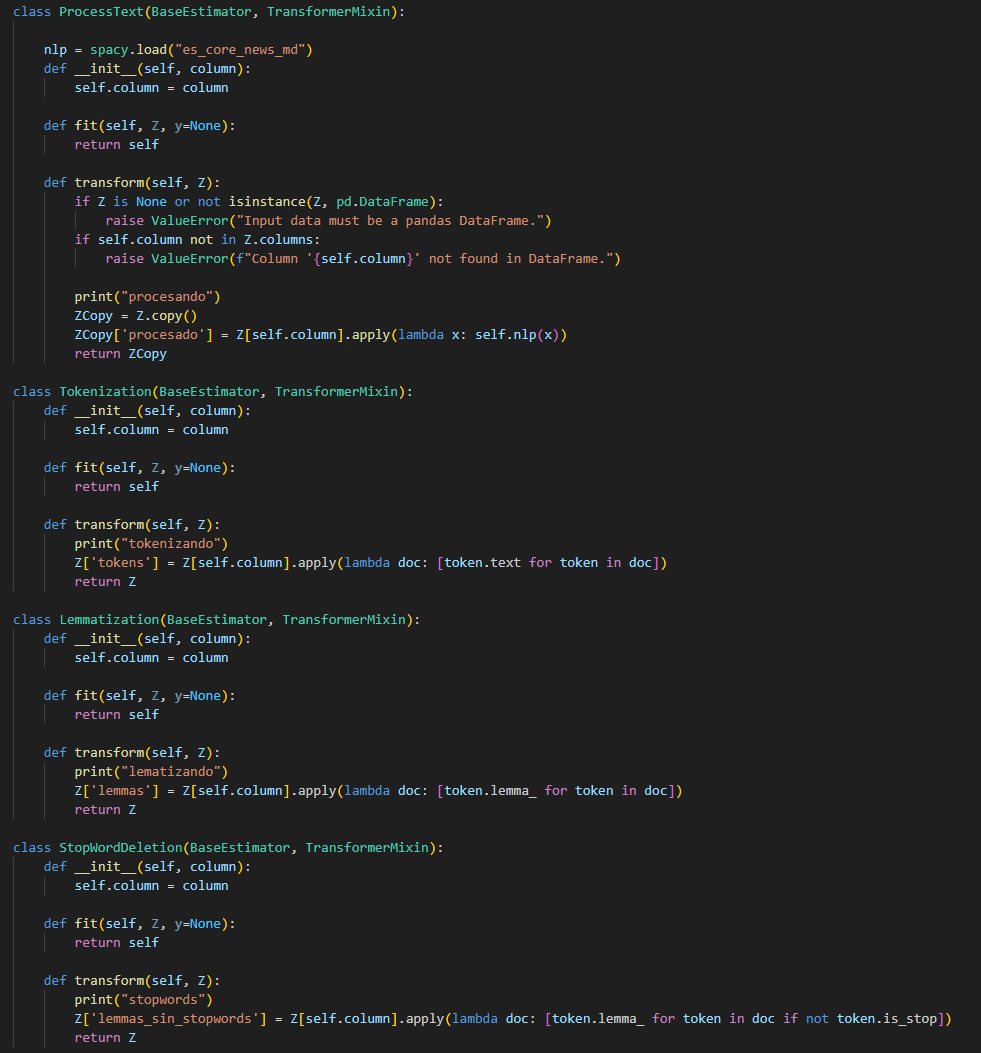
|  |
| --- |
| **Informe Proyecto 1, Entrega 2 Inteligencia de Negocios** |
|  |

|  |
| --- |
| **Maria Fernanda De la Hoz 202214512, Silvana Sandoval Galvis 202123682, Gabriela Soler 202123744**  **12 de octubre de 24** |

**Sección 1. 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.**

Para el proceso de automatización del proceso de preparación de datos, se implementó un pipeline que realiza tareas como la normalización de texto, la eliminación de caracteres especiales, la tokenización y la extracción de características. Dentro del pipeline se usa la librería pandas, que se emplea para el procesamiento y limpieza inicial de los datos, y también se usa NLTK y spaCy para la tokenización y lematización de los textos. De esta manera, el pipeline garantiza que haya un flujo replicable que permite transformar los datos de entrada de manera consistente, lo cual reduce la variabilidad y errores humanos en la preparación de los datos.



