Proyecto 1 Inteligencia de Negocios

Problematica

La Organización de las Naciones Unidas (ONU) adopta, el 25 de septiembre del año 2015, la Agenda 2030i para el desarrollo sostenible, cuyo fin es reducir la pobreza, garantizar acceso a la salud y educación, buscar igualdad de género y oportunidades, disminuir el impacto ambiental, entre otros. Esta agenda se basa en 17 objetivos de desarrollo sostenibleii (ODS) y 169 metas (derivadas de los diferentes ODS).

1. Analisís

Oportunidades de Negocio:

Eficiencia Operativa: La automatización de la clasificación de solicitudes puede aumentar la eficiencia al reducir la carga de trabajo manual. Esto puede permitir a tu ONU procesar un mayor volumen de solicitudes en menos tiempo.

Toma de Decisiones Más Rápida: Al categorizar rápidamente las solicitudes en función de los ODS correspondientes, tu organización puede tomar decisiones más informadas y oportunas sobre cómo responder a cada solicitud.

Seguimiento y Reporte de ODS: La clasificación automatizada facilita el seguimiento y la presentación de informes sobre el cumplimiento de los ODS. Esto puede ser esencial para demostrar el compromiso de tu organización con la sostenibilidad y la responsabilidad social.

Reducción de Errores Humanos: La automatización reduce la posibilidad de errores humanos en la clasificación, lo que es especialmente importante cuando se trata de asignar solicitudes a los ODS correctos.

Problemas de Negocio:

Precisión de la Clasificación: La automatización de la clasificación debe ser precisa. Si el modelo o algoritmo utilizado no clasifica correctamente las solicitudes en función de los ODS, esto puede llevar a problemas de cumplimiento o a respuestas inadecuadas.

Necesidad de Datos de Calidad: La calidad de los datos es fundamental para la automatización. Si los datos de entrada no son precisos o están desactualizados, la clasificación automatizada puede ser ineficaz.

Integración Tecnológica: La implementación de sistemas de automatización puede requerir la integración con los sistemas existentes. Esto puede ser un desafío.

Privacidad y Seguridad de los Datos: La automatización de solicitudes implica el manejo de datos sensibles. Garantizar la privacidad y la seguridad de estos datos es esencial y puede ser un desafío.

Organizacion y roles beneficiados:

Gobierno Local o Nacional-Rol de Departamento de Desarrollo Sostenible: Los gobiernos pueden usar la automatización para gestionar y clasificar solicitudes relacionadas con iniciativas de desarrollo sostenible en su jurisdicción. Los departamentos de desarrollo sostenible pueden beneficiarse al agilizar el procesamiento de solicitudes y el seguimiento de proyectos.

Empresas con Iniciativas de Responsabilidad Social Corporativa (RSC)-Rol de RSC o Sostenibilidad: Las empresas que tienen iniciativas de RSC o sostenibilidad pueden usar la automatización para clasificar las solicitudes de patrocinio, donaciones o proyectos sociales. Los equipos de RSC o sostenibilidad pueden gestionar estas solicitudes de manera eficiente.

Instituciones Educativas-Rol de Departamento de Becas o Ayudas: Las instituciones educativas pueden utilizar la automatización para clasificar solicitudes de becas o ayudas relacionadas con los ODS. El personal del departamento de becas o ayudas puede procesar estas solicitudes de manera más eficiente.

ONGs y Agencias de Ayuda Humanitaria-Rol de Gestión de Proyectos de Ayuda: Las organizaciones no gubernamentales y las agencias de ayuda humanitaria pueden automatizar la clasificación de solicitudes de ayuda y proyectos relacionados con los ODS. Los equipos de gestión de proyectos de ayuda pueden beneficiarse al simplificar la gestión de solicitudes.

Inversionistas y Fondos de Inversión Socialmente Responsables (SRI)- Rol de Evaluación de Proyectos de Inversión: Los inversionistas y los fondos de inversión SRI pueden utilizar la automatización para evaluar solicitudes de inversión en proyectos alineados con los ODS. Los equipos de evaluación de proyectos pueden acelerar el proceso de toma de decisiones. Emprendedores Sociales:

Grupo Estadistica:

s.saiz@uniandes.edu.co Zoom 21 – 10 - 2023

1.1 Analisís del negocio

Objetivos del negocio

Para el negocio debemos tener en cuenta los objetivos de desarrollo sostenible planteados. Los cuales son los siguientes:

- Garantizar la disponibilidad de agua y su gestión sostenible y el saneamiento para todos.
- Garantizar el acceso a una energía asequible, segura, sostenible y moderna.
- Promover sociedades justas, pacíficas e inclusivas.

Estos objetivos se hacen para definir metas que al cumplirlas mejorarán la calidad de vida de todos los habitantes del mundo. Teniendo en cuenta los OBS el negocio tiene un objetivo claro:

- Poder analizar automaticamente informacion textual recopilada sobre la opinion de habitantes sobre problematicas locales, teniendo en cuenta los ODS se hará enfasis en las opiniones relacionadas con agua, violencia y energia.

UNFPA es una organización que tiene como objetivo principal la ejecuccion de los ODS. Este proceso requiere de meticula coordinación, planeación y seguimiento. El analisis de información relacionada con estos procesos es necesario para asegurar que los recursos de la organización se gasten eficientemente y puedan cumplir sus objetivos.

Que haremos?

Segun lo analizado anteriormente, creemos que un modelo de clasificación es el primer paso para lograr un analisís eficiente de la información relacionada con los ODS. Pues, una vez conocidos el ODS al que esta relacionado un texto podemos concentrar mas el analisis de la información relacionada con ese ODS.

1.2 Enfoque análitico

Despues de explorar los datos y ver lo que el negocio desea, se decidió que el siguiente enfoque análitico para resolver el problema:

1. **Tipo de aprendizaje:** Supervisado, **Tarea de aprendizaje:** Clasificación **Técnica de aprendizaje:** Regresión Logística **Algoritmo:** Tfidf (Term Frequency-Inverse Document Frequency). (Implementado por **Juan Coronel**)
2. **Tipo de aprendizaje:** Supervisado, **Tarea de aprendizaje:** Clasificación **Técnica de aprendizaje:** Regresión Logística **Algoritmo:** Bag of words. (Implementado por **Julian Villate**)
3. **Tipo de aprendizaje:** Supervisado, **Tarea de aprendizaje:** Clasificación **Técnica de aprendizaje:** Regresión Logística **Algoritmo:** BERT. (Implementado por **Andres Parraga**)

Esta decisión es la decisión apropiada ya que los principales objetivos del proyecto son:

Desarrollar un modelo de clasificación, con técnicas de aprendizaje automático, que permita relacionar de manera automática un texto según los ODS.

