Proyecto 1

Inteligencia de Negocios

Etapa 2

Grupo 30

German Alberto Rojas Cetina 202013415

María Paula Almeciga Moreno 202023369

Juan Diego Sarmiento Sánchez 202121484

Tabla de Contenido

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

Para comenzar con la implementación de la etapa 2 se creó un pipeline con 3 fases. La primera fase hace referencia a la preparación de los textos a través de varios métodos personalizados. Luego se utilizó el TfidfVectorizer a partir de los tokens generados dentro del paso anterior, para generar la vectorización del texto y poder ser utilizado dentro del modelo. Para finalmente utilizar el modelo de Random Forest, el cual fue el que se identificó como mejor, con los mejores parámetros definidos de la etapa1. A continuación, se puede evidenciar un fragmento del código utilizado para implementar el pipeline. En caso de desear ver todo el código para este se puede referir al Anexo 1.

```
procesamiento_transformer = FunctionTransformer(aplicar_procesamiento)
parametros_2 = {
    "vectorizer": TfidfVectorizer(tokenizer=word_tokenize,
stop_words=stop_words),
    "vectorizer__lowercase": True,
    "classifier__n_estimators": 300,
    "classifier__criterion": 'gini',
    "classifier__max_depth": 100
}
pipeline_2 = Pipeline(steps=[
    ("procesamiento", procesamiento_transformer),
    ("vectorizer", parametros_2["vectorizer"]),
    ("classifier", RandomForestClassifier(
        n_estimators=parametros_2["classifier__n_estimators"],
        criterion=parametros_2["classifier__criterion"],
        max_depth=parametros_2["classifier__max_depth"],
        random_state=4))])
```

A partir del pipe line se implementó una API desarrollada con FastAPI, la cual permite realizas predicciones y reentrenamiento de un modelo de clasificación de texto. Esta API es fundamental para la automatización de tareas analíticas, permitiendo a los usuarios interactuar con modelos de Machine Learning de manera sencilla y eficiente.

La API cuenta con dos endpoints principales: ‘/predict’ y ‘/retrain’. El endpoint de ‘/predict’ permite a los usuarios enviar texto o archivos para obtener predicciones sobre el contenido. Es capaz de manejar dos tipos de entradas:

- Texto: Los usuarios pueden enviar texto directamente mediante un formulario.
- Archivo: Los usuarios pueden subir un archivo Excel que contenga textos para su análisis.

La implementación del endpoint sigue una estructura lógica:

-Recepción de datos. El endpoint recibe el texto o archivo mediante un formulario. Si se proporciona un archivo, se valida que este sea de tipo Excel.

-Lectura y Procesamiento. Los datos son leídos y almacenados en un DataFrame para facilitar su manipulación.

-Predicción. Se utilizan los métodos del pipeline previamente entrenado para realizar predicciones y calcular las probabilidades asociadas a cada clase.

-Respuesta. Las predicciones y probabilidades se devuelven como un objeto JSON, lo que facilita su uso en aplicaciones web y móviles.

Gracias a esta estructura, se garantiza que los usuarios puedan obtener resultados precisos de manera eficiente, ya sea que trabajen con textos individuales o conjuntos de datos más grandes.

El segundo endpoint, '/retrain', permite actualizar el modelo con nuevos datos. Este proceso de reentrenamiento es crucial para mantener la precisión del modelo a medida que se dispone de nueva información. La lógica de este endpoint se estructura de la siguiente manera:

- Recepción de archivos. Los usuarios envían un archivo que contenga datos en el formato CSV o Excel. Este archivo debe incluir las columnas 'Textos_espanol' y 'sdg', que son escenciales para el reentrenamiento.
- Validación de datos. Se verifica que el archivo contenga las columnas necesarias. En caso contrario, se devuelve un error detallado.
- Reentrenamiento. Se crea un nuevo DataFrame a partir de los textos y etiquetas y se utiliza para reentrenar el modelo. Este enfoque asegura que el modelo pueda adaptarse a nuevos patrones y tendencias de los datos.
- Cálculo de métricas. Depsués del reentrenamiento, el modelo se evalúa mediante métricas como precisión, recall y F1-score. Estas métricas son fundamentales para entender el rendimiento del modelo y realizar ajustes si es necesario.

El reentrenamiento incremental se ha implementado en la API por varias razones estratégicas y operativas. En primer lugar, este enfoque permite que el modelo se adapte de manera continua a los nuevos datos sin necesidad de ser entrenado desde cero. Esto es particularmente importante en entornos donde los datos están en constante evolución, como es el caso del análisis de texto, donde las tendencias, terminología y contextos pueden cambiar rápidamente. Al utilizar reentrenamiento incremental, la API puede incorporar información reciente de manera más eficiente, mejorando así la precisión y relevancia del modelo sin requerir grandes recursos computacionales o tiempo significativo de entrenamiento. Además, el reentrenamiento incremental facilita la implementación de un flujo de trabajo más ágil, permitiendo a los usuarios actualizar el modelo con nuevos datos según sea necesario. Esto no solo optimiza el rendimiento del modelo, sino que también proporciona a los usuarios la flexibilidad necesaria para adaptarse a las dinámicas cambiantes

del mercado o del contexto en el que se utilizan los modelos, asegurando que el análisis se mantenga alineado con las necesidades actuales de la organización.

