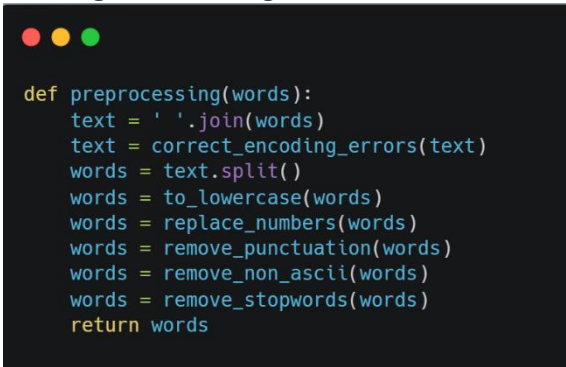


## Proyecto 1 Etapa 2

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

Para esta etapa del proyecto se utilizó fastAPI para el backend donde se encuentran los endpoints y react para el frontend.

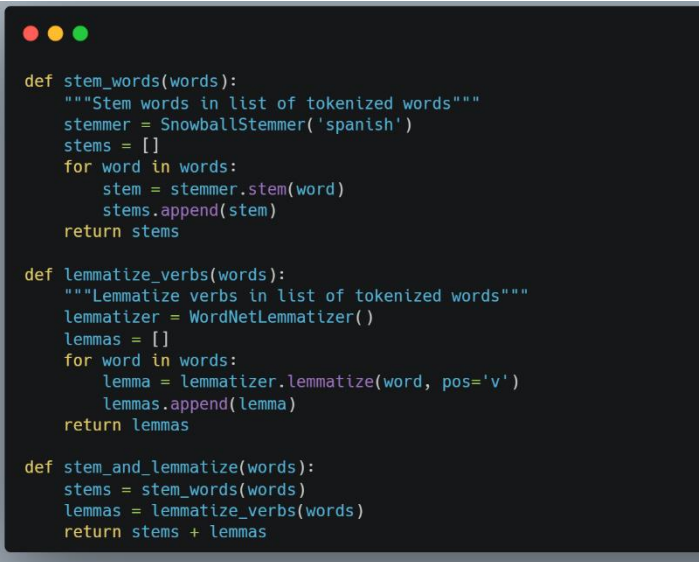
Este proceso fue realizado por el ingeniero de datos y consta de 4 etapas. En la primera etapa se hace la automatización del procesamiento de datos, primero se tokenizaban las palabras lo que significa dividir cada texto en sus palabras o tokens de manera aislada. Luego se corregían errores de codificación de las palabras, quitar signos de puntuación, quitar stopwords las cuales no aportan al modelo porque son palabras del común. Esto se hace con la siguiente código.



```
def preprocessing(words):  
    text = ' '.join(words)  
    text = correct_encoding_errors(text)  
    words = text.split()  
    words = to_lowercase(words)  
    words = replace_numbers(words)  
    words = remove_punctuation(words)  
    words = remove_non_ascii(words)  
    words = remove_stopwords(words)  
    return words
```

Ilustración 1. Vista del código función preprocessing

El segundo paso para el procesamiento de datos se hace dos funciones lemmatize y stemmatize, las cuales reducen la palabra a su base y quitan los sufijos o prefijos. Esto se hace en las siguientes funciones:



```
def stem_words(words):  
    """Stem words in list of tokenized words"""  
    stemmer = SnowballStemmer('spanish')  
    stems = []  
    for word in words:  
        stem = stemmer.stem(word)  
        stems.append(stem)  
    return stems  
  
def lemmatize_verbs(words):  
    """Lemmatize verbs in list of tokenized words"""  
    lemmatizer = WordNetLemmatizer()  
    lemmas = []  
    for word in words:  
        lemma = lemmatizer.lemmatize(word, pos='v')  
        lemmas.append(lemma)  
    return lemmas  
  
def stem_and_lemmatize(words):  
    stems = stem_words(words)  
    lemmas = lemmatize_verbs(words)  
    return stems + lemmas
```