2. Entendimiento de datos

2.1 Perfilamiento de datos

2.1.1 Importación de librerias

```
!pip install nltk
!pip install spacy
!python -m spacy download es
!pip install unidecode
```

```
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy) (67.7.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.3.0)
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spac
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (20
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy) (0.7.11
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy) (
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0->spacy) (8.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy) (2.1.3)
2023-10-15 18:41:51.917087: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-10-15 18:41:52.933213: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2023-10-15 18:41:54.263306: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from Sy
2023-10-15 18:41:54.263788: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from Sy
2023-10-15 18:41:54.263966: I tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:995] successful NUMA node read from Sy
⚠ As of spaCy v3.0, shortcuts like 'es' are deprecated. Please use the
full pipeline package name 'es_core_news_sm' instead.
Collecting es-core-news-sm==3.6.0
  Downloading https://github.com/explosion/spacy-models/releases/download/es_core_news_sm-3.6.0/es_core_news_sm-3.6.0-py3-none-any.whl
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.9/12.9 MB 30.0 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.7.0,>=3.6.0 in /usr/local/lib/python3.10/dist-packages (from es-core-news-sm==3.6.0) (3.6.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-co
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-co
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-new
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-n
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-sm==3
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-sm==3
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-n
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-sm==3.6.0) (3
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-sm==3.6.0
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-news-sm==
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy<3.7.0,>=3.6.0->es-core-n
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spac
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.7.0,>=3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.7
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy<3.7
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy<3.7.0,>=
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy<3
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0->spacy<3.7.0,
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy<3.7.0,>=3.6.0->es-core-n
Installing collected packages: es-core-news-sm
Successfully installed es-core-news-sm-3.6.0
✓ Download and installation successful
You can now load the package via spacy.load('es_core_news_sm')
```

```
# librería Natural Language Toolkit, usada para trabajar con textos
import nltk
import numpy as np

# librerías bases
import pandas as pd
import matplotlib.pyplot as plt
from unidecode import unidecode
from sklearn.model_selection import train_test_split
import seaborn as sns
# Librerias para hacer el entrenamiento y la regresion logistica con parametros adecuados
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# librería para lematizar los textos en español y realizar flexiones gramaticales
```

```
# Nota: Un lema es la forma base de una palabra.
import spacy

# librería para limpiar el texto con expresiones regulares, regex de python
import re

# Punkt nos permite separar un texto en frases.
nltk.download('punkt')

# Descargamos todas las palabras vacias, es decir, aquellas que no aportan nada al significado del texto
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
import spacy

# Carga el modelo en español
nlp = spacy.load("es_core_news_sm")

# Texto que deseas analizar
texto = "Los gatos están saltando sobre las cajas."

# Procesa el texto
doc = nlp(texto)

# Itera a través de las palabras del texto y obtén información gramatical
for token in doc:
    print(f"Palabra: {token.text}, POS: {token.pos_}, Flexión gramatical: {token.morph}")
```

```
Palabra: Los, POS: DET, Flexión gramatical: Definite=Def|Gender=Masc|Number=Plur|PronType=Art
Palabra: gatos, POS: NOUN, Flexión gramatical: Gender=Masc|Number=Plur
Palabra: están, POS: AUX, Flexión gramatical: Mood=Ind|Number=Plur|Person=3|Tense=Pres|VerbForm=Fin
Palabra: saltando, POS: VERB, Flexión gramatical: VerbForm=Ger
Palabra: sobre, POS: ADP, Flexión gramatical:
Palabra: las, POS: DET, Flexión gramatical: Definite=Def|Gender=Fem|Number=Plur|PronType=Art
Palabra: cajas, POS: NOUN, Flexión gramatical: Gender=Fem|Number=Plur
Palabra: ., POS: PUNCT, Flexión gramatical: PunctType=Peri
```

2.1.2 Lectura de los datos

Note que efecitvamente al cargar los datos esperamos un dataframe con 3000 filas y 2 columnas (texto, sdg)

```
# Usamos la libreria pandas para leer el archivo excel
data = pd.read_excel("./data/cat_6716.xlsx")
data.shape
```

```
(3000, 2)
```

- Se muestra un ejemplo de los datos presentes en el dataframe:

```
data.head()
```

	Textos_espanol	sdg
0	Es importante destacar que, en un año de sequí...	6
1	Hay una gran cantidad de literatura sobre Aust...	6
2	Los procesos de descentralización, emprendidos...	6
3	Esto puede tener consecuencias sustanciales pa...	6
4	La función de beneficio también incorpora pará...	6

- Se hace una descripción de los datos para destacar las variables categóricas o numéricas:

```
data.dtypes
```

```
Textos_espanol    object
sdg                int64
dtype: object
```

Note que en efecto todas las sentencias, como se espera, son strings y son clasificadas en la etiqueta numerica sdg.

- Describamos la variable numerica, sdg, con sus caracteristicas estadisticas para tener una observacion mas profunda de esta variable:

```
data['sdg'].describe()
```

```
count    3000.000000
mean        9.666667
std         4.497662
min         6.000000
25%         6.000000
50%         7.000000
75%        16.000000
max        16.000000
Name: sdg, dtype: float64
```

Note que la etiqueta mas presente en el archivo es 16 y la menos presente es 6, informacion muy importante a tener en cuenta mas adelante.

- Para entender las sentencias, se realizan 'estadísticas descriptivas' para poder obtener un análisis de los textos:

```
# Número de palabras por oración
data['Num_Palabras'] = data['Textos_espanol'].apply(lambda text: len(text.split(' ')))
print("Número de palabras por oración:")
print(data['Num_Palabras'])

# Longitud promedio de las oraciones
promedio_longitud = data['Num_Palabras'].mean()
print(f"\nLongitud promedio (palabras) de las oraciones: {round(promedio_longitud, 2)}")

# Longitud máxima y mínima de las oraciones
longitud_maxima = data['Num_Palabras'].max()
longitud_minima = data['Num_Palabras'].min()
print(f"Longitud máxima (palabras) de las oraciones: {round(longitud_maxima, 2)}")
print(f"Longitud mínima (palabras) de las oraciones: {longitud_minima}")

# Eliminar la columna 'Num_Palabras'
data = data.drop(columns=['Num_Palabras'])
```

```
Número de palabras por oración:
0       103
1       142
2        89
3       116
4       111
       ...
2995    159
2996    140
2997    122
2998    151
2999    102
Name: Num_Palabras, Length: 3000, dtype: int64

Longitud promedio (palabras) de las oraciones: 119.54
Longitud máxima (palabras) de las oraciones: 266
Longitud mínima (palabras) de las oraciones: 24
```

2.2 Calidad de datos

2.2.1 Unicidad

Verifiquemos el numero de filas duplicadas, es decir, el numero de filas que tienen exactamente los mismos valores en las mismas columnas

```
data.duplicated().sum()
```

```
0
```

¡Excelente! no hay filas duplicadas, por lo cual no tenemos necesidad de tomar medidas al respecto.

