Proyecto 1 – Etapa 2
Análisis de textos: Objetivos de desarrollo sostenible (API)

Inteligencia de negocios – ISIS3301

Contenido

1 Automatización de la preparación de datos

1.1 División de datos

Para comenzar, se tomó la decisión de repartir los datos en baches de 2400 filas para cada división. Note que, obviamente, se tomaron los datos etiquetados que fueron otorgados por el negocio. Del mismo modo, estos datos fueron usados para entrenar el modelo que posteriormente es guardado en un pipeline para poder ser persistido. A continuación, se presenta como se hizo esta carga y división de datos. Recuerde que todo lo sustentado aquí, hace parte del notebook desarrollado con el nombre de 'Proyecto_1_etapa_2'.

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

# 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
```

1.2 Preprocesador para la transformación de datos

Para tokenizar los datos de entrada, se define la función tokenizer y se tomó la decisión de usar la librería spacy, junto con sus lemas y sus non stop words, puesto que ofrece el idioma español para estos propósitos. La tokenizacion es sumamente importante en este proyecto para el procesamiento de lenguaje natural, ya que divide el texto en unidades significativas, lo que permite el análisis, la normalización y el procesamiento eficiente, además de ser fundamental en diversas aplicaciones de NLP. Finalmente, también se hace el preprocesador para limpiar los datos, aplicar flexiones gramaticales y otros cambios necesarios para poder hacer uso de los datos de entrada.

```
# Definimos la funcion para tokenizar el texto con la libreria en
español spacy
def tokenizar(texto):
    return [token.text for token in nlp(texto) if not token.is_stop]
```

```
# Definimos el preprocessador para transformar los datos antes de
ajustarlos al modelo
def preprocessor(tokens):

    # Funciones de limpieza para preprocesar una lista de tokens
    tokens = [re.sub('[\W]+', ' ', token.lower()) for token in tokens]
    tokens = [unidecode(token) for token in tokens]
    tokens = ['[NUM]' if re.match(r'\d+(\.\d+)?', token) else token for
token in tokens]
    tokens = [token for token in tokens if token not in
nlp.Defaults.stop_words]  # Filtrar stop words

    # Aplicamos las limpiezas adicionales
    for i in range(len(tokens)):
        tokens[i] = tokens[i].replace('Ã¡', 'a')
        tokens[i] = tokens[i].replace('Ã©', 'e')
        tokens[i] = tokens[i].replace('Ã³', 'o')
        tokens[i] = tokens[i].replace('Ã±', 'ñ')
        tokens[i] = tokens[i].replace('Ã', 'i')

    return tokens

# Creamos el pipeline para la preparación de datos
data_prep_pipeline = Pipeline([
    ('preprocessor', FunctionTransformer(func=preprocessor,
validate=False)),
])

# Aplicamos la transformación al conjunto de entrenamiento
X_train_transformed = data_prep_pipeline.fit_transform(X_train)
```

2 Automatización del modelo entrenado

Para automatizar el modelo, que es construido usando el modelo de tf-idf desarrollado en la etapa 1, se usaron los mismos parámetros encontrados con el cross validation de dicha etapa. En el pipeline se guarda el vectorizador Tfidf y la regresión logística que emplea el algoritmo. Finalmente ajustamos y entrenamos el modelo que esta siendo persistido en el pipeline.

```
best_params = {
    'vect': {
        'ngram_range': (1, 1),
        'tokenizer': None,
    },
    'clf': {
        'C': 100.0,
        'penalty': 'l2',
```

```
    }
}

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

# Ajustamos el modelo al conjunto de entrenamiento transformado
model_pipeline.fit(X_train_transformed, y_train)
```

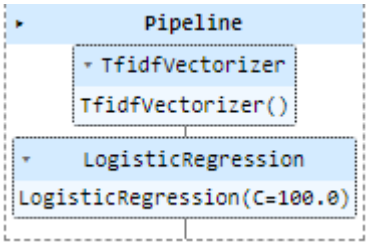

ILUSTRACIÓN 1: OUTPUT PIPELINE MODELO

3 Persistencia de los pipelines

En conclusión, hasta el momento tenemos dos (2) pipelines para persistir y usar como procesamiento en el API que diseñamos. La exportación de los pipelines se hizo gracias a la librería *joblib* la cual empaqueta estos procesos y transformaciones en archivos *.pkl* que después vamos a desempaquetar y usar en nuestro API. Así pues, a continuación, se muestra como se hizo dicho proceso de empaquetamiento y exportación en el notebook, el cual realmente, fue el proceso más sencillo del proyecto.