Respecto a la construcción del modelo, el pipeline incluye la selección y construcción del modelo analítico que automatiza el proceso de entrenamiento y la selección del modelo. Para esto, se utilizó *scikit-learn* para definir un pipeline que engloba el pre procesamiento de texto, la extracción de características y el entrenamiento del modelo. Además, la persistencia del modelo se logra mediante el uso de *joblib* para guardar el modelo final una vez entrenado. Esto permite almacenar el modelo en formato *pkl*, el cual se puede reutilizar para realizar predicciones en el futuro. La persistencia también tiene en cuenta el almacenamiento de hiperparámetros relevantes y el pipeline completo para que sea posible actualizar o ajustar el modelo sin necesidad de repetir todo el proceso de configuración.

Para el acceso por medio de API, se desarrolló una API REST utilizando fastAPI con dos endpoints principales: /predict/ y /reentrenar-con-archivo/. El primero recibe un archivo de datos que puede ser CSV, Excel o JSON a través de un campo de archivo. Al recibirlo, lo procesa automáticamente y devuelve las predicciones correspondientes. Según el formato del archivo de entrada, las predicciones se devuelven al usuario. Este endpoint facilita la predicción de categorías asociadas a cada opinión, relacionando las opiniones con uno de los ODS 3, 4 o 5. El segundo endpoint, /reentrenar-con-archivo/, permite realizar el reentrenamiento del modelo utilizando nuevos datos etiquetados. Se admite la carga de archivos CSV, Excel o JSON, los cuales se leen y se procesan automáticamente para preparar los datos. Posteriormente, el pipeline procesa el archivo, y el modelo se reentrena utilizando los datos proporcionados. Tras el reentrenamiento, el endpoint devuelve un reporte de clasificación que incluye métricas para saber la calidad del modelo. Además, el archivo del modelo es actualizado para reflejar la nueva versión, lo que permite mejorar su rendimiento de manera continua.





**Sección 2. Desarrollo de la aplicación y justificación. Descripción del usuario/rol de la organización que va a utilizar la aplicación, la conexión entre esa aplicación y el proceso de negocio que va a apoyar (si aplica), y la importancia que tiene para ese rol la existencia de esta aplicación.**

La aplicación desarrollada está diseñada para ser utilizada por analistas de políticas públicas y gestores de programas sociales en el Fondo de Poblaciones de las Naciones Unidas (UNFPA), así como por funcionarios de otras entidades públicas que trabajan en temas relacionados a estos objetivos de desarrollo sostenible específicos y personas interesadas en estos. Estos usuarios tienen el rol de analizar las opiniones de los ciudadanos para identificar problemáticas y proponer soluciones que estén alineadas con los Objetivos de Desarrollo Sostenible 3, 4 y 5. Por esta razón, los roles se benefician en gran medida de la automatización del análisis de datos ya que permite priorizar acciones y enfocar los recursos de manera más eficiente al contar con una herramienta que permite clasificar opiniones de los ciudadanos a los ODS.

La aplicación automatiza el proceso de análisis de opiniones ciudadanas y las categoriza según los ODS, lo cual es un proceso crucial en la formulación de políticas en las distintas organizaciones. De igual manera, le permite a los ciudadanos también expresar de una manera ágil y eficiente sus opiniones frente a las diferentes problemáticas que enfrentan en el país. Con lo anterior, se evidencia que la aplicación logra también conectar y acercar a los ciudadanos y a las entidades públicas, pues hay una comunicación que puede ser más eficiente y cercana entre ambas partes. Anteriormente, el análisis de estas opiniones requería un gran esfuerzo manual, lo cual hacía el proceso ineficiente y costoso. Con la implementación de esta aplicación, los analistas pueden subir los datos directamente y recibir resultados de forma automática, lo que facilita la rápida identificación de áreas problemáticas en temas de salud, educación y equidad de género. Así, las entidades pueden tener una respuesta más oportuna y una mejor toma de decisiones basada en la evidencia generada por el modelo.