2.2.2 Completitud

Note que la completitud se muestra en porcentaje de valores no nulos, es decir, una completitud de 100% indica que no hay ningun valor nulo en esa columna

```
completitud = data.count() / len(data) * 100
print(completitud)
```

```
Textos_espanol    100.0
sdg               100.0
dtype: float64
```

¡Excelente! no hay valores nulos, todas las filas estan completas, por lo cual no tenemos necesidad de tomar medidas al respecto. Lo cual tiene mucho sentido, porque si son textos no podemos tener una fila sin ninguna palabra o sin etiqueta.

2.2.3 Consistencia

Hay muchos errores de inconsitencia en los datos puesto que el Negocio ha realizado una toma de datos con valores mal formateados. Por ejemplo, hay muchos textos que en las tildes el formato esta mal declarado y aparece asi: "provocarÃ¡ una reducciÃ³n del caudal de los rÃ\xados que estÃ¡n". Ademas, tambien hay muchas palabras que tienen muchos codigos y elementos que no tienen sentido ni aportan valor, como por ejemplo: "El Tribunal de ApelaciÃ\x83Â³n de InmigraciÃ\x83Â³n". Finalmente, hay muchas sentencias en idioma ingles, se esperaba que todas las sentencias esten en español.

- Decisiones tomadas:
  - Dado que poner un encoding no sirve para arreglar estos errores, se pueden solucionar manualmente ya que por ejemplo Ã³ representa una tilde en la o, Ã¡ representa una tilde en la a y asi sucesivamente.
  - Las palabras que tienen muchos elementos o simbolos desconocidos seran eliminadas o no tenidas en cuenta para el modelo
  - Para las sentencias en ingles se ha tomado la decision de DECISIOOOOOOOOOON INGLEEEES.

2.2.4 Validez

Miremos que tan validos son los valores en la columnas, para eso, tenemos que imprimir los valores unicos de cada columna y comprobar que resultados atipicos obtenemos

```
# Mostrar los valores únicos de las columnas categóricas
for col in data.select_dtypes(include=['object']):
    print(f"Valores únicos de {col}: {data[col].unique()}")

# Mostrar el rango de valores de las columnas numéricas
for col in data.select_dtypes(include=['float64', 'int64']):
    print(f"\nRango de valores de {col}: [{data[col].min()}, {data[col].max()}]")
```

```
Valores únicos de Textos_espanol: ['Es importante destacar que, en un año de sequía, se espera que disminuyan todos los aportes, pero qu
 'Hay una gran cantidad de literatura sobre Australia en esta área que sugiere fuertemente que el comercio de agua, especialmente en los
 'Los procesos de descentralización, emprendidos para una serie de actividades que antes realizaba el gobierno central, llevaron a modif
 ...
 'En este libro, el autor argumenta que el activismo judicial dirigido a la protección de los derechos humanos y la dignidad y el derech
 '* Facultad de Derecho, Universidad de Manchester. 1 SW v Secretary of State for the Home Department (Adjudicatorâ€™s questions) Somali
 'Este caso 1 constituye el primer pronunciamiento de la Corte Internacional de Justicia (CIJ) sobre la obligación de extraditar o juzga

Rango de valores de sdg: [6, 16]
```

Note que no hay errores de invalidez puesto que el rango de valores esperados para las etiquetas es correcto, no se sale de lo definido en el proyecto y todos los textos son strings.

2.3 Limpieza y preparacion de datos

Es necesario que limpiemos y preparemos los datos antes de comenzar con el modelo. Para empezar, porque no todos los datos vienen en un formato perfecto, si no que tenemos que considerar el idioma y los errores del texto para asi modificarlo y dejarlo ideal para ingresarlo en el modelo.

2.3.1 Limpieza de datos

- **Cambios en consistencia:** Vamos a cambiar el error en las tildes antes mencionada de manera manual para no perder tantas palabras que pueden representar un determinismo muy importante en el modelo:

```
# Primero miremos como esta una fila antes de los cambios a realizar
valor = data.at[293, 'Textos_espanol']
print(valor)
```

```
Los acuerdos y arreglos varÃan en tÃ©rminos de cobertura geogrÃ¡fica -abarcando todas las aguas compartidas por las partes contratantes
```

```
# Ahora hagamos el cambio manual de las tildes en todas las filas y miremos el resultado

data['Textos_espanol'] = data['Textos_espanol'].str.replace('Ã¡', 'a')
data['Textos_espanol'] = data['Textos_espanol'].str.replace('Ã©', 'e')
data['Textos_espanol'] = data['Textos_espanol'].str.replace('Ã³', 'o')
# Tambien realicemos un cambio para que se pueda detectar la ñ
data['Textos_espanol'] = data['Textos_espanol'].str.replace('Ã±', 'ñ')
data['Textos_espanol'] = data['Textos_espanol'].str.replace('Ã', 'i')
```

```
# Ahora veamos como esta la fila actualizada
valor = data.at[293, 'Textos_espanol']
print(valor)
```

```
Los acuerdos y arreglos varian en terminos de cobertura geografica -abarcando todas las aguas compartidas por las partes contratantes o
```

```
# Ademas, podemos evidenciar que el texto ya no contiene el caracter 'Ã'
resultados = data[data['Textos_espanol'].str.contains('Ã', case=False, na=False)].drop(columns=['sdg'])
resultados
```

	Textos_espanol

- **conversion a texto plano:** es necesario eliminar los caracteres no deseados y darle un buen formato a los textos encontrados para que el modelo puedo aproximarse de mejor manera al resultado deseado. Para esto, haremos uso de la libreria python regular expresion ***(regex)***.

```
# Definimos el preprocessador para transformar los textos y limpiarlos.
def preprocessor(texto):

    # Primero convertimos todas las palabras a lowercase y quitamos los simbolos
    texto = re.sub('[\W]+', ' ', texto.lower())

    # Reemplazamos la "ñ" con un marcador temporal para que se tenga en cuenta y no se borre
    texto = texto.replace("ñ", "TEMP_N")

    # Eliminamos tildes o acentos de las letras
    texto = unidecode(texto)

    # Restauramos la "ñ" desde el marcador temporal
    texto = texto.replace("TEMP_N", "ñ")

    # Reemplazar los valores numéricos por un token específico (por ejemplo, "[NUM]")
    texto = re.sub(r'\d+(\.\d+)?', '[NUM]', texto)

    # Ahora quitamos las stop words que nos otorga la libreria spacy en español
    doc = nlp(texto)
    texto = ' '.join(token.text for token in doc if not token.is_stop)
```

```
    return texto

print(preprocessor('Esto -es *** (una) "!pRiíí" ñPRuebA!)) :) +) ????*^&(#$%%!@#!@#^&*^()) de como eliminar stop words en español. 123123 131

     priii ñprueba eliminar stop words español [ NUM ] [ NUM ] [ NUM ] [ NUM ] [ NUM ]
```