A pesar de sus ventajas, este tipo de reentrenamiento también presenta varias desventajas que deben ser consideradas. En primer lugar, existe el riesgo de concept drift, lo cual se traduce en la degradación del rendimiento del modelo debido a cambios en la distribución de los datos a lo largo del tiempo. Si el modelo no se adapta adecuadamente a estos cambios, puede resultar en predicciones inexactas. Además, este tipo de reentrenamiento puede llevar a una acumulación de errores, donde los errores de las iteraciones anteriores se transfieren al modelo en nuevas actualizaciones, afectando la calidad general de las predicciones.
Otra desventaja por considerar es que la gestión de datos puede volverse más compleja. Para esto es importante mantener un registro de los datos de entrenamiento y asegurarnos de que se cumplan las normas de calidad y consistencia. Esto podría requerir una infraestructura adicional para manejar los nuevos datos y la integración en el modelo existente.
Adicionalmente, el reentrenamiento incremental puede no ser adecuado para todos los tipos de modelos, especialmente aquellos que requieren un entrenamiento más exhaustivo para ajustar sus parámetros. Por último, el uso frecuente de reentrenamiento puede llevar a un incremento en el consumo de recursos computacionales ya que cada actualización del modelo requiere tiempo y procesamiento, lo que puede ser un problema en entornos con recursos limitados.

Por otro lado, se identificaron otros tipos de rentrenamiento que no se consideraron para la implementación. El primero es el de rentrenamiento completo, en donde se elimina el modelo actual para a partir de nuevo datos entrenar desde 0 otro modelo completamente nuevo. Este tiene una gran ventaja, que en el caso que se tenga un gran cambio de los datos del negocio o del comportamiento de estos mismo el modelo será capaz de adaptarse a estos cambios de una forma efectiva. Pero cada vez que se rentrene se estará olvidado todo lo que ya se ha realizado, lo cual puede perjudicar si no se ha han hecho cambios tan significativos a los datos al eliminar todo lo que ya se poseía.

Finalmente, existen los modelos de entrenamiento de transición, los cuales consisten en realizar una combinación de los dos mencionados previamente. Donde se realiza una combinación del modelo viejo con un nuevo modelo actualizado. Estos en general permiten que el modelo se pueda ajustar a los cambios que se presentan dentro de los datos, al mantener conocimientos previos, pero integrarlos a conocimientos completamente nuevos. Por otro lado, estos pueden ser muy exigentes computacionalmente al tener que almacenar dos modelos, que de por si cada uno puede ser exigente para la memoria. Además de que, sí se presentan errores dentro del modelo viejo estos seguirán después del rentrenamiento.

Sección 2. Desarrollo de la aplicación y justificación.

Una vez ya creada la API con los dos endpoint para el procesamiento de los textos se desarrolló una aplicación para que pueda ser utilizada por algún usuario. En específico se desarrolla la aplicación para trabajadores de la UNPFA, es decir personas que son encargadas del papeleo respecto a los ODS.

Dentro de los roles mencionados dentro de la etapa 1 se pueden ver especialmente beneficiados Personal general de la UNFPA y Asistentes de seguridad local. Debido a que la idea de la aplicación es automatizar la lectura de las peticiones realizadas por los usuarios, que no necesariamente tienen un ODS asignado. Estas peticiones las revisan estos roles de la UNFPA, los cuales pueden recibir una gran cantidad de peticiones, las cuales deben clasificar, la aplicación es capaz de automatizar esta tarea. Aun así, esta aplicación puede ser útil para las personas que suban peticiones

Esta aplicación tiene una gran importancia para el negocio en general, gracias que el objetivo del negocio es poder realizar estrategias para mejorar la calidad de vida basadas en las evaluaciones de los ODS. Las peticiones son las bases para estas decisiones, gracias que de esta forma se puede ver que es lo que está fallando y solucionar. La aplicación permite identificar y clasificar de forma automática estas peticiones, lo cual permite al negocio ver en que ODS se debe enfocar de una forma más rápida y efectiva. Lo que beneficia a los empleados, gracias que cumplen con su labor de forma más sencilla; las directivas, gracias que se puede empezar a desarrollar las estrategias y políticas de forma más rápida; las personas, gracias que sus peticiones podrán ser revisadas más rápido.

La aplicación fue desarrollada mediante Flask, en dónde se pueden subir las peticiones en dos formatos, ya sea dentro de la caja de texto presente dentro de la vista o mediante la carga de un archivo csv. Este archivo debe contener todas las peticiones que se deseen revisar y clasificar. La aplicación responde a partir de esto cual es el ODS que tiene más probabilidades de pertenecer la petición. Adicionalmente de mostrar la probabilidad que tiene de pertenecer a este mismo.

