ASIGNATURA: INTELIGENCIA DE NEGOCIOS 2023-II
PROFESOR: FABIÁN CAMILO PEÑA LOZANO
ESTUDIANTE: JAVIER CERINO, DANIEL ARANGO, MARCO ZULIANI
CÓDIGOS: 202020873, 202110646, 202022412
PROGRAMA DE ESTUDIOS: INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

PROYECTO 1- Etapa 2

Tabla de Contenido

1. Proceso de automatización de la preparación de datos, construcción del modelo y acceso por medio de API.

Para el proceso de automatización de la preparación de datos es importante tener en cuenta aquellas transformaciones fundamentales que permiten facilitar el procesamiento de datos a la hora de ser pasados al algoritmo. Estos cambios permiten que el algoritmo pueda procesar los datos de manera natural sin que ningún tipo de anomalía cree un sesgo y evite que realice una correcta clasificación. Una vez identificados dichas transformaciones indispensables es necesario la construcción de un *Pipeline*, el cual no es más que una serie de procesos que ejecutará el modelo una vez sea cargada y reciba un nuevo conjunto de datos, que permita adecuar los datos para que finalmente pueda ser procesado por el algoritmo de clasificación y brinde la predicción o predicciones correspondientes.

```
dataPipe = pd.read_excel("data/cat_6716.xlsx")
X_train_Pipe, X_test_Pipe, y_train_Pipe, y_test_Pipe = train_test_split(dataPipe[['Textos_espanol']], dataPipe["sdg"], test_size=0.3, stratify=dataPipe["sdg"], random_state=1)

class TextPreprocessor(BaseEstimator, TransformerMixin):
    def fit(self, X, y=None):
        return self

    def transform(self, X):
        X = X.astype(str)
        X = X.apply(fix_malformed_words)
        X = X.apply(remove_punctuation)
        X = X.apply(replace_numbers_with_spanish_text)
        return X

text_pipe = Pipeline([
    ('transform', TextPreprocessor())
])

pipeline = Pipeline(steps = [
    ("preprocessor", text_pipe),
    ("vectorizer", TfidfVectorizer(tokenizer=word_tokenize, stop_words=stop_words, lowercase=True)),
    ("classifier", linear_model.LogisticRegression(multi_class = 'ovr', solver = 'liblinear',random_state=2))
])
```

Imagen 1. Pipeline construido.

Para el proyecto se construyó el Pipeline de la *imagen 1.* En dicho Pipeline se realizan varias transformaciones las cuales permiten, antes de realizar las predicciones, preparar correctamente los datos. En primer lugar, se crea una clase *TextPreprocessor* el cual se encarga de realizar las siguientes transformaciones:

1. **Arreglar codificación de las palabras**: Esta transformación permite que los datos no tengan anomalías de codificación, esto quiere decir que no se encuentre con palabras que por tener una tilde se vean de la forma "comunicaci@~n", sino que cuenten con la codificación correcta y luzcan como "comunicación".
2. **Remover puntuación:** La eliminación de caracteres especiales reduce la complejidad del texto, lo que puede ser beneficioso en tareas de análisis de texto o procesamiento automatizado. Esto facilita la tarea de comprensión y procesamiento del texto.
3. **Reemplazar números por texto:** es importante porque ayuda a garantizar la consistencia de formato, a seguir las reglas lingüísticas, a minimizar errores de traducción, a adaptarse al contexto cultural y a facilitar el procesamiento lingüístico en modelos.

En segundo lugar, se utiliza la técnica de vectorización escogida TF-IDF, un vectorizador se encargan de asignar valores numéricos a las palabras o términos en un corpus de texto para que las computadoras puedan trabajar con ellos de manera eficiente. El vectorizador crea vectores de características que representan el contenido y las características del texto. TF-IDF fue escogido ya que asigna un peso numérico a cada término en función de cuán frecuentemente aparece en un documento (TF) y cuán común es en el corpus en general (IDF). Esto significa que los términos que son específicos del documento y no se encuentran en muchos otros documentos obtendrán un peso más alto, lo que los hace más relevantes en la representación del texto.