- Ahora aplicamos las funciones del preprocessador a todos los textos del dataframe:

```
# Aplicamos la eliminacion del ruido
data['Textos_espanol'] = data['Textos_espanol'].apply(preprocessor)


data.head(5)
```

	Textos_espanol	sdg
0	importante destacar año sequia espera disminuy...	6
1	cantidad literatura australia area sugiere fue...	6
2	procesos descentralizacion emprendidos serie a...	6
3	consecuencias sustanciales calidad agua especi...	6
4	funcion beneficio incorpora parametros afectan...	6

2.3.2 Tokenización

La tokenización permite dividir frases u oraciones en palabras. Con el fin de desglozar las palabras correctamente para el posterior análisis.

La parte mas importante, la **tokenizacion** la realizaremos con la libreria spacy puesto esta en español y nos ayudara a enlistar las palabras que encontramos en cada texto. Tambien, es necesario hacer la **lematizacion** para asi poder llevar a todas las palabras a su lema, es decir a su forma inicial y asi eliminar el ruido que pueden generar diversas formas en la palabra que tienen el mismo significado. Todo esto sera realizado automaticamente en la funcion tokenizar ya que cada token sera el lemma de la palabra:

```
# Función para tokenizar un texto y devolver una lista de tokens
def tokenizar(texto):
    doc = nlp(texto)
    tokens = [token.lemma_ for token in doc]
    return tokens

# Aplicamos la función de tokenización a cada fila del DataFrame y agregar una nueva columna
data['words'] = data['Textos_espanol'].apply(tokenizar)
data.head(5)
```

	Textos_espanol	sdg	words
0	importante destacar año sequia espera disminuy...	6	[importante, destacar, año, sequia, esperar, d...
1	cantidad literatura australia area sugiere fue...	6	[cantidad, literatura, australio, areo, sugeri...
2	procesos descentralizacion emprendidos serie a...	6	[proceso, descentralizacion, emprendido, serie...
3	consecuencias sustanciales calidad agua especi...	6	[consecuencia, sustancial, calidad, agua, espe...
4	funcion beneficio incorpora parametros afectan...	6	[funcion, beneficio, incorporar, parametro, af...

3. Implementacion del modelo (Tfidf (Term Frequency-Inverse Document Frequency))

El algoritmo TF-IDF (Term Frequency-Inverse Document Frequency) es una técnica de procesamiento de lenguaje natural que se utiliza comúnmente para evaluar la importancia relativa de una palabra o término en un documento dentro de un corpus de documentos.

3.1 Variable objetivo

```
# Creación de variable objetivo (etiqueta).
Y = data['sdg']
```

```
# Creacion de las variables de entrada
X = data.drop(['sdg'], axis=1)

num_muestras = data.shape[1]
num_muestras
```

```
3
```

3.2 Entrenamiento

- Ahora usaremos un objeto de GridSearchCV para encontrar el conjunto de parametros optimo para nuestro modelo de regresion logistica usango 5-fold stratified cross-validation. Esto es necesario ya que asi no tendremos un modelo estatico si no un modelo que se entrene con mas divisiones de datos y no este sobre ajustado en un solo conjunto de datos en el entrenamiento.

```
# Dividimos el conjunto de datos
X_train = data.loc[:2400, 'Textos_espanol'].values
y_train = data.loc[:2400, 'sdg'].values
X_test = data.loc[2400:, 'Textos_espanol'].values
y_test = data.loc[2400:, 'sdg'].values

# El algoritmo utiliza una clasificacion del tipo tfidf, asi que inicializamos el objeto
tfidf = TfidfVectorizer(strip_accents=None,
                        lowercase=False,
                        preprocessor=None)

# Usamos el tokenizer de spacy en español
def spacy_tokenizer(text):
    tokens = nlp(text)
    return [token.text for token in tokens if not token.is_stop]


# Definimos el diccionario de parametros para la busqueda en el gridsearchCV
param_grid = [
      {'vect__ngram_range': [(1,1)],
                'vect__tokenizer': [spacy_tokenizer],
                'clf__penalty': ['l1', 'l2'],
                'clf__C': [1.0, 10.0, 100.0]},
               {'vect__ngram_range': [(1,1)],
                'vect__tokenizer': [spacy_tokenizer],
                'vect__use_idf':[False],
                'vect__norm':[None],
                'clf__penalty': ['l1', 'l2'],
               'clf__C': [1.0, 10.0, 100.0]}

]

lr_tfidf = Pipeline([('vect', tfidf),
                     ('clf',
                      LogisticRegression(random_state=0,
                                         solver='liblinear'))])

gs_lr_tfidf = GridSearchCV(lr_tfidf, param_grid,
                           scoring='accuracy',
                           cv=5, verbose=2,
                           n_jobs=1)

gs_lr_tfidf.fit(X_train, y_train)
```

```
[CV] END clf__C=1.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=10.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vect
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vec
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vec
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vec
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vec
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l1, vect__ngram_range=(1, 1), vect__norm=None, vec
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWa
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, vec
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserW
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, ve
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserW
  warnings.warn(
[CV] END clf__C=100.0, clf__penalty=l2, vect__ngram_range=(1, 1), vect__norm=None, ve
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserW
```

- Ademas, en el primer diccinoario estamos utsando TfidVectorizer con sus parametros por defecto para calcular el tf-idfs; en el segundo diccionario estamos usando los parametros use_idf=False, smooth_idf=False, y norm=None para entrenar un modelo basado en frecuencias de terminos brutos.
- Finalmente, considere que tambien usamos dos tipos de normalizacion para comparar la fuerza con la que la regresion logistica normaliza los vectores
- Ahora, imprimimos el mejor conjunto de parametros que encontramos con el GridSearchCV:

```
print('Best parameter set: %s ' % gs_lr_tfidf.best_params_)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-20-917d0f718b14> in <cell line: 1>()
----> 1 print('Best parameter set: %s ' % gs_lr_tfidf.best_params_)

NameError: name 'gs_lr_tfidf' is not defined
```

BUSCAR EN STACK OVERFLOW

- Note que los mejores hiperparametros son:
  - C = 100, es decir que tan fuerte es la regularizacion del modelo, en este caso, un valor de 100 puntos significa que el modelo tolera muchos errores en el conjunto de entrenamiento, pero puede tener un sobreajuste que se vera representado en el conjunto de prueba
  - penalty = l2, parece que la normalizacion l2 es la que mejor penaliza los vectores en sus normas generando un modelo mas aproximado a la solucion verdadera
  - ngram = 1, es decir, que la busqueda nos indica que el mejor valor para cortar las palabras es 1, osea, que usaremos palabras individuales como tokens en vez de conjuntos de palabras
- Ahora imprimimos los scores del cross validation 5-fold:

```
print('CV Accuracy: %.3f'
      % gs_lr_tfidf.best_score_)

clf = gs_lr_tfidf.best_estimator_
print('Test Accuracy: %.3f'
       % clf.score(X_test, y_test))
```

Esto indica que nuestro modelo de machine learning puede clasificar a que etiqueta pertenece el texto con una precision del 99%.