Adicionalmente, la existencia de esta aplicación tiene una importancia crítica para el rol del analista pues, reduce significativamente el tiempo y el esfuerzo requerido para procesar grandes volúmenes de opiniones ciudadanas. Gracias a la automatización, el tiempo dedicado al análisis de datos disminuye, lo cual permite a los analistas enfocarse en tareas más estratégicas y de mayor valor, como la búsqueda de soluciones a las problemáticas identificadas. Además, el reentrenamiento continuo del modelo mediante la funcionalidad de la API asegura que el análisis se mantenga relevante y ajustado a las necesidades cambiantes de la sociedad, es decir, se tendrán decisiones más informadas y con mejor adaptación frente a los contextos de los ciudadanos. Finalmente, la confiabilidad y escalabilidad de la aplicación también permiten que la organización responda con datos precisos y actualizados, que es fundamental para cumplir con los objetivos de desarrollo de una manera efectiva y eficiente.

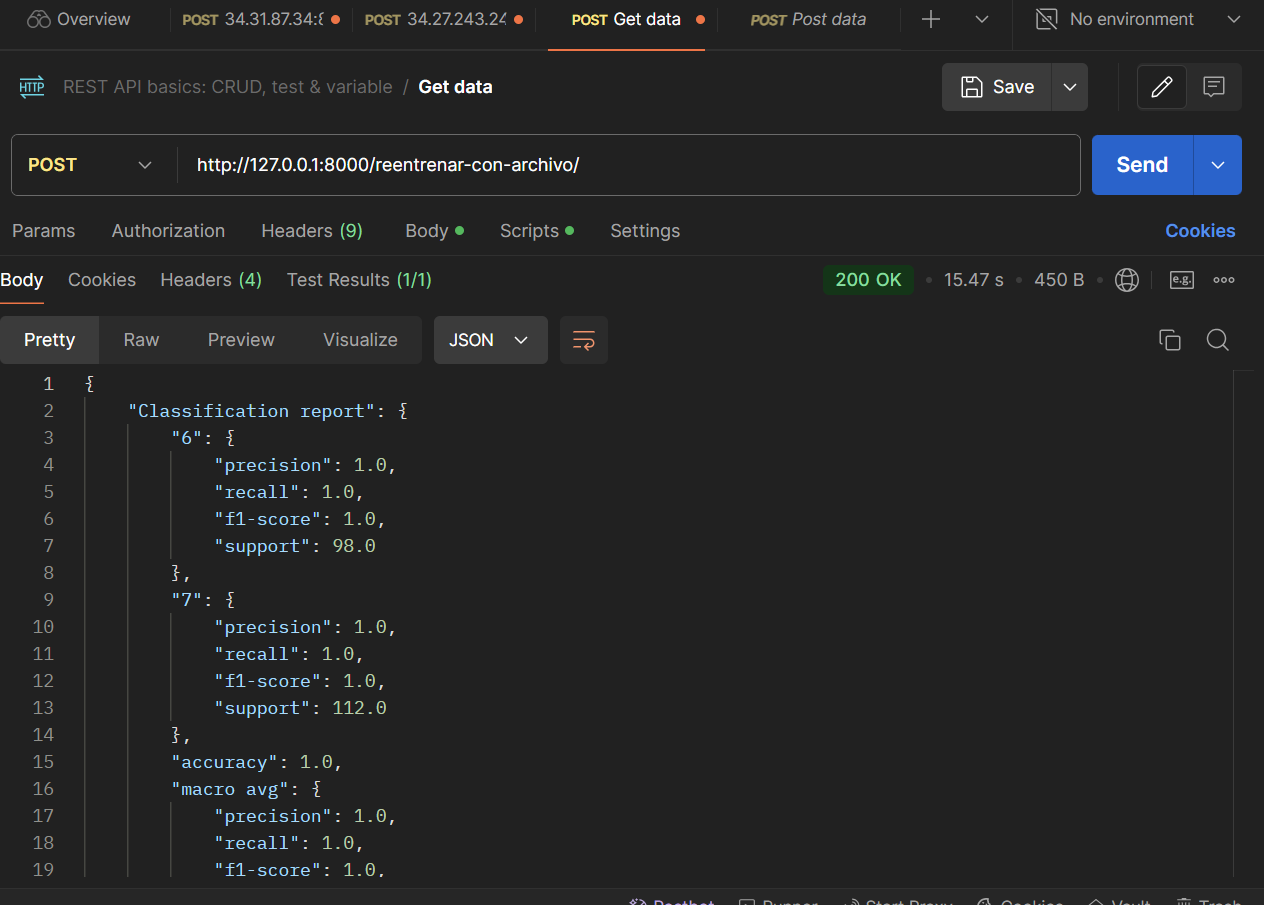
**Sección 3. Trabajo en equipo.**

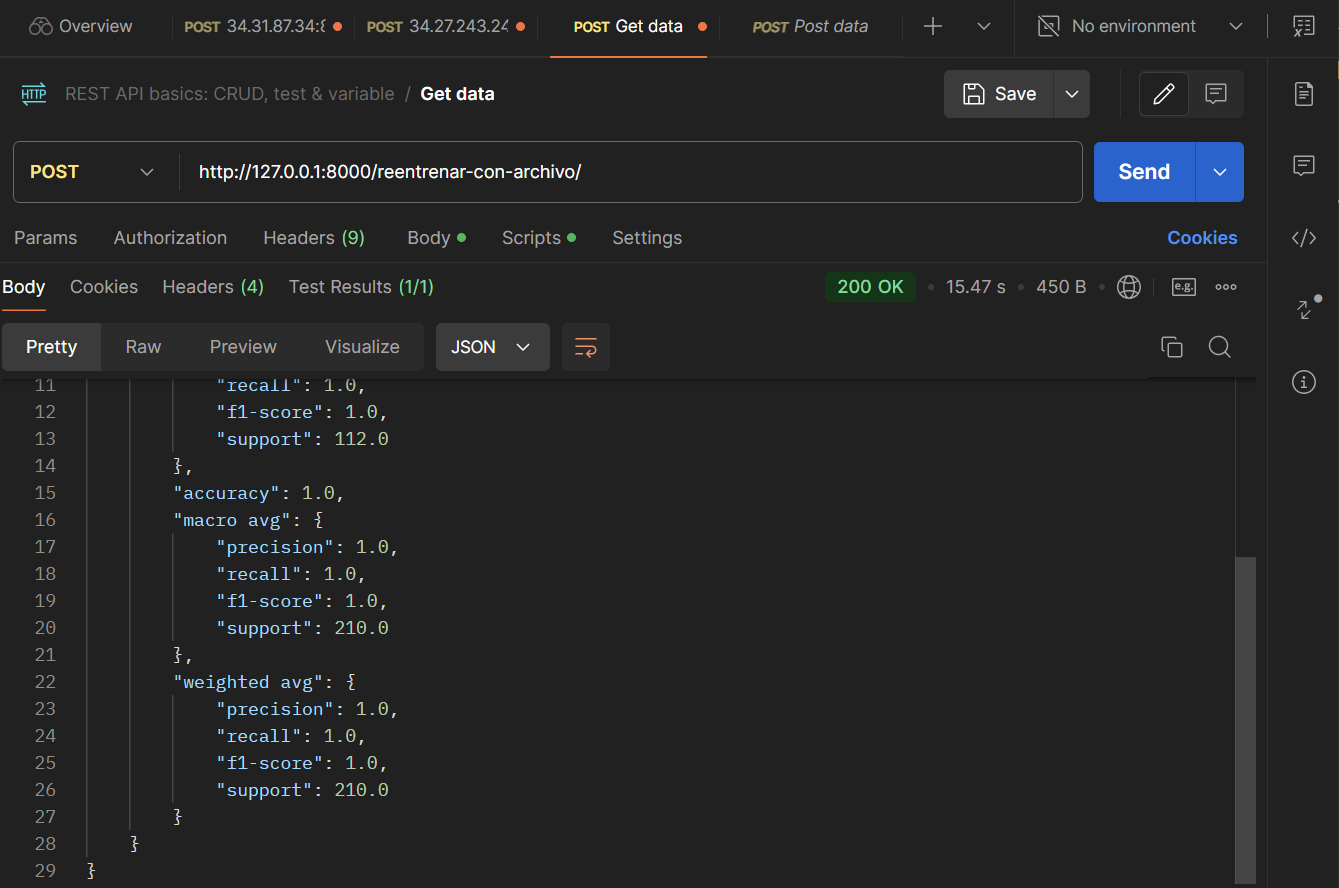
* Maria Fernanda: Ella se encargó de hacer todo el front end y la integración con la API. Rol: Ingeniera de Software. **Número de horas trabajadas:** 8
* Gabriela: Se encargó de hacer el segundo endpoint del API, el documento y el video. Además, hizo parte del back-end. Rol: Líder de grupo. **Número de horas trabajadas:** 8
* Silvana: Ingeniería de Datos. Ella se encargó de hacer todo el pipeline, al igual que implementar el primer endpoint del API y los módulos de este. **Número de horas trabajadas:** 8