Ilustración 2. Vista del código para lematizar y stemmatizar las palabras

Luego estas transformaciones del preprocessing se agrupan en única función para poder ser usado en el pipeline.

```
def pipeline_datos(df: pd.DataFrame):
    # Realizar una copia del conjunto de datos
    df_pipeline = df.copy()

    # Renombrar las columnas para que sean más descriptivas
    df_pipeline.rename(columns={'Textos_espanol': 'Textos'}, inplace=True)

    # Aplicar el preprocesamiento
    df_pipeline['words'] = df_pipeline['Textos'].apply(word_tokenize).apply(preprocessing)
    df_pipeline['words'] = df_pipeline['words'].apply(stem_and_lemmatize)
    df_pipeline['words'] = df_pipeline['words'].apply(lambda x: ' '.join(map(str, x)))

    return df_pipeline['words']
```

Ilustración 3. Vista de la función del pipeline\_datos que agrupa las transformaciones realizadas sobre los datos.

Ahora si hacemos el pipeline y exportamos el modelo como joblib. En este modelo primero se aplica la función anteriormente descrita pipeline\_datos, luego se hace vectorización tf-idf para convertir textos en vectores numéricos con un max\_features de 1000 lo que significa que solo se tomarán en cuenta las 10,000 palabras más importantes del corpus para representar los textos. Para las predicciones se utiliza el algoritmo de Regresión Logística Multinomial; previamente en la etapa 1 ya habíamos encontrado los mejores hiperparametros entonces estos se incluyen en el pipeline.

```
# Crear el pipeline con preprocesamiento y modelo de clasificación
pipeline = Pipeline([
    ('preprocesamiento', FunctionTransformer(pipeline_datos)),
    ('tfidf', TfidfVectorizer(max_features=10000)),
    ('clf', LogisticRegression(C=100, max_iter=100, multi_class='multinomial', solver='newton-cg'))
])

# Entrenar el pipeline con los datos (X_train['Textos_espanol']) y las etiquetas (X_train['sdg'])
pipeline.fit(X_train.drop(columns=['sdg']), X_train['sdg'])

# Guardar el pipeline entrenado en un archivo
joblib.dump(pipeline, 'backend/src/assets/pipeline_funcional.joblib')
```

Ilustración 4. Construcción del pipeline con los hiperparametros encontradas anteriormente.

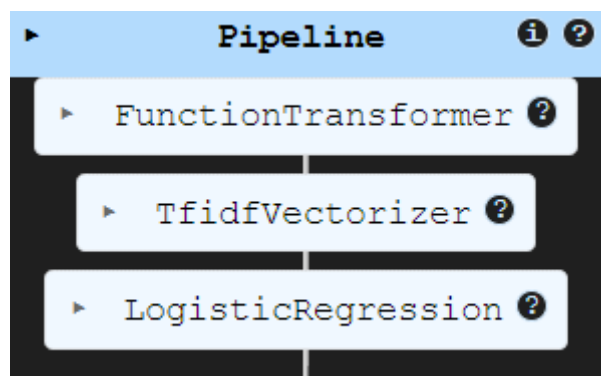


Ilustración 5. Vista de los pasos del Pipeline

Ahora se procede a crear el primer endpoint para hacer predicciones sobre textos que recibe mediante el body en formato json:

```
@app.post("/predict")
def make_prediction(dataModel: DataModel):
    # Convertir la entrada del modelo Pydantic a DataFrame
    df = pd.DataFrame(dataModel.dict(), columns=dataModel.dict().keys())

    # Usar el pipeline cargado para hacer la predicción
    predictions = pipeline.predict(df).tolist() # Predecir todas las instancias
    probabilities = pipeline.predict_proba(df).tolist() # Obtener las probabilidades para todas

    # Devolver todas las predicciones y probabilidades
    return {"predictions": predictions, "probabilities": probabilities}

return {"important_words": review_important}
```