Finalmente, una vez ya el conjunto de datos haya sido transformado correctamente estos son delegados al algoritmo para realizar que luego de correr sus pasos correspondientes arroje una predicción. En el caso del proyecto el algoritmo escogido fue "*Logistic Regression"* este algoritmo implica seleccionar una clase como objetivo y considerar todas las demás clases como una segunda categoría virtual. Luego, se aplica la regresión logística binaria a esta configuración. Repetimos este proceso para cada una de las clases presentes en el conjunto de datos. Como resultado, se obtiene clasificadores binarios individualizados, cada uno diseñado para identificar una clase específica dentro del conjunto de datos. Se ponderó este algoritmo sobre los otros probados ya que se requiere una solución computacionalmente eficiente y rápida, especialmente en conjuntos de datos grandes. La Regresión Logística utiliza menos recursos, es más rápida en entrenamiento y predicción, y es fácilmente accesible a través de bibliotecas de aprendizaje automático. Sin embargo, Random Forest puede ser más adecuado para casos donde la precisión es crítica y se pueden asignar más recursos computacionales.

Persistencia y acceso mediante API:

Una vez ya construido y entrenado el Pipeline se persiste el mismo en un archivo ". joblib" que permite guardar todo el flujo de procesamiento de datos y modelado en un solo archivo para su posterior uso. En el caso del proyecto este archivo es cargado en el API construido para que pueda realizar predicciones sobre fragmentos de textos suministrado por un usuario o sobre archivos que el usuario cargue. Para el API se

utilizó el framework "*FastAPI",* el cual facilita la creación de aplicaciones web y API REST de manera eficiente gracias a su enfoque de programación asincrónica y su generación automática de documentación interactiva basada en especificaciones OpenAPI.

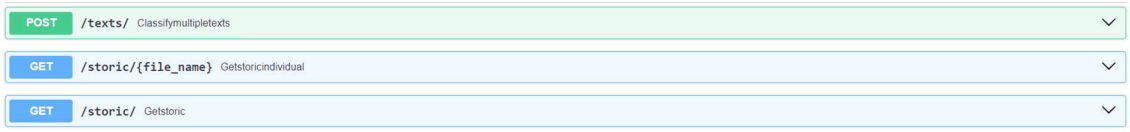

Imagen 2. EndPoints del API.

El Api creado llamado "*ODS Classifier"* cuenta con 2 recursos y 3 operaciones las cuales facilitan el manejo histórico de las consultas del usuario y la predicción de nuevas, estos son posibles verlos en la imagen 2.

Recursos del API:

El API cuenta con dos recursos principales: textos y historias. Por el lado de textos este recurso representa un conjunto de párrafos o peticiones de ciudadanos los cuales utilizan el mecanismo de participación ciudadana para dar a conocer los problemas que identifican. Por otro lado, las historias representan el consolidado de peticiones realizadas que son guardadas en un archivo y pueden ser consultadas y modificadas.

Representación de los recursos:

Histórico de predicción:

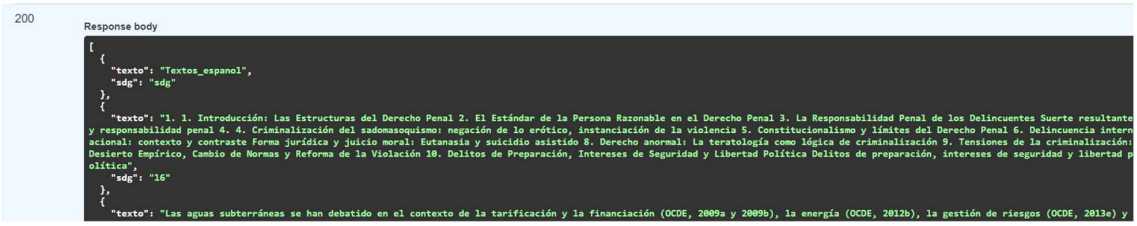

Imagen 3. Representación histórica de predicción.

En este caso cada historia tiene un conjunto de textos los cuales poseen el texto suministrado por el usuario y una predicción "*SDG"* realizada por el modelo.

Textos:

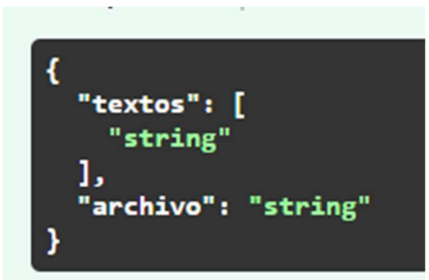

Imagen 4. Replantación texto.