Se hicieron varias reuniones para repartir equitativamente el trabajo, tener fechas límites de entrega y poder conectar todas las partes desarrolladas. Además, tuvimos reuniones en las que cada una hacía su parte y se resolvían dudas en conjunto para poder seguir con los objetivos del proyecto.

**Reentrenamiento**

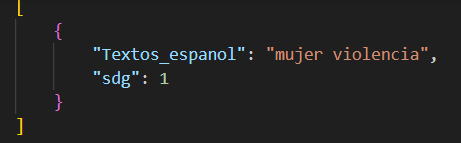
Si bien la API tiene el endpoint de reentrenamiento y funcional (como muestran las tomas de pantalla de Postman, no es recomendable hacerlo desde la aplicación ya que este proceso toma un largo tiempo que lleva a un RunTime error. Adicionalmente, al hacer este reentrenamiento, los datos usados anteriormente no son reconocidos por el modelo y este se daña, por lo cual solo es recomendable hacer esto al final.

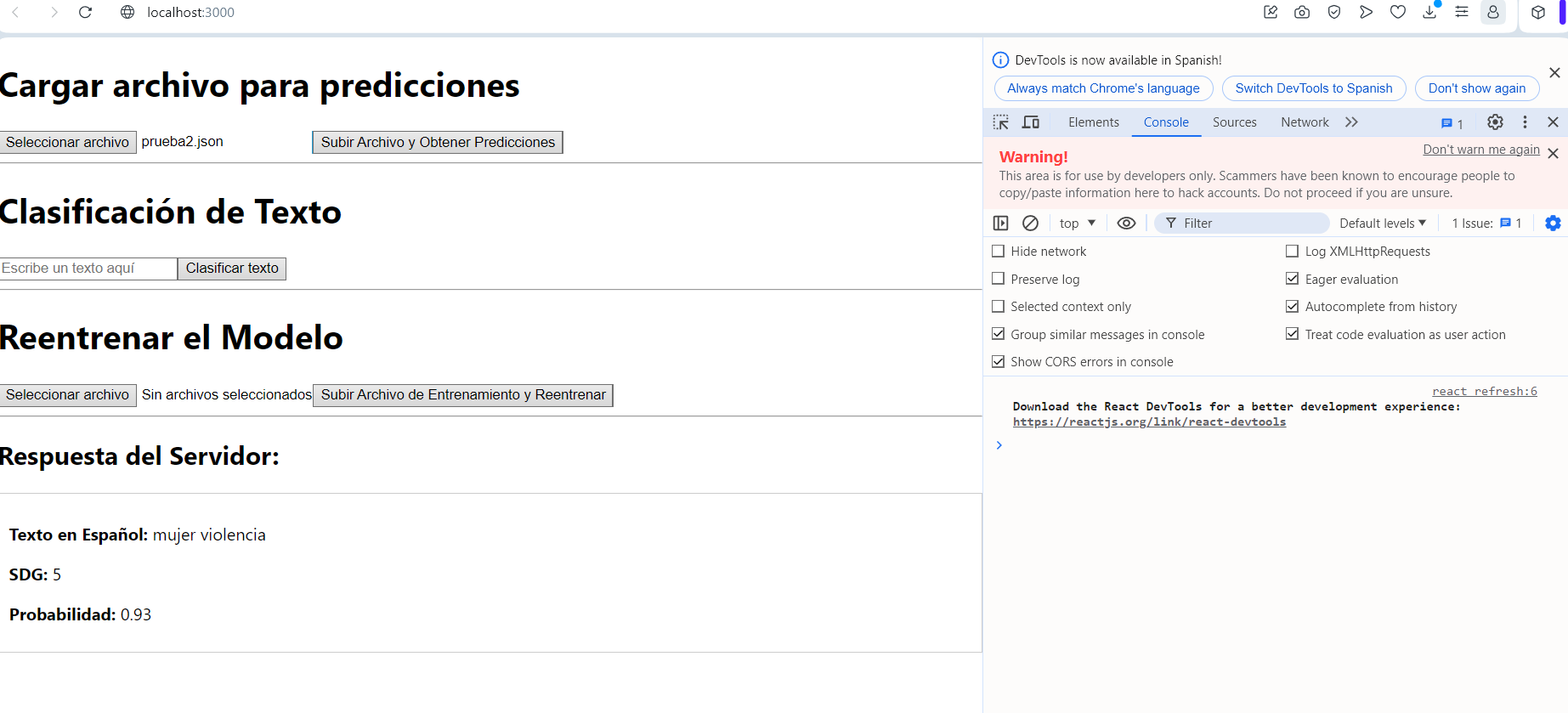




**Escenarios de prueba de la API**

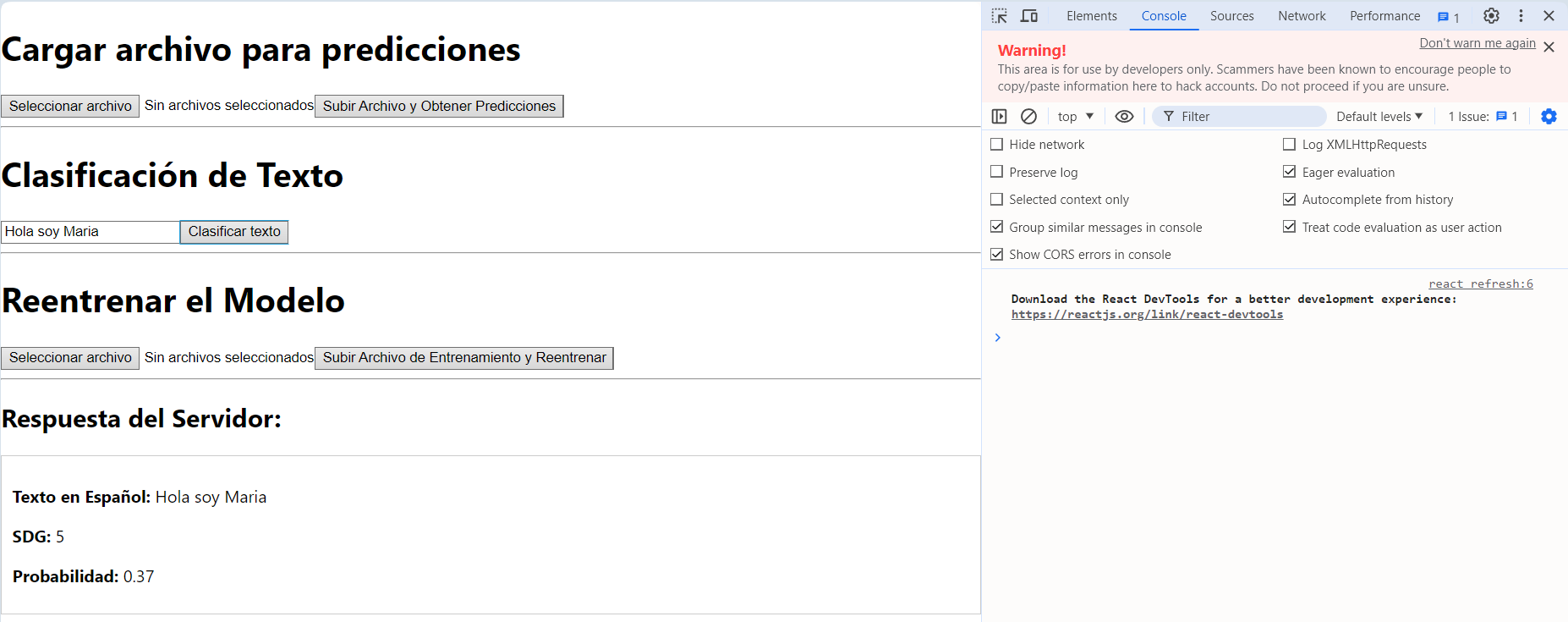
Para realizar pruebas del funcionamiento de la API, diseñamos diversos archivos almacenados en la carpeta “uploads” del proyecto. En este caso utilizamos como archivo de entrada un .json que contiene el comentario “mujer violencia”, palabras que guardan una estrecha relación con el ODS 5, por lo que muestra una alta probabilidad de éxito en la clasificación. Por lo que podemos notar un buen comportamiento de la app y el modelo de clasificación para comentarios con palabras “importantes” o con una incidencia directa en el SDG.