```
import joblib

# Guardamos el modelo entrenado en un archivo
joblib.dump(model_pipeline, 'modelo_logreg_0.pkl')
# Exportamos tambien el pipeline de transformación de datos
joblib.dump(data_prep_pipeline, 'data_prep_pipeline_0.pkl')
```

4 Acceso del API

Flask es un framework de desarrollo web para Python que permite crear aplicaciones web de manera sencilla y eficiente. Note que todo lo se va a explicar a continuación hace referencia al código desarrollado en el Backend del github adjunto en este proyecto. En este contexto, hemos utilizado Flask para desarrollar una aplicación web con las siguientes características:

- Configuración de la aplicación: Hemos configurado una aplicación Flask llamada app y definido las rutas a las que los usuarios pueden acceder.

- Ruta para cargar archivos (end-point de la API): Hemos creado una ruta /predict que permite a los usuarios subir su texto. Esta ruta es accesible mediante una solicitud POST, y cuando un usuario carga un archivo, el código dentro de la función predict() maneja la carga del archivo.

- Procesamiento de archivos: El código dentro de la función upload_file() verifica el texto que se envió mediante el POST, aplica las transformaciones, hace uso de los pipelines importados y finalmente imprime en un template renderizado el resultado de la predicción.

- Plantillas HTML: Flask permite renderizar plantillas HTML para presentar información en una interfaz web. Se han definido plantillas personalizadas para brindar una breve experiencia al usuario y una interfaz agradable para poder usar la api diseñada.

- Integración con librerías: Hemos utilizado librerías como pandas, joblib y otras para cargar modelos y realizar el procesamiento de datos. Note que todas estas dependencias son instaladas localmente en la maquina que despliega el servicio (endpoint) para la API.

```
app = Flask(__name__)
model = joblib.load('model.pkl')

@app.route('/')
def hello():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        input_texts = request.form.getlist('text')

        # Realiza las predicciones para cada texto
        predictions = model.predict(input_texts)

        # Devuelve las predicciones en un formato adecuado
        response = {'predictions': predictions.tolist()}

        return render_template('data.html',
predictions=response['predictions'])
    else:
```

```
        return 'Unsupported method'

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

5 Uso en la organización

5.1 Usuarios de la aplicación

Los usuarios se describen como organizaciones humanitarias o personas naturales interesadas en el monitoreo y análisis de opiniones de diferentes sectores de la población sobre diferentes las diferentes problemáticas se azotan sus respectivas comunidades, buscando identificar el tipo de ODS relacionado con la problemática descrita en cada opinión. Para este caso el usuario principal va a ser la organización UNFPA.

Se espera que el usuario tenga un rol de análisis dentro de la organización. El cual será responsable de ayudar en la toma de decisiones haciendo uso de la información suministrada por el modelo de clasificación.

5.2 Proceso de negocio que apoyará la aplicación

Se espera que como hemos mencionado anteriormente la aplicación se use como un recurso adicional de información que será util en la toma de decisiones. Concretamente, se espera facilitar la toma de decisiones para los planes de acciones de UNFPA ayudando con la recopilación de información relevante relacionada con los ODS que esta organización quiere trabajar.

El uso de la aplicación será de alta importancia, ya que la clasificación permite reducir en gran medida el tiempo invertido en el análisis de las problemáticas trabajadas por la organización haciendo que el análisis solo se ejerza sobre información relacionada.

6 Trabajo con equipo interdisciplinario

6.1 Decisiones esenciales

En el transcurso del proyecto tuvimos que tomar varias decisiones importantes para la aplicación. Una de estas fue ¿porque medio seria accesible? Finalmente decidimos aplicarlo todo en una página web para hacer el servicio accesible para todo el mundo. Esta y otras decisiones respecto a la interfaz fueron tomadas teniendo en cuenta la opinión de expertos en diferentes campos. Las opiniones de los expertos nos ayudaron a planear un buen despliegue y corregir errores en lo que sería el diseño de la aplicación.

Un proceso que también necesito tiempo de análisis fue la representación de los resultados y que tan amplio debía ser el formate de inputs que la aplicación debía recibir. Hablando con los expertos se decidió que la prioridad era poder procesar texto y archivos de texto planos.