*Ilustración 6. Endpoint para las predicciones de los textos introducidos*

En este endpoint se recibe los datos de entrada a través de un modelo de Pydantic, los convierte a un DataFrame de pandas para facilitar su manipulación. Luego, carga y utiliza el modelo en formato joblib (creado anteriormente) para hacer predicciones sobre esos datos, generando una lista de etiquetas predichas para cada instancia. Además, calcula las probabilidades asociadas a cada predicción usando el método `predict_proba`, que devuelve las probabilidades de pertenencia a cada clase. Finalmente, la función retorna un diccionario con dos claves: "predictions", que contiene las etiquetas predichas, y "probabilities", que contiene las probabilidades correspondientes a esas predicciones para cada instancia.

Ahora antes de construir el segundo endpoint se evaluó cuál de las 3 formas de reentrenamiento se acomoda más a nuestros intereses. Este análisis se encuentra en la siguiente tabla.

Método de Reentrenamiento	Descripción	Ventaja	Desventaja
Completo	Se entrena modelo desde 0 usando tanto datos antiguos como nuevos	Mejor rendimiento al estar actualizado con todos los datos.	Es computacionalmente costoso, ya que requiere reentrenar el modelo desde el inicio.
Incremental	El modelo es actualizado únicamente con los nuevos datos	Es eficiente en cuanto a tiempo y recursos, ya que solo se agregan los nuevos datos.	Puede no mejorar tanto como el reentrenamiento completo, ya que los datos antiguos no se reconsideran.

Validación cruzada	El modelo se entrena mediante una estrategia de <b>validación cruzada</b> como parte del proceso de actualización del modelo, evaluando el impacto de los nuevos datos antes de modificar definitivamente el modelo.	Ofrece una forma controlada de evaluar si los nuevos datos realmente mejoran el modelo.	Es más lento y consume más recursos porque el modelo se entrena y valida varias veces en diferentes subconjuntos de los nuevos datos.
--------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------

Escogimos el reentrenamiento completo porque, aunque es más intensivo en términos de recursos y tiempo, garantiza que el modelo se actualice utilizando tanto los datos antiguos como los nuevos, proporcionando una visión global de todo el conjunto de datos. Este enfoque asegura que no se pierda información importante de los datos previos y que el modelo esté completamente ajustado con todo el conocimiento disponible.

Cabe resaltar que, no se cambian los hiperparámetros del modelo reentrenado. Solo se vuelve a reentrenar, pero no se realiza un ajuste fino nuevamente. Esto con el objetivo de evitar el sobreajuste cuando los nuevos datos son significativos y para realizar de forma más rápida todo el proceso del reentrenamiento, pues la búsqueda de hiperparámetros desde cero implicaría un mayor tiempo de duración de la petición.

Ahora este es el segundo endpoint, el cual incluye el método de reentrenamiento completo.

```

@app.post("/reentrenamiento")
def reentrenar_modelo(data: ReentrenamientoModel):
    textos_nuevos = data.textos
    etiquetas_nuevas = data.etiquetas

    if len(textos_nuevos) != len(etiquetas_nuevas):
        raise HTTPException(status_code=400, detail="El número de textos y etiquetas no coincide")

    # Cargar el dataset original
    data_original = pd.read_excel("data/ODScat_345.xlsx")
    textos_originales = data_original['Textos_espanol']
    etiquetas_originales = data_original['sdg']

    # Combinar los datos originales con los nuevos
    textos_combinados = pd.concat([textos_originales, pd.Series(textos_nuevos)],
    ignore_index=True)
    etiquetas_combinadas = pd.concat([etiquetas_originales, pd.Series(etiquetas_nuevas)],
    ignore_index=True)

    # Crear el DataFrame con los textos y etiquetas combinados
    df = pd.DataFrame({'Textos_espanol': textos_combinados, 'sdg': etiquetas_combinadas})

    # Dividir los datos en entrenamiento y prueba (80% entrenamiento, 20% prueba)
    X_train, X_test, y_train, y_test = train_test_split(
        df['Textos_espanol'], df['sdg'], test_size=0.2, random_state=42, stratify=df['sdg']
    )

    pipeline1 = pipeline
    pipeline1.fit(X_train, y_train)

    # Realizar predicciones en el conjunto de prueba
    y_pred = pipeline1.predict(X_test)

    # Calcular las métricas de desempeño
    precision = precision_score(y_test, y_pred, average='weighted', zero_division=0)
    recall = recall_score(y_test, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=0)

    # Formatear las métricas a 5 cifras decimales
    precision = "{0:.5f}".format(precision)
    recall = "{0:.5f}".format(recall)
    f1 = "{0:.5f}".format(f1)

    # Guardar el modelo actualizado
    dump(pipeline1, model_path)

    return {
        "message": "Modelo reentrenado correctamente",
        "metrics": {
            "precision": precision,
            "recall": recall,
            "f1_score": f1
        }
    }