****

Por otro lado, también realizamos escenarios de pruebas con comentarios más comunes o generales, para verificar el comportamiento del modelo ante datos vagos. Para esto, ingresamos como input en la opción de clasificación de texto la frase “Hola, soy Maria”, en texto plano y luego es transformada a un formato JSON. Este input que podría ser considerado como una entrada válida pero con un contexto vago para el modelo. En este ejemplo obtuvimos una probabilidad de acierto baja de un 37%, dado que para el modelo es más difícil clasificar una opinión sí en esta no se muestran palabras clave.





Para el primer caso se considera coherente el resultado puesto que el SDG 5 está directamente relacionado con violencia a la mujer, y el ejemplo propuesto tiene explícitamente las palabras “mujer violencia”. A rasgos generales esto indica que la presencia de estas palabras es significativa en la clasificación del texto y ayuda a reconocer fácilmente la opinión del usuario. Por otro lado, el segundo caso muestra un desempeño bajo puesto que para entradas generales no logra reconocer el SDG, ya que para la entrada “Hola soy Maria”, no se puede saber con certeza a qué SDG hace referencia.

Para mitigar las incoherencias en la predicción y los errores de ejecución en el modelo de clasificación de texto, consideramos importante desarrollar una estrategia que incluya la ampliación del conjunto de datos de entrenamiento con frases más comunes y contextualmente ambiguas, como "Hola, soy Maria". Esto ayudaría a que el modelo maneje mejor las entradas vagas. También sería útil incluir un mecanismo de retroalimentación para ajustar el modelo en función de sus errores recurrentes y así mejorar su precisión con el tiempo.

**Maneras para reentrenar un modelo y justificación de la escogida**

Para seleccionar la manera de reentrenar un modelo de aprendizaje automático, existen diferentes enfoques dependiendo del caso de uso, la naturaleza de los datos y los objetivos del negocio. La primera opción es reentrenar el modelo con los nuevos datos únicamente, descartando los datos históricos previos (Dilmegani, 2024). Esto puede ser beneficioso cuando los datos históricos ya no son representativos del problema actual o el contexto ha cambiado significativamente, pues haría que el modelo previamente entrenado con datos viejos tenga un mal desempeño. Por ejemplo, el UNFPA realiza encuestas anualmente para conocer las diferentes opiniones de los ciudadanos, sin embargo, los temas de preocupación cambian significativamente debido a los nuevos contextos. Así, la ventaja de este enfoque es que se evita el arrastre de tendencias y patrones antiguos que ya no aplican, mientras que la desventaja es que se pierde información valiosa que los datos históricos pueden ofrecer.

La segunda opción es añadir los nuevos datos a los datos existentes y luego se reentrena el modelo con todo el conjunto de datos combinado, lo cual permite al modelo aprender tanto de los nuevos patrones como de los históricos (RandomTrees, 2024). Esto ayuda a que mantenga una buena capacidad de generalización. Por esta razón, el método es adecuado cuando se espera que los patrones y tendencias anteriores sigan teniendo relevancia y sean útiles. Por ejemplo, el UNFPA tiene datos de opiniones recopiladas durante varios años, donde gran parte de los problemas persisten. La ventaja de este modelo es que cada vez se vuelve más robusto al aprender de más datos, sin embargo, con datos masivos puede volverse costoso en almacenamiento y requerimientos computacionales.