Textos es una lista donde recibes los textos que vas a revisar separados por comas, estos tienen asociado un archivo el cual indica en donde se esta guardando esa predicción.

2. Desarrollo de aplicación y justificación.

Para la aplicación se buscó tener una interfaz lo suficientemente intuitiva teniendo en cuenta que el público objetivo de la aplicación son analistas de textos que utilizarán la aplicación para poder clasificar de manera mucho más eficiente los problemas manifestados por la herramienta de participación ciudadana. Es por ello por lo que se escogió una estructura similar a la que posee “ChatGpt” esto dado que es una interfaz que resulta común entre los usuarios y permite revisar el histórico de peticiones y la generación de nuevas peticiones.

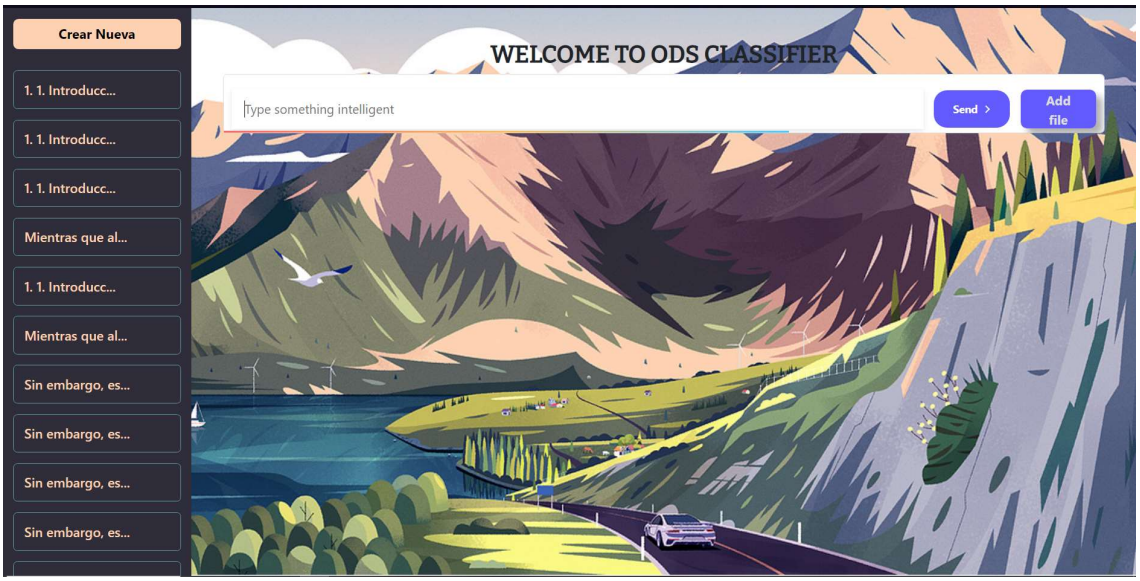

Imagen 4. Interfaz de la aplicación.

En la interfaz mostrada en la imagen 4 es posible ver en el lado derecho un histórico de las consultas realizadas y en el lado central de la aplicación una sección donde el usuario puede tanto ingresar su petición o problema identificado como cargar un archivo donde haya consolidado varios problemas. Luego de ello cada una de las peticiones o archivos adjuntos se les realizará una clasificación para que luego los directivos puedan filtrar y revisar las peticiones relacionadas con determinado objetivo de desarrollo sostenible.