- Ahora, podemos poner a entrenar el modelo con los parametros ideales que hemos encontrado:

```
best_params = {
    'vect': {
        'ngram_range': (1, 1),
        'tokenizer': spacy_tokenizer,
    },
    'clf': {
        'C': 100.0,
        'penalty': 'l2',
    }
}

# Creamos un nuevo pipeline con TfidfVectorizer y LogisticRegression con los mejores parámetros
best_lr_tfidf = Pipeline([
    ('vect', TfidfVectorizer(**best_params['vect'])),  # Configura el vectorizador con los mejores parámetros
    ('clf', LogisticRegression(**best_params['clf']))  # Configura el clasificador con los mejores parámetros
])

# Ajustamos el nuevo pipeline a los datos de entrenamiento originales
best_lr_tfidf.fit(X_train, y_train)
```

```
# Usamos el modelo para hacer predicciones en tus datos de prueba y asignar etiquetas
y_pred = best_lr_tfidf.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be u
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

- Ahora probemos que tal es el resultado obtenido para los datos de prueba no etiquetados:

```
# Cargamos el nuevo archivo CSV que contiene solo la columna 'Textos_espanol'
nuevo_data = pd.read_excel('./data/SinEtiquetatest_cat_6716.xlsx')

# Limpiamos y preparamos el nuevo data set
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(preprocessor)
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã¡', 'a')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã©', 'e')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã³', 'o')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã±', 'ñ')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã', 'i')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(tokenizar)
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(lambda x: ' '.join(map(str, x)))

X_nuevo = nuevo_data['Textos_espanol'].values

# Realizamos las predicciones en el nuevo archivo
y_pred_nuevo = best_lr_tfidf.predict(X_nuevo)

nuevo_data['sdg_predict'] = y_pred_nuevo
nuevo_data.drop(columns=['sdg'], inplace=True)

# Guardamos el nuevo resultado etiquetado en un archivo
nuevo_data.to_csv('etiquetas_predichas_Tfidf.csv', index=False)
nuevo_data
```

	Textos_espanol	sdg_predict
0	[NUM] [NUM] introduccion estructura derech...	16
1	agua subterranea debatido contexto tarificacio...	6
2	presente contribucion evalua jurisprudencio tr...	16
3	creditir fiscal expirar [NUM] wemau [NUM]a ...	7
4	estudio explorar actitud comportamiento percep...	16
...	...	...
975	articulo explorar historia impacto accion afir...	16
976	enfasis manipulacion precio financiacion merca...	7
977	innovacion importante garantizar soporte efect...	7
978	salvador continuo luchar nivel elevado violenc...	16
979	reflejar bajo conciencia papel jugar recurso a...	6

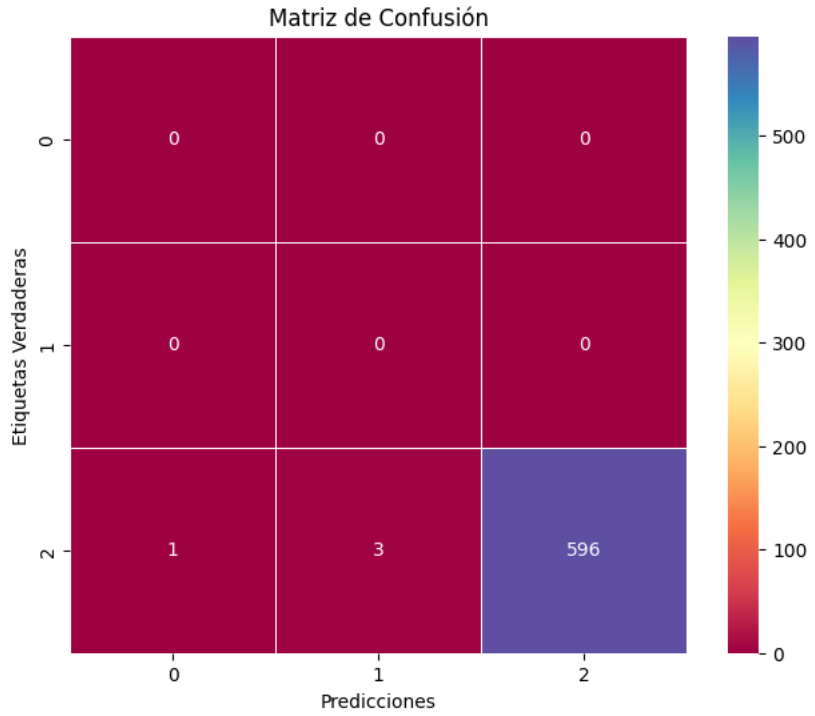
980 rows × 2 columns

Miremos la matriz de confusion para analizar que tal fue el desempeño del modelo para los falsos negativos y los positivos verdaderos:

```
cm = confusion_matrix(y_test, y_pred)

# Creamos una visualización de la matriz de confusión
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Spectral", linewidths=.5, square=True, cbar=True)
plt.xlabel('Predicciones')
plt.ylabel('Etiquetas Verdaderas')
```

```
plt.title('Matriz de Confusión')
plt.show()
```



Finalmente observemos la relacion armonica entre la precision y la exactitud de nuestro modelo atraves de el puntaje f1-score:

```
# Calcula el F1-score
f1 = f1_score(y_test, y_pred, average='weighted')
print("F1-score:", f1)
```