Finalmente, el enfoque modelo de transferencia o transfer learning, se toma un modelo ya entrenado en una tarea similar se ajusta para la nueva tarea o datos específicos. Así, en lugar de entrenar un modelo desde cero, este método reutiliza el conocimiento que el modelo ha adquirido anteriormente y lo adapta al nuevo conjunto de datos (IBM, 2024). Este enfoque se usa cuando se dispone de nuevos datos pero el conjunto de datos es pequeño o se quiere hacer uso de los patrones aprendidos de fuentes más grandes. Por ejemplo, el UNFPA está interesado en utilizar el análisis de opiniones ciudadanas para identificar temas emergentes relacionados con los ODS, pero no hay suficientes datos específicos del contexto de Colombia. La ventaja del modelo es que ahorra tiempo y recursos al no tener que reentrenarlo desde cero pero, su desventaja es que puede ser menos flexible si el nuevo dominio es muy diferente al del modelo pre entrenado.

**Retos:**

1. **Creación del API**: Inicialmente, enfrentamos dificultades para definir y estructurar correctamente el API, lo que nos retrasó en su implementación. Para solucionar este reto, investigamos y pedimos ayuda a personas con mayor experiencia.
2. **Integración del Backend con el Frontend**: Encontramos varios desafíos al intentar integrar ambos componentes, principalmente en la comunicación entre servicios y en la sincronización de datos. Para esto, hicimos múltiples pruebas y buscamos por qué se daban ciertas fallas. Con base en esto, reajustamos la conexión.
3. **Desajuste en los nombres de las URLs**: Hubo discrepancias entre los nombres de las rutas en la documentación y los realmente implementados, lo que causó errores al realizar peticiones. Esto lo solucionamos conectando bien los endpoints de la API con las partes del front que la consumían.
4. **Presentación de resultados**: La visualización de los resultados en el frontend no coincidía con lo esperado debido a problemas en la manipulación de los datos y su presentación. Esto lo resolvimos abriendo un campo en el front que permite visualizar los resultados.
5. **Falta de experiencia en la creación del API**: Al inicio del proyecto, no teníamos claridad sobre el proceso de construcción del API, lo que requirió tiempo adicional de investigación y aprendizaje. Esto lo resolvimos averiguando y preguntando a personas con más experiencia.
6. **Desconocimiento en la integración del pipeline**: Nos encontramos con dificultades para configurar e integrar correctamente el pipeline de desarrollo, lo que resultó en varios ciclos de prueba y error. Esto se resolvió investigando como se hacía la integración.
7. **Fallos en el pipeline de PKL**: El pipeline basado en archivos PKL presentó errores durante su ejecución, impidiendo el correcto flujo de trabajo y la integración continua. Esto lo resolvimos realizando un modulo dentro de la API que hace las funciones del pipeline y que consume los pkl del modelo y el vectorizador. Esto permite hacer todo el procesamiento de los datos como si fuera el pkl del pipeline.

# Bibliografía

Dilmegani, C. (3 de septiembre de 2024). *Model Retraining: Why & How to Retrain ML Models?* Obtenido de AIMultiple: https://research.aimultiple.com/model-retraining/

IBM. (12 de febrero de 2024). *What is transfer learning*. Obtenido de IBM: https://www.ibm.com/topics/transfer-learning#:~:text=Transfer%20learning%20is%20a%20machine,improve%20generalization%20in%20another%20setting.

RandomTrees. (15 de abril de 2024). *Mastering Model Retraining in MLOps*. Obtenido de Medium: https://randomtrees.medium.com/mastering-model-retraining-in-mlops-4bb961ee7070#:~:text=Model%20retraining%2C%20in%20essence%2C%20involves,present%20in%20the%20underlying%20data.

*Retraining Models on New Data - Amazon Machine Learning*. (2024). Amazon.com. <https://docs.aws.amazon.com/machine-learning/latest/dg/retraining-models-on-new-data.html>

*Amazon Machine Learning Developer Guide Version Latest*. (2024). <https://docs.aws.amazon.com/pdfs/machine-learning/latest/dg/machinelearning-dg.pdf>