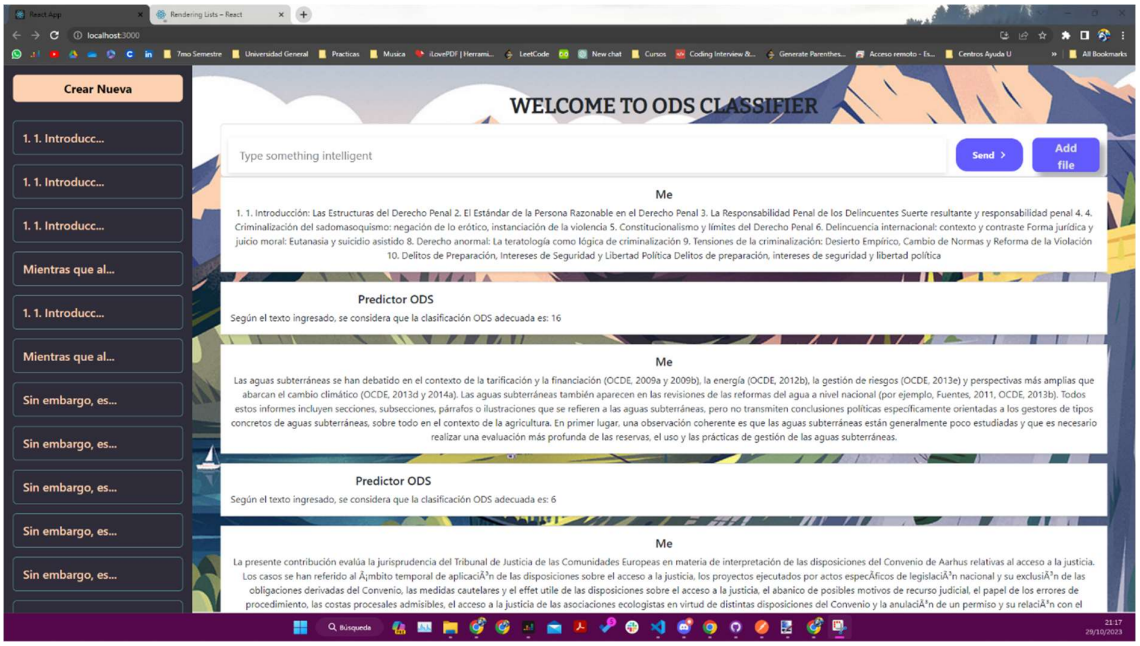

Imagen 5. Muestra de chat con usuario.

Descripción del usuario objetivo:

Rol	Tipo de actor	Beneficio	Riesgo
Analista de textos de UNFPA	Usuario – Cliente	Disminuye el tiempo necesario para clasificar un texto para poder realizar reportes de estado de las diferentes áreas de interés para identificar un potencial proyecto a desarrollar.	Una mala predicción de los contenidos de los artículos lo que generaría un impacto grave en la organización de los trabajos malgastando tiempo y recursos de áreas a los que no les compete la información de dicho artículo.
Trabajador de la ONU	Financiador	Una disminución en costos y tiempo que permite analizar más textos en el mismo tiempo permitiendo crear proyectos con una mejor perspectiva.	En caso de que el modelo no funcione la inversión de capital que se realizó en la construcción en este es un desperdicio que se pudo haber utilizado en

			contratar más empleados capaces de suplir el modelo.
Comunidad publicadora del artículo.	Beneficiado	Permite obtener una retroalimentación y/o apoyo de la ONG o algún ente competente de forma mucho más rápida y eficiente.	Una mayor demora en al momento de recibir financiación o apoyo de expertos debido a la ausencia de categorización de los proyectos que se podrían generar por una mala clasificación de los artículos.

Teniendo en cuenta la tabla de usuarios, esta aplicación web busca ser útil para los analistas de textos los cuales a diario reciben una gran cantidad de problemas y de manera manual deben encargarse de clasificar y filtrar todos los problemas para que luego puedan ser procesados. La herramienta busca que el experto pueda subir grandes archivos los cuales quedarán almacenados en un histórico y luego puedan ser consultados. Esto permite que a la hora en la que se quiera realizar políticas publicas que vayan en pro a los problemas manifestados por los ciudadanos, estos puedan ser filtrados y encontrados según su clasificación dada, con esto se logra un mayor dinamismo en las discusiones y posibles propuestas que se puedan generar ya que tienen un banco de problemáticas el cual pueden analizar y atender de manera más eficiente y selecta.

Impacto del proyecto dentro de la problemática:

El proyecto es de vital importancia dentro del contexto de UNFPA y la Agenda 2030 para el Desarrollo Sostenible, ya que aborda de manera eficiente uno de los desafíos más críticos en la identificación y seguimiento de problemas relacionados con los Objetivos de Desarrollo Sostenible (ODS). Al automatizar la clasificación de textos que representan las preocupaciones y solicitudes de la comunidad local, la aplicación y el modelo de clasificación permiten a UNFPA acelerar el proceso de identificación de problemas y relacionarlos con los ODS. Esto no solo ahorra tiempo y recursos al reducir la dependencia de la labor manual de analistas de texto, sino que también promueve una mayor participación ciudadana al valorar la voz de la comunidad en la formulación de políticas y programas orientados al desarrollo sostenible.

