en Docker (variables: MODEL_DIR=/data, MODEL_FILENAME) con volumen para persistencia y logs a stdout.
  - Front con Streamlit con API_BASE apuntando al dominio público de la API.

Integración organizacional:

- Ingreso de datos desde formularios/encuestas o CSV exportados de herramientas corporativas.
- Operación: clasificación periodica (semanal o mensual) de los nuevos lotes de texto: revisión de *outliers* y etiquetado humano de casos límite. Asimismo, hacer reentrenamiento con aprobación del responsable de ML. Despues del reentrenamiento se hace la publicación de versión y se comunica al equipo sobre la nueva versión.
- Consumo: los resultados (etiqueta + probabilidad) se integran a tableros y reportes de gestión programática de la UNFPA.

### 3.4 Riesgos para el usuario final y mitigaciones.

1. Riesgos analíticos (sesgo/desbalance, *drift*, sobreajuste).
   - *Mitigación*: usar métricas macro por clase, hacer monitoreo periódico, definir umbrales mínimos para publicación, hacer el reentrenamiento con datos curados, realizar comparación entre versiones y tener en cuenta la capacidad de *rollback*.
2. Riesgos de interpretación (sobreconfianza en una sola salida).
   - *Mitigación*: mostrar probabilidades y tenerlas en cuenta, advertir casos de baja confianza, fomentar revisión humana en decisiones sensibles.
3. Privacidad y seguridad (textos con PII, exposición de endpoints).
   - *Mitigación*: anonimización previa, HTTPS, autenticación/autorización en retrain y hacer trazabilidad con los logs.
4. Disponibilidad y continuidad (pérdida del modelo o caída del servicio).
   - *Mitigación*: persistencia en volumen /data, hacer *backups* automáticos, *health checks* periódicos y hacer múltiples réplicas (instancias de los servidores) según demanda del servicio ofrecido a la UNFPA.
5. Dependencias técnicas (recursos NLTK, versiones de librerías).

- *Mitigación*: *pinning* de versiones en requirements.txt, precarga de recursos en la imagen Docker y pruebas automatizadas.

**Sección 4. (18%) Resultados.**

Video de máximo 5 minutos con la descripción y visualización en la aplicación de los resultados del modelo analítico, que permita a un rol dentro de la organización comprenderlos y usarlos. El video debe simular la interacción del usuario final con la aplicación y describir dos acciones que puede realizar como resultado de dicha interacción, haciendo énfasis en la forma como el resultado del modelo aporta en esas acciones y reflexionando sobre el posible impacto a nivel Colombia el uso de la aplicación desarrollada. Realice pruebas de facilidad de uso de la aplicación y utilidad para el usuario para el cual se diseñó, puede vincular a otros estudiantes del curso o personas cercanas para esta validación.

**Sección 5. (10%) Trabajo en equipo.**

**ROLES Y TAREAS:**

**Mario Velásquez** – Líder de Proyecto e Ingeniero de Software Responsable del Desarrollo de la Aplicación (10 horas)
Rol: Como líder de proyecto, asumió la coordinación general del equipo, definiendo los cronogramas, organizando las reuniones, asignando responsabilidades y garantizando la entrega oportuna de los entregables. Además, fue responsable del desarrollo técnico de la aplicación, asegurando la correcta integración con el modelo analítico.
Tareas realizadas: Diseño y desarrollo de la aplicación web para la interacción del usuario con el modelo analítico; implementación de funcionalidades para la predicción y reentrenamiento del modelo mediante los endpoints definidos en la API REST; documentación técnica del despliegue de la aplicación.
Retos enfrentados: Integrar correctamente la aplicación con la API y el modelo reentrenado, garantizando la estabilidad del sistema y la correcta visualización de los resultados.
Uso de ChatGPT: Apoyo en la definición de la arquitectura de la aplicación, la configuración de los endpoints en el frontend y la depuración de errores durante el proceso de desarrollo.

**Sofía Vásquez** – Ingeniera de Software Responsable del Proceso de Automatización (10 horas)
Rol: Encargada del diseño y desarrollo del proceso de automatización completo, que incluyó la preparación de datos, construcción, persistencia y exposición del modelo analítico mediante una API REST.
Tareas realizadas: Implementación de pipelines para la preparación de datos, entrenamiento y reentrenamiento del modelo; desarrollo de los dos endpoints de la API REST —uno para la generación de predicciones y otro para el reentrenamiento—; documentación de las tres definiciones evaluadas de reentrenamiento, con su respectivo análisis de ventajas y desventajas.
Retos enfrentados: Garantizar la correcta ejecución de los pipelines y la comunicación entre el modelo y la API, así como la gestión del archivo binario actualizado tras el reentrenamiento.
Uso de ChatGPT: Apoyo en la estructuración del pipeline, validación del formato JSON en los requests y responses de la API, y revisión conceptual de las definiciones de reentrenamiento.

**Isabella Caputi –** Ingeniera de Datos Responsable del Aumento y Reentrenamiento del Modelo (10 horas)
Rol: Responsable de aplicar técnicas de aumentación de datos y reentrenamiento del modelo, utilizando el nuevo conjunto de datos (Datos_etapa 2) y los ejemplos sintéticos generados mediante técnicas de prompting para mejorar el equilibrio de clases.
Tareas realizadas: Evaluación del rendimiento del modelo anterior con el nuevo conjunto de datos; generación de datos sintéticos para la clase minoritaria; incorporación de estos al conjunto original; reentrenamiento del modelo con el algoritmo que presentó el mejor desempeño en la etapa 1; elaboración de una tabla comparativa de métricas (precisión, recall, F1-score) antes y después del reentrenamiento. Además, colaboró con Sofía en la interpretación de resultados y en la grabación del video de presentación de resultados.
Retos enfrentados: Lograr mejoras significativas en las métricas del modelo tras la aumentación, así como asegurar la coherencia entre los nuevos datos sintéticos y el conjunto de datos original.
Uso de ChatGPT: Asistencia en la generación de ejemplos sintéticos mediante técnicas de prompting, ajuste de hiperparámetros durante el reentrenamiento y comparación de métricas entre los modelos antes y después de la intervención.

**DISTRIBUCIÓN DE PUNTOS (100):**
• Mario Velásquez: 33,33 puntos
• Sofía Vásquez: 33,33 puntos
• Isabella Caputi: 33,33 puntos

**REUNIONES ETAPA 2:**

1. Reunión de lanzamiento y planeación (6 de octubre de 2025): Definición de roles específicos (Isabella: aumentación y reentrenamiento; Sofía: automatización; Mario: desarrollo de aplicación; resultados: Sofía e Isabella) y revisión de los requerimientos técnicos del proyecto.
2. Reunión de ideación (7 de octubre de 2025): Discusión sobre el enfoque metodológico del reentrenamiento, diseño de los pipelines y definición de la arquitectura general de la aplicación.
3. Reuniones de seguimiento: Comunicación continua por medio de WhatsApp para monitorear avances, resolver inconvenientes técnicos y coordinar entregas parciales.
4. Reunión de finalización (12 de octubre de 2025): Validación de resultados finales, revisión de métricas comparativas, verificación del funcionamiento completo de la aplicación y consolidación de los entregables en GitHub.

**PUNTOS POR MEJORAR**:
• Fortalecer la planificación del tiempo para permitir mayor margen de prueba e iteración.
• Incrementar la frecuencia de reuniones sincrónicas para mejorar la comunicación técnica entre las etapas del proyecto.
• Estandarizar la documentación técnica (API, modelo y aplicación) para asegurar la trazabilidad de los procesos y facilitar futuras actualizaciones.