```

Ilustración 7. Endpoint del reentrenamiento del modelo

Primero, recibe listas de nuevos textos y sus correspondientes etiquetas las cuales provienen del json enviado y valida que ambas listas tengan la misma longitud, esto porque se necesita que cada texto tenga su clasificación para poder reentrenar el modelo. Luego, carga el conjunto de datos original desde un archivo Excel (ODScat\_345.xlsx), y concatena estos datos originales con los nuevos datos proporcionados y crea un DataFrame que luego es dividido en conjuntos de entrenamiento y prueba, utilizando un 80% para entrenamiento y 20% para prueba, asegurando una división estratificada basada en las etiquetas. El pipeline de machine learning previamente cargado se reentrena con los datos combinados y se evalúa con el conjunto de prueba. Por último, se obtienen las meticas de desempeño Precision, Recall, y F1-score. Finalmente, el modelo reentrenado se guarda **reemplazando** la versión anterior, y el endpoint devuelve un mensaje confirmando el éxito del reentrenamiento, junto con las métricas calculadas.

## 2. Desarrollo de la aplicación y justificación

Para este apartado, se pensó en los usuarios dentro de las organizaciones que podrían usar la aplicación orientada a la clasificación de opiniones ciudadanas en relación con los Objetivos de Desarrollo Sostenible (ODS). Hemos diseñado el servicio para servir a varios actores dentro del Fondo de Poblaciones de las Naciones Unidas (UNFPA), así como a organizaciones públicas y privadas relacionadas con los ODS 3, 4 y 5 (Salud y Bienestar, Educación de Calidad, e Igualdad de Género).

Para el **UNFPA**, esta herramienta proporcionaría una fuente de datos en tiempo real sobre las opiniones de los ciudadanos acerca de los problemas relacionados con los ODS, facilitando el análisis de las principales preocupaciones y áreas de mejora. Al recopilar y analizar opiniones, el UNFPA podría identificar tendencias y crear políticas públicas más alineadas con las necesidades de la población, promoviendo un mayor bienestar y desarrollo.

Por otro lado, las **entidades públicas colaboradoras** encontrarían en esta aplicación una manera de evaluar el impacto de sus políticas y proyectos relacionados con los ODS. Al analizar las opiniones clasificadas por ODS, se puede determinar la efectividad de las iniciativas implementadas en cada sector (salud, educación e igualdad) y ajustar las políticas en consecuencia, maximizando el impacto de los recursos destinados.

Las **empresas de tecnología y consultoría especializadas** encontrarían valor en este servicio, ya que podrían utilizar los datos recopilados para ofrecer servicios de consultoría más informados y personalizados a sus clientes del sector público. La aplicación les proporcionaría una fuente de datos actualizada y relevante sobre la percepción de los ciudadanos respecto a los ODS, mejorando su capacidad para ofrecer recomendaciones estratégicas basadas en datos.

Finalmente, tanto los **ciudadanos** como las **organizaciones civiles** se benefician al tener acceso a una plataforma donde pueden compartir y consultar opiniones sobre los problemas que enfrentan en las áreas de salud, educación y género. Esta retroalimentación ayuda a mejorar la toma de decisiones a nivel institucional y asegura que las políticas públicas reflejen las necesidades reales de la población.