Por otro lado, en el caso de tener una gran cantidad de datos nuevo y se deseen implementar en el modelo estos se pueden cargar en otra pestaña. A partir, de los datos en un archivo csv y con el formato explicado previamente se pueden añadir al modelo y este se volverá a entrenar. La aplicación luego mostrará las métricas de evaluación del modelo ya entrenado nuevamente.

Sección 3. Resultados.

Una vez implementada la aplicación se puede ver la clasificación de todas las peticiones que se deseen, lo cual puede brindar implicaciones fuertes para el desarrollo sostenible. Como se ha mencionado previamente se pueden identificar los ODS 3,4 y 5 de una forma más rápida.

Adicionalmente se puede ver Colombia implicada dentro de esto, gracias a como se mencionó dentro de la etapa 1. La aplicación puede brindar información sobre la igual de género la educación y la salud para la sociedad Colombiana a partir de las peticiones realizadas. Lo cual permite al Gobierno y a las organizaciones interesadas en el desarrollo colombiano realizar políticas y planes de accione adecuados y enfocados en los ODS que más se presentan.

Adicionalmente a partir de esta aplicación se puede tener un inicio para evaluar las políticas y planes implementados a partir de los ODS que se logren identificar.

Sección 4. Trabajo en equipo.

Roles:

- Líder de proyecto:
    - Juan Diego Sarmiento Sánchez:
        - Definición de fechas y planificación general del proyecto.
        - Coordinación de reuniones y entregables del equipo.
        - Verificación de las asignaciones de tareas para garantizar la equidad.
        - Responsable de la subida de la entrega final del grupo.

- Ingeniero de datos:
    - Germán Alberto Rojas Cetina
        - Creación del pipeline.
        - Resolución de los conflictos causados por el pipeline y el modelo que se desea guardar.
        - Selección del mejor modelo construido previamente en la etapa 1.

- Ingeniero de software responsable del diseño de la aplicación y resultados:
    - Germán Alberto Rojas Cetina
        - Creación de la API para los endpoints.
        - Configuración de los endpoints para recibir los datos deseado.
        - Selección del método de rentrenamiento del modelo.

- Ingeniero de software responsable de desarrollar la aplicación final:
    - María Paula Almeciga Moreno
        - Creación de la aplicación interactiva para el usuario.
        - Configuración de la app para que se amigable con el usuario.
        - Recepción de archivos CSV para rentrenamiento y predicción.

Tiempos y retos

- El proyecto se desarrolló durante 2 semanas, con aproximadamente 6 horas de trabajo por semana para cada miembro del equipo.
- Un reto principal fue la creación del pipeline para el uso dentro de la API, gracias que este generaba diferentes errores a la hora de realizar la limpieza de los datos con un método personalizado del pipeline.
- Se presentó un desafió al utilizar nuevas herramientas de software para la implementación de la API y de los dos endpoints.
- La comunicación efectiva y la coordinación de tareas fueron esenciales para mantener el proyecto en marcha.

Reparto de puntos

- Germán Alberto Rojas Cetina: 100/3
- María Paula Almeciga Moreno: 100/3
- Juan Diego Sarmiento Sánchez: 100/3

Puntos a mejorar

Dentro del proyecto se tuvo una división clara de los trabajos realizados por cada persona, y de los roles que cada uno debía tener. Esto facilitó la realización del proyecto y sus respectivas tareas. Una parte la cual hizo falta dentro del proyecto fue la realización de una mayor cantidad de reuniones, gracias que el contacto que se realizaba dentro del grupo se hacía de forma asíncrona.

Si bien no hubo falta de comunicación gracias que el canal asíncrono fue bastante activo, se pudo tener reuniones más seguido para coordinar las tareas, dar actualizaciones y entender el problema de una mejor forma.

Reuniones de grupo

- Se llevó a cabo una reunión de lanzamiento y planeación del inicio del proyecto para definir los roles y estrategias de trabajo.
- Se programó una reunión de finalización en la última fase de la etapa 2 del proyecto para consolidar el trabajo, corregir errores y analizar oportunidades de mejora.

Anexos Electrónicos

Anexo 1

text_preprocessing.py	Archivo de Python con todas las funciones de preparación para el texto
AnalisisTextos.py	Note book de Python con la etapa 1 donde se crea el pipeline para ser utilizado en esta etapa.

Referencias.

[1]. UNFPA en Colombia. (s. f.). UNFPA-Colombia. https://colombia.unfpa.org/es/unfpa-en-colombia

[2]. *Objetivos de desarrollo sostenible*. (s. f.). UNDP. https://www.undp.org/es/sustainable-development-goals

[3]. UNFPA. (2013). Roles and Responsibilities of actors within UNFPA. En *UNFPA*. https://www.unfpa.org/sites/default/files/admin-resource/OSC_Annex%20Security%20Accountability_1.pdf

[4]. Dilmegani, C. (2024, 3 septiembre). *Model Retraining: Why & How to Retrain ML Models?* AIMultiple: High Tech Use Cases &Amp; Tools To Grow Your Business. https://research.aimultiple.com/model-retraining/