Además, este proyecto mejora significativamente la capacidad de UNFPA para el seguimiento y la evaluación de políticas públicas, permitiendo una toma de decisiones más informada y eficaz. Al relacionar automáticamente las contribuciones de los ciudadanos con los ODS, se crea un puente crucial entre la participación ciudadana y la implementación de políticas basadas en datos, lo que es esencial para alcanzar

los objetivos de desarrollo sostenible y reducir la brecha entre las necesidades locales y las soluciones políticas. En última instancia, el proyecto fortalece la capacidad de UNFPA para cumplir con su misión de promover el bienestar de la población y garantizar un enfoque más eficaz en la planificación y ejecución de políticas públicas orientadas al desarrollo sostenible.

La interfaz de la aplicación diseñada para este proyecto ha sido cuidadosamente concebida para ofrecer a los analistas de texto una experiencia de navegación intuitiva y eficiente. Inspirada en la estructura amigable de "ChatGpt", la interfaz proporciona una familiaridad que simplifica la interacción de los usuarios con la herramienta. Al presentar un histórico de consultas en el lado derecho y una sección central que permite la carga de archivos o la introducción de solicitudes, se facilita la navegación y el manejo de datos, brindando a los usuarios la comodidad de acceder rápidamente a sus registros pasados y de cargar múltiples problemas para su posterior clasificación. Esta orientación hacia la usabilidad se alinea perfectamente con el propósito de la aplicación: permitir a los analistas de texto clasificar de manera más eficiente y selecta los problemas manifestados por la ciudadanía, contribuyendo así a un proceso más ágil y eficaz en la toma de decisiones relacionadas con políticas públicas.

3. Trabajo en equipo.

Roles:

Líder del proyecto: Marco Zuliani – Desarrollo front end

Ingeniero de datos: Daniel Arango – Desarrollo front end

Ingeniero de software responsable del diseño de la aplicación y resultados: Marco Zuliani y Daniel Arango

Ingeniero de software responsable de desarrollar la aplicación final: Javier Cerino – Desarrollo BackEnd

Reuniones:

Se realizaron las siguientes reuniones: Domingo 15 de octubre: reunión de lanzamiento y planeación. En esta se leyó el enunciado y se definieron los roles a usar. De igual forma se planearon las siguientes reuniones del equipo. Martes 17 de octubre: reunión de ideación: En este se definieron los temas de organización/ empresa/ institución para darles el apoyo necesario con la construcción de nuestro modelo. La primera reunión de seguimiento se realizó el Viernes 20 de octubre en donde se definieron cuáles serían los recursos y diseño de la interfaz de la aplicación web . El domingo 22 de octubre se concluyó el desarrollo del backend. El Miercoles 25 se redactó el documento Word a la vez que se desarrollaba el front. El Jueves 26 se tuvo la reunión con el equipo de estadística para ultimar detalles. El domingo 29 se terminaron de pulir detalles del documento y se grabó el video correspondiente.

4. Referencias

Hamdaoui, Y. (2021, Marzo 24). *TF(Term Frequency)-IDF(Inverse Document Frequency) from scratch in python .* Medium. https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558

K, G. L. (2014, octubre 5). Lista de stop words o palabras vacías en español. *SEO para Google*. https://googleseo.marketing/lista-de-stop-words-o-palabras-vacias-en-espanol/

Moran, M. (s. f.). La Agenda para el Desarrollo Sostenible. *Desarrollo Sostenible*. Recuperado 14 de octubre de 2023, de https://www.un.org/sustainabledevelopment/es/development-agenda/

Nations, U. (s. f.). *Naciones Unidas | Paz, dignidad e igualdad
en un planeta sano*. United Nations; United Nations. Recuperado 14 de octubre de 2023, de https://www.un.org/es/

UNFPA Colombia | UNFPA en Colombia. (s. f.). Recuperado 14 de octubre de 2023, de https://colombia.unfpa.org/es/unfpa-en-colombia

Subasi, A. (2020). Machine learning techniques. En A. Subasi (Ed.), *Practical Machine Learning for Data Analysis Using Python* (pp. 91–202). Elsevier.

Multiclass logistic regression Using sklearn. (2020, junio 1). Kaggle.com; Kaggle. https://www.kaggle.com/code/satishgunjal/multiclass-logistic-regression-using-sklearn