Nos enfocaremos específicamente en las empresas de tecnología y consultoría especializadas, reconociendo su importancia en el desarrollo y aplicación de soluciones tecnológicas para el análisis de datos a gran escala. Decidimos que la información recopilada se presentará a través de una aplicación, donde las opiniones de los ciudadanos serán procesadas y clasificadas mediante un modelo predictivo en tiempo real. Esta decisión busca facilitar el acceso y comprensión de los datos para estas organizaciones, permitiéndoles aprovechar eficazmente la información para ofrecer soluciones más informadas y adaptadas a las necesidades relacionadas con los ODS.

Tabla 1 Mapa de beneficiarios de la solución web

Rol	Beneficio
Ciudadanos	Sus opiniones serán analizadas automáticamente para identificar problemáticas relevantes relacionadas con los ODS 3, 4 y 5.
Fondo de Poblaciones de las Naciones Unidas (UNFPA)	Obtiene un mecanismo eficiente para analizar grandes volúmenes de datos de opiniones ciudadanas, lo que agiliza la toma de decisiones para los ODS.
Entidades Públicas Colaboradoras	Aseguran que el modelo cumpla con los estándares de calidad y privacidad de datos, lo que mejora la confianza en el análisis de opiniones.
Financiadores (UNFPA)	Proporciona los recursos financieros necesarios para el desarrollo y ejecución del proyecto, asegurando la continuidad del mismo.

### Funcionalidades de la aplicación web:

El desarrollo de la aplicación web se realiza utilizando servicios construidos con FastAPI y React JS, presentando cuatro pestañas principales:

- **Inicio:** Página estática que muestra el contexto resumido de la aplicación, el objetivo del proyecto y los actores beneficiados.
- **Contexto:** Página estática que establece el objetivo de esta etapa, presentando nuevamente los actores involucrados y el contexto de negocio relacionado con los ODS.
- **Predicciones:** Página donde se pueden observar las opiniones de la base de datos y hacer predicciones para nuevas opiniones en el endpoint `/predict/`. Introduciendo una opinión, la página otorga una calificación relacionada con los ODS por medio del modelo y persiste la información en la base de datos.
- **Reentrenar:** En esta página, los usuarios pueden cargar nuevos datos en formato CSV para reentrenar el modelo de clasificación en el endpoint `/reentrenamiento/`. El reentrenamiento permite actualizar el modelo con datos adicionales sin cambiar los hiperparámetros, lo que garantiza que el modelo sea capaz de adaptarse a nuevas opiniones y tendencias. Esta funcionalidad asegura que la clasificación de opiniones siga siendo precisa y relevante con el tiempo.

El código de la aplicación está disponible en el repositorio del proyecto y también se puede ver en funcionamiento en el siguiente video:

### 3. Resultados

Para esta segunda etapa del proyecto, se automatizó el desarrollo de la analítica de textos creada en la primera etapa. En esta ocasión, la aplicación fue diseñada con un enfoque más orientado al usuario, donde los principales usuarios serían las organizaciones y entidades involucradas en la evaluación de los ODS, quienes estarían probando y utilizando la herramienta.

La aplicación incluye una **página de inicio**, que presenta el objetivo del proyecto y algunos aspectos técnicos utilizados, como el modelo de clasificación, la técnica de vectorización aplicada a las opiniones, y algunas métricas clave para evaluar el desempeño del modelo. También se presenta el requerimiento del usuario a modo de

historia de usuario para contextualizar el desarrollo de este proyecto. Dado que esta sección utiliza un lenguaje más técnico, se creó una **sección de contexto**, donde se explica de manera más accesible el propósito del proyecto, los actores involucrados, y los beneficios que estos obtendrían al usar la aplicación.