```
F1-score: 0.9966555183946489
```

4. Implementacion del modelo (bag of words)

Bag of words consiste en representar un texto como una colección desordenada de palabras, ignorando el orden y la estructura gramatical. Se crea un "saco de palabras" que contiene todas las palabras únicas en un corpus de texto y se cuentan cuántas veces aparece cada palabra en un documento específico. Esta representación se utiliza para construir características numéricas que alimentan algoritmos de machine learning, permitiendo analizar y comparar textos en función de la frecuencia de palabras. A pesar de su simplicidad, el BoW es efectivo en tareas como la clasificación de documentos y la recuperación de información.
En este caso vamos a utilziar bag of words junto con el clasificador Naive Bayes. Ya estos dos funcionan muy bien juntos. El clasificador Naive Bayes es un algoritmo de aprendizaje supervisado que se basa en el teorema de Bayes para realizar tareas de clasificación. A pesar de su simplicidad, a menudo funciona muy bien en una variedad de aplicaciones, como la clasificación de texto, detección de spam, diagnóstico médico y más.

4.1 Variable objetivo

Vamos a usar la misma variable objetivo y de entrada que en el modelo anterior ya que lo que buscamos es clasificar los textos segun el ODS

```
Y = data['sdg']

X = data.drop(['sdg'], axis=1)

num_muestras = data.shape[0]
print(num_muestras)
```

```
3000
```

4.2 Transformacion de las palabras en vectores

```
X_train = data.loc[:2400, 'Textos_espanol'].values
y_train = data.loc[:2400, 'sdg'].values
X_test = data.loc[2400:, 'Textos_espanol'].values
y_test = data.loc[2400:, 'sdg'].values

# Crear un vectorizador
count = CountVectorizer()

# Ajustar y transformar los datos de entrenamiento
train_bag = count.fit_transform(X_train)

# Transformar los datos de prueba
test_bag = count.transform(X_test)

print(train_bag.shape, y_train.shape)
```

```
(2401, 16381) (2401,)
```

Se dividieron los datos de esta forma ya que la tradicional suele dar problemas por las dimensiones de los datos

4.3 Entrenamiento

Para Naive Bayes se utilizara una distribución multinomial, ya que es la que mejor encaja con nuestra distribucion de datos. Y el aplha en 1 para que las palabras poco frecuentes no afecten el modelo.

```
# Crear y entrenar el modelo
bagmodel = MultinomialNB(alpha=1.0)
bagmodel.fit(train_bag, y_train)

# Realizar predicciones
y_pred = bagmodel.predict(test_bag)
```

4.4 Resultados y metricas

```
accuracy = accuracy_score(y_test, y_pred)
print("Precisión del modelo:", accuracy)
```

```
Precisión del modelo: 0.9933333333333333
```

```
precision = precision_score(y_test, y_pred, average='weighted')
print(f"Precisión (Precision): {precision:.2f}")
```

```
Precisión (Precision): 1.00
```

Una alta precisión significa que la mayoría de las predicciones positivas son correctas.

```
recall = recall_score(y_test, y_pred, average='weighted')
print(f"Recuperación (Recall): {recall:.2f}")
```

```
Recuperación (Recall): 0.99
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall is ill-defined and being
  _warn_prf(average, modifier, msg_start, len(result))
```

Una alta recuperación significa que el modelo puede encontrar la mayoría de los ejemplos positivos en el conjunto de datos.

```
f1 = f1_score(y_test, y_pred, average='weighted')
print(f"Puntuación F1 (F1 Score): {f1:.2f}")
```

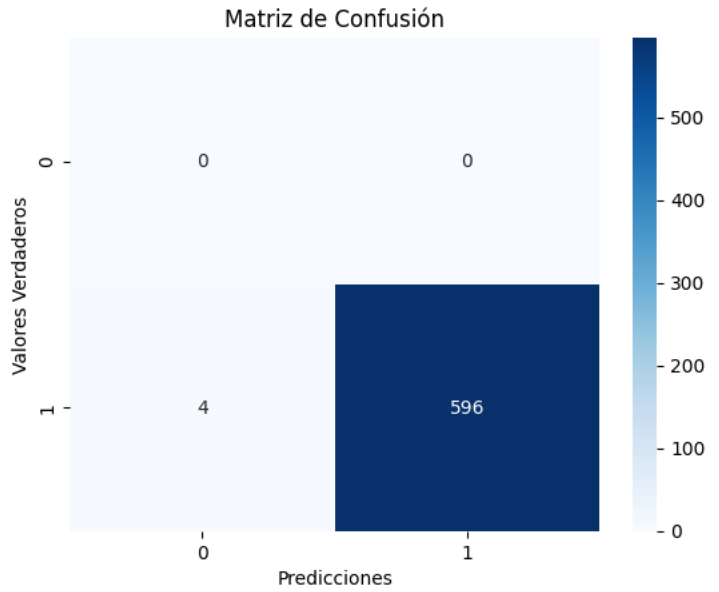
```
Puntuación F1 (F1 Score): 1.00
```

un puntaje F1 alto refleja un buen equilibrio entre la capacidad del modelo para hacer predicciones precisas y su capacidad para encontrar ejemplos positivos en el conjunto de datos, lo que lo convierte en una métrica valiosa para evaluar el rendimiento de un modelo de clasificación.

Como se puede ver el modelo tiene una alta precisión, por lo que se puede considerar util para el negocio.

```
confusion = confusion_matrix(y_test, y_pred)

# Crea una gráfica de matriz de confusión
sns.heatmap(confusion, annot=True, fmt="d", cmap="Blues" ,square=True, cbar=True)
plt.xlabel("Predicciones")
plt.ylabel("Valores Verdaderos")
plt.title("Matriz de Confusión")
plt.show()
```



Aunque una matriz de confusión no es la mejor manera de analizar los resultados, podemos ver que solo hubo 4 predicciones fallidas en el modelo. Lo cual es una muestra de su alta calidad.

4.5 Aplicación a datos sin etiquetar

```
# Cargamos el nuevo archivo CSV que contiene solo la columna 'Textos_espanol'
nuevo_data = pd.read_excel('./data/SinEtiquetatest_cat_6716.xlsx')

# Limpiamos y preparamos el nuevo data set
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(preprocessor)
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã¡', 'a')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã©', 'e')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã³', 'o')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã±', 'ñ')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã', 'i')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(tokenizar)
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(lambda x: ' '.join(map(str, x)))

X_nuevo = nuevo_data['Textos_espanol'].values
unlabeled_bag = count.transform(X_nuevo)
# Realizamos las predicciones en el nuevo archivo
```

```
y_pred_nuevo = bagmodel.predict(unlabeled_bag)

nuevo_data['sdg_predict'] = y_pred_nuevo
nuevo_data.drop(columns=['sdg'], inplace=True)