6.2 Proceso de desarrollo

Durante el proceso de desarrollo se trabajó con un estudiante de estadística junto con el que se pudo ir estudiando la precisión del modelo clasificador. Lo cual permitió que el modelo fuera ajustado, ya que se empezaron a tener en cuenta las variables poblaciones y los sesgos bajo los cuales se estaba construyendo.

Una vez implementados los ajustes, el modelo se volvió más generalizable ya que se empezó a tomar muestras significativas provenientes de todos los sectores de la población, aumentando aún más la precisión del modelo.

Finalmente, se puede decir que la intervención del equipo de estadística en el proyecto fue pertinente, ya que nos dio una perspectiva mas ampli y nos permitió mejorar la calidad del producto generado, haciéndolo más consistente y coherente con lo que se buscaba lograr.

7 Trabajo en equipo

ROLES	MIEMBRO
Líder de proyecto:	Juan Coronel
Ingeniero de Datos:	Julian Villate
Ingeniero de Software – Diseño:	Diego Andres Parraga
Ingeniero de Software – Aplicativo:	Juan Coronel

Reuniones:

Fecha	25-10-2023
Objetivo	Seguimiento
Proceso	Se realiza demostración del funcionamiento del sistema, el equipo de estadística hace revisión de los resultados presentados. Se interactúa con la interfaz y se evalúa que tan amigable es con el usuario.
Compromisos:	• El equipo de analítica de datos mejorara la interfaz para incrementar la usabilidad por los usuarios • El equipo de estadística evaluará los resultados dados y que otras métricas pueden resultar de interés para el proyecto

Fecha	29-10-2023

Objetivo	Finalización
Proceso	Se realiza seguimiento a los compromisos previos. No se encuentran correcciones a las métricas y se aprueban las mejoras a la interfaz.
Compromisos:	• Realizar entrega del proyecto

ACTIVIDAD	RESPONSABLE	HORAS	RETOS
Configuración Aplicación	Juan Coronel	6	Selección de herramientas
Video	Andres Parraga	2	
Documentación reuniones	Andres Parraga	1	Documentación y formalización de los consensos
Diseño de HTML	Julian Villate, Juan Coronel	9	Correcciones por concepto del grupo d estadística.
persistencia del modelo y acceso por medio de API	Juan Coronel	7	Se almacena el modelo en un repositorio para el trabajo y aporte colaborativo
Definición de Usuario	Julian Villate	2	
Proceso de automatización del proceso de preparación de datos,	Juan Coronel	5	Análisis de las necesidades y el tipo de datos con los que se cuenta
construcción del modelo	EQUIPO CONJUNTO	12	Se realizo la evaluación de múltiples modelos, sus arquitecturas y cual se adaptaba mejor para la solución del problema

MIEMBRO	PARTICIPACION
Juan Coronel	40
Julian Villate	30
Andres Parraga	30

Valoración de los Resultados:

1. Alta Precisión: La precisión del 99.6% es excepcional. Esto significa que el modelo es altamente preciso en la clasificación o predicción de datos. Es fundamental para la toma de decisiones confiables.

2. Confiabilidad: Los resultados muestran que el modelo es confiable y estable. Puede utilizarse para tomar decisiones críticas en función de los datos analizados.

3. Interfaz de Usuario: La interfaz de usuario proporciona una forma efectiva de presentar estos resultados de manera accesible.

Conclusiones Generales:

1. Alta Capacidad Predictiva: Los resultados demuestran que el modelo tiene una capacidad excepcional para predecir o clasificar datos con una precisión del 99.6%. Esto lo hace adecuado para aplicaciones críticas.

2. Toma de Decisiones Informadas: La interfaz de usuario brinda a los usuarios la capacidad de acceder a resultados precisos y relevantes de manera eficiente, lo que les permite tomar decisiones informadas.

3. Optimización de Procesos: La alta precisión del modelo puede conducir a una optimización significativa de los procesos empresariales al reducir los errores y mejorar la eficiencia.

4. Comunicación y Muestra de resultados: La interfaz de usuario es fácil de entender y utilizar. La comunicación efectiva de los resultados y la forma en que se llegó a ellos es fundamental para la usabilidad de los usuarios.

Cosas Por Mejorar:

- Sensible a la longitud del documento: TF-IDF no considera la longitud de los documentos, lo que puede llevar a sesgos en documentos largos que simplemente contienen más palabras.
- No considera la semántica: TF-IDF se basa únicamente en la frecuencia de palabras y no tiene en cuenta la relación semántica entre las palabras.