En la pestaña de **Predicciones**, los usuarios pueden ingresar nuevas opiniones ciudadanas ya sea manualmente o cargando un archivo CSV. La aplicación utilizará un modelo de clasificación para predecir a cuál de los Objetivos de Desarrollo Sostenible (ODS) 3 (Salud y Bienestar), 4 (Educación de Calidad), o 5 (Igualdad de Género) pertenece cada opinión. El modelo analizará el texto proporcionado y mostrará las probabilidades de que la opinión esté relacionada con cada uno de estos ODS. Además, la aplicación es capaz de gestionar textos con posibles errores ortográficos sin perder precisión en la clasificación, asegurando un análisis robusto y eficiente.

Finalmente, se ha implementado una nueva pestaña de **Reentrenar el Modelo**, donde los usuarios pueden cargar archivos CSV con nuevas opiniones. Este proceso permite actualizar y mejorar el modelo con datos adicionales, garantizando que las clasificaciones de las opiniones sigan siendo precisas y actualizadas. De esta manera, el modelo se mantiene en constante evolución para adaptarse a los cambios en las percepciones ciudadanas y sus nuevas inquietudes en relación con los ODS.

Los comentarios generales proporcionados por las personas que han probado la aplicación destacan que es intuitiva y rápida, y que la sección de contexto es especialmente útil para entender mejor el propósito y los beneficios de la herramienta. Además, se valoró positivamente la estética y el diseño de la aplicación, que facilita su uso.

#### 4. Trabajo en Equipo

Actividad	Santiago Jaimes	Nicolas Rincón	Nicolas Casas
Implementación de automatización de preparación de datos	X	X	
Construcción de un Pipeline			X
Construcción de la API		X	X
Validación de proceso y resultado de negocio	X		X
Desarrollo de aplicación web		X	
Realizar video	X		
<b>Aporte total</b>	33.33%	33.34%	33.33%

**Dedicación horaria:** Nicolas Rincón = 7 horas aprox, Nicolas Casas = 6 horas aprox, Santiago = 6 horas aprox

Rol desempeñado	Santiago Jaimes	Nicolas Rincón	Nicolas Casas
Líder de proyecto	X		
Ingeniero de datos	X		



Ingeniero de software responsable del diseño de la aplicación y resultados			X
Ingeniero de software responsable de desarrollar la aplicación final		X	

Reunión	Fecha	Descripción
Reunión de lanzamiento y planeación	07/10/2024	Se establecen objetivos, alcance, roles, requisitos del cliente y plan de acción. Se asignan roles, se delinean hitos clave, se identifican riesgos y se establecen estrategias de mitigación.
Reunión de ideación	08/10/2024	Se generan ideas creativas y soluciones para el análisis de los textos. Se fomenta la participación abierta y la colaboración entre los asistentes. Se establecen los siguientes pasos para desarrollar y evaluar las ideas.
Reunión de seguimiento	09/10/2024	Se realiza el seguimiento del proyecto, revisando el progreso, identificando desafíos y estableciendo medidas correctivas. Se asignan responsabilidades para la etapa final y se programa la próxima reunión de finalización.
Reunión de finalización	08/09/2024	Se evalúa la finalización del proyecto, revisando logros y objetivos. Se analizan lecciones aprendidas, se destacan puntos fuertes y áreas de mejora, y se planifica la entrega del resultado.

## 5. Enlace del Video

El enlace del video en el Padlet es el siguiente: <https://youtu.be/BtgGqRjJCRk>

## 6. Referencias

[1]Awan, A.A. & Naviani, A. (2023). Naive Bayes Classification Tutorial using Scikit-learn. DataCamp. Disponible en: <https://www.datacamp.com/tutorial/naive-bayes-scikit-learn>

[2]Geeks for Geeks. (2024). Logistic Regression in Machine Learning. Disponible en: <https://www.geeksforgeeks.org/understanding-logistic-regression/>

[3]Shafi, A. (2023). Random Forest Classification with Scikit-Learn. DataCamp. Disponible en: <https://www.datacamp.com/tutorial/random-forests-classifier-python>