# Guardamos el nuevo resultado etiquetado en un archivo
nuevo_data.to_csv('etiquetas_predichas_bagofwords.csv', index=False)
nuevo_data
```

	Textos_espanol	sdg_predict
0	[NUM] [NUM] introduccion estructura derech...	16
1	agua subterranea debatido contexto tarificacio...	6
2	presente contribucion evalua jurisprudencio tr...	16
3	creditir fiscal expirar [NUM] wemau [NUM]a ...	7
4	estudio explorar actitud comportamiento percep...	16
...	...	...
975	articulo explorar historia impacto accion afir...	16
976	enfasis manipulacion precio financiacion merca...	7
977	innovacion importante garantizar soporte efect...	7
978	salvador continuo luchar nivel elevado violenc...	16
979	reflejar bajo conciencia papel jugar recurso a...	6

980 rows × 2 columns

5. Implementacion del modelo BERT (Bidirectional Encoder Representations from Transformers)

BERT es una algoritmo implementado para una tarea de pre-entrenamiento bidireccional. Esto significa que BERT puede entender el contexto de una palabra o una frase al observar todas las palabras de la oración simultáneamente, tanto a la izquierda como a la derecha de la palabra en cuestión. Esta capacidad hace que BERT sea extremadamente eficaz para tareas de procesamiento del lenguaje natural (NLP) que implican comprensión del contexto y relación semántica entre palabras y oraciones. Esta capacidad de definir las palabras como elemento polisemicos, nos permite definir representaciones contextuales de palabras. Esto significa que el significado de una palabra puede variar según su contexto en una oración.

5.1 Variable Objetivo:

Se espera realizar la clasificación de textos, de los textos que representan la voz de los habitantes locales sobre problemáticas de su entorno. Este proceso tendra de la finalidad clasificar de modo que permitan hacer un análisis automatizado de opiniones. Para esta finalidad, definimos la etiqueta (que sera asignada) como nuestra variable Objetivo. Estos valores pueden exisitir entre los diferentes valores posibles para sdg (ODS).

```
!pip install transformers
!pip install torch
!pip install unidecode
!pip install nltk
!pip install spacy

# librerÃa Natural Language Toolkit, usada para trabajar con textos
import nltk
import numpy as np

# librerÃas bases
import pandas as pd
import matplotlib.pyplot as plt
from unidecode import unidecode
from sklearn.model_selection import train_test_split
import seaborn as sns
# Librerias para hacer el entrenamiento y la regresion logistica con parametros adecuados
from sklearn.model_selection import GridSearchCV
```

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

# librerÃa para lematizar los textos en espaÃ±ol y realizar flexiones gramaticales
# Nota: Un lema es la forma base de una palabra.
import spacy

# librerÃa para limpiar el texto con expresiones regulares, regex de python
import re

# Punkt nos permite separar un texto en frases.
nltk.download('punkt')

# Descargamos todas las palabras vacias, es decir, aquellas que no aportan nada al significado del texto
nltk.download('stopwords')


import spacy

# Carga el modelo en espaÃ±ol
nlp = spacy.load("es_core_news_sm")

# Texto que deseas analizar
texto = "Los gatos estÃ¡n saltando sobre las cajas."

# Procesa el texto
doc = nlp(texto)

# Itera a travÃ©s de las palabras del texto y obtÃ©n informaciÃ³n gramatical
for token in doc:
    print(f"Palabra: {token.text}, POS: {token.pos_}, FlexiÃ³n gramatical: {token.morph}")

# Usamos la libreria pandas para leer el archivo excel
data = pd.read_excel("./data/cat_6716.xlsx")
data.shape

# Primero miremos como esta una fila antes de los cambios a realizar
valor = data.at[293, 'Textos_espanol']

# Ahora hagamos el cambio manual de las tildes en todas las filas y miremos el resultado

data['Textos_espanol'] = data['Textos_espanol'].str.replace('ÃÂ¡', 'a')
data['Textos_espanol'] = data['Textos_espanol'].str.replace('ÃÂ©', 'e')
data['Textos_espanol'] = data['Textos_espanol'].str.replace('ÃÂ³', 'o')
# Tambien realicemos un cambio para que se pueda detectar la Ã±
data['Textos_espanol'] = data['Textos_espanol'].str.replace('ÃÂ±', 'Ã±')
data['Textos_espanol'] = data['Textos_espanol'].str.replace('Ã', 'i')

# Ahora veamos como esta la fila actualizada
valor = data.at[293, 'Textos_espanol']

# Definimos el preprocessador para transformar los textos y limpiarlos.
def preprocessor(texto):

    # Primero convertimos todas las palabras a lowercase y quitamos los simbolos
    texto = re.sub('[\W]+', ' ', texto.lower())

    # Reemplazamos la "Ã±" con un marcador temporal para que se tenga en cuenta y no se borre
    texto = texto.replace("Ã±", "TEMP_N")

    # Eliminamos tildes o acentos de las letras
    texto = unidecode(texto)

    # Restauramos la "Ã±" desde el marcador temporal
    texto = texto.replace("TEMP_N", "Ã±")

    # Reemplazar los valores numÃ©ricos por un token especÃfico (por ejemplo, "[NUM]")
    texto = re.sub(r'\d+(\.\d+)?', '[NUM]', texto)

    # Ahora quitamos las stop words que nos otorga la libreria spacy en espaÃ±ol
    doc = nlp(texto)
```

```
    texto = ' '.join(token.text for token in doc if not token.is_stop)

    return texto

data['Textos_espanol'] = data['Textos_espanol'].apply(preprocessor)
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2023.7.22)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.0.1+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.12.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.0.0)
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch) (3.27.6)
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch) (17.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: unidecode in /usr/local/lib/python3.10/dist-packages (1.3.7)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packages (3.6.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.2.0,>=8.1.8 in /usr/local/lib/python3.10/dist-packages (from spacy) (8.1.12)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.9.0)
Requirement already satisfied: pathy>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.10.2)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (4.66.1)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.23.5)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.10.13)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from spacy) (67.7.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.3.0)
Requirement already satisfied: typing-extensions>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spac
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (20
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy) (0.7.11
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from thinc<8.2.0,>=8.1.8->spacy) (
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/python3.10/dist-packages (from typer<0.10.0,>=0.3.0->spacy) (8.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->spacy) (2.1.3)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Palabra: Los, POS: DET, FlexiÃ³n gramatical: Definite=Def|Gender=Masc|Number=Plur|PronType=Art
Palabra: gatos, POS: NOUN, FlexiÃ³n gramatical: Gender=Masc|Number=Plur
Palabra: estÃ¡n, POS: AUX, FlexiÃ³n gramatical: Mood=Cnd|Number=Plur|Person=3|VerbForm=Fin
Palabra: saltando, POS: VERB, FlexiÃ³n gramatical: VerbForm=Ger
```

```
from transformers import BertTokenizer, BertForSequenceClassification,DistilBertTokenizer,DistilBertForSequenceClassification
import torch

# Supongamos que el DataFrame tiene dos columnas: 'texto' y 'etiqueta'
textos = data['Textos_espanol'].values
etiquetas = data['sdg'].values
```

5.2 Entrenamiento:

Para el modelamiento con Bert, es necesario escoger el modelo que mejor se adopte a las necesidades de la compañia y los requerimientos de los datos. Es por eso que se tomaron las siguientes decisiones: Modelo: bert - base- uncased Bert: Representa un modelo BERT, por la capacidad de analisis bidireccional es exccelente para el procesamiento de texto natural y capturar el contexto de los mensajes Base: Hace referencia a la arquitectura del modelo. Define el numero de capas y parametros que procesa el modelo. Se selecciono "Base" dado que

contamos con una base de datos con mayor complejidad;, y dado que "large" es una arquitectura más costosa en procesamiento y especializada en labores especificas. Uncased: Nos ayuda con el manejo de mayusculas y minusculas, para que el modelo no haga disitnción entre palabras iguales con diferencias en Mayusculas o Minusculas.

```
# Tokenización utilizando BERT
model_name = 'bert-base-uncased'  # Modelo BERT pre-entrenado
tokenizer = BertTokenizer.from_pretrained(model_name)
```

```
from transformers import BertTokenizer, BertForSequenceClassification,DistilBertTokenizer,DistilBertForSequenceClassification
import torch

# Supongamos que el DataFrame tiene dos columnas: 'texto' y 'etiqueta'
textos = data['Textos_espanol'].values
etiquetas = data['sdg'].values

# TokenizaciÃ³n utilizando BERT
model_name = 'bert-base-uncased'  # Modelo BERT pre-entrenado
tokenizer = BertTokenizer.from_pretrained(model_name)
inputs = tokenizer(textos.tolist(), padding=True, truncation=True, return_tensors='pt')

# Cargar modelo BERT pre-entrenado para clasificaciÃ³n de secuencias
model = BertForSequenceClassification.from_pretrained(model_name)

# Ajuste fino del modelo BERT
with torch.no_grad():
    outputs = model(**inputs)

# Representaciones de caracterÃsticas de BERT
features = outputs.logits

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(features, etiquetas, test_size=0.2, random_state=42)

# Entrenar un modelo de regresiÃ³n logÃstica
modelo_logistico = LogisticRegression(max_iter=1000)
modelo_logistico.fit(X_train, y_train)

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

# Calcular la precisiÃ³n
precision = accuracy_score(y_test, predicciones)
print(f'PrecisiÃ³n del modelo: {precision}')

# Para hacer predicciones en nuevos datos, tokeniza los nuevos textos y utiliza el modelo de regresiÃ³n logÃstica entrenado.
```

```
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initiali
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
tensor([[0.6312, 0.3688]], grad_fn=<SoftmaxBackward0>)
```

Los resultados de el entrenamiento de los datos arrojan un desbordamiento en la capacidad de la memoria RAM. Esto se puede deber dado que BERT utiliza una arquitectura de red neuronal profunda con múltiples capas. Cada capa de la red introduce una cantidad significativa de parámetros y requiere recursos adicionales para el cálculo de los gradientes durante el entrenamiento y la inferencia. Dando como resultado un mayor uso de la memoria.

Como alternativa se propone la implenetacion de TinyBert. Esta es una versión más ligera y compacta del modelo BERT (Bidirectional Encoder Representations from Transformers) que está diseñada para ser más eficiente en términos de consumo de recursos, especialmente en entornos con restricciones de memoria, como dispositivos con recursos limitados o aplicaciones que requieren una carga rápida.

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Tokenización utilizando el tokenizador de TinyBERT
data = pd.read_excel("./data/cat_6716.xlsx")

# Extract text and labels from the DataFrame
```

```
texts = data['Textos_espanol'].values
labels = data['sdg'].values

model_name = 'prajjwal1/bert-tiny'  # Modelo TinyBERT pre-entrenado
tokenizer = BertTokenizer.from_pretrained(model_name)
inputs = tokenizer(textos.tolist(), padding=True, truncation=True, return_tensors='pt')

# Cargar modelo TinyBERT pre-entrenado para clasificación de secuencias
model = BertForSequenceClassification.from_pretrained(model_name)

# Ajuste fino del modelo TinyBERT
with torch.no_grad():
    outputs = model(**inputs)

# Representaciones de características de TinyBERT (puedes usar estas como entrada para el modelo de regresión logística)
features = outputs.logits

# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(features, etiquetas, test_size=0.2, random_state=42)

# Entrenar un modelo de regresión logística
modelo_logistico = LogisticRegression(max_iter=100)
modelo_logistico.fit(X_train, y_train)

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

# Calcular la precisión
precision = accuracy_score(y_test, predicciones)
print(f'Precisión del modelo: {precision}')
```

```
Asking to truncate to max_length but no maximum length is provided and the model has no predefined maximum length. Default to no truncat
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at prajjwal1/bert-tiny and are newly initia
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

5.3 Resultados y metricas:

La precisión obtenida es relativamente baja, con un valor de 0.44. Esto sugiere que el modelo no está funcionando de manera óptima para esta tarea de clasificación. Esto se puede deber principalmente a las capacidades del sistema. Si el modelo BERT es muy grande y los recursos son limitados, puede haber dificultades para entrenar o inferir con precisión. Como lo es en este caso especifico, en el cual tuvimos que realizar un cambio de modelo, sacrificando la precision del modelo más completo/complejo para optar por un modelo con un uso más eficiente de los recursos. Adicionalmente,podemos considerar factores tales como: Los hiperparámetros y la arquitectura del modelo de clasificación o fallos la limpieza y preprocesamiento de los datos.

5.4 Aplicación a datos sin etiquetar:

```
# Cargamos el nuevo archivo CSV que contiene solo la columna 'Textos_espanol'
nuevo_data = pd.read_excel('./data/SinEtiquetatest_cat_6716.xlsx')

# Limpiamos y preparamos el nuevo data set
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(preprocessor)
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã¡', 'a')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã©', 'e')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã³', 'o')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã±', 'ñ')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].str.replace('Ã', 'i')
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(tokenizar)
nuevo_data['Textos_espanol'] = nuevo_data['Textos_espanol'].apply(lambda x: ' '.join(map(str, x)))

X_nuevo = nuevo_data['Textos_espanol'].values
unlabeled_bag = count.transform(X_nuevo)
# Realizamos las predicciones en el nuevo archivo
predicciones = modelo_logistico.predict(unlabeled_bag)

nuevo_data['sdg_predict'] = y_pred_nuevo
nuevo_data.drop(columns=['sdg'], inplace=True)

# Guardamos el nuevo resultado etiquetado en un archivo
```