Universidad de Los Andes

Inteligencia de Negocios

Informe del Proyecto

Integrantes:

* Miguel Angel Ariza Jimenez – 202010620
* Julian Escobar Rivera – 202111361
* Juan David Obando Novoa – 202123148

**Proyecto 1 – Analítica de datos – Etapa 2**

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

1. Preparación de Datos:

El proceso comienza con la preparación de los datos, que es una etapa fundamental para garantizar que los datos estén limpios y listos para ser utilizados por el modelo. Esta etapa fue automatizada dentro del API utilizando funciones de preprocesamiento que se encargan de:

* Limpieza de texto: Se eliminaron caracteres especiales, se convirtieron los textos a minúsculas, y se eliminaron los números y los espacios extra para normalizar los datos de entrada.
* Corrección de caracteres especiales: Se implementó una función específica para corregir caracteres mal codificados que podrían haber aparecido debido a errores de codificación en los datos originales.
* Vectorización de los textos: Una vez que los textos fueron limpiados, se transformaron en vectores de características utilizando un vectorizador TF-IDF. Este paso convierte los textos en representaciones numéricas que el modelo puede procesar.

1. Construcción del Modelo:

El modelo seleccionado para el proceso de clasificación fue Naive Bayes, que es particularmente adecuado para tareas de clasificación de texto debido a su simplicidad y efectividad en situaciones donde los datos son representaciones de conteos, como TF-IDF. Tambien se escogió este debido a que el modelo SVC, que resulto tener las mejores métricas en la etapa anterior, no permita el reentrenamiento incremental de manera nativa. Esto se explicará más adelante.

El proceso automatizado incluye las siguientes etapas:

* Cargar el modelo: Como el modelo ya había sido entrenado previamente en el notebook de la etapa 1, se cargó desde un archivo persistente para ser utilizado de inmediato.
* Reentrenamiento incremental: En cada nueva solicitud de reentrenamiento, el modelo es actualizado mediante entrenamiento incremental. Esto permite agregar nuevos datos sin necesidad de reentrenar el modelo desde cero, optimizando el uso de recursos computacionales.

1. Persistencia del Modelo:

Una vez que el modelo ha sido entrenado o reentrenado, este es persistido para asegurar que las mejoras y actualizaciones se mantengan para futuras consultas. Este proceso se gestiona utilizando joblib, que guarda el modelo entrenado en un archivo local, de modo que pueda ser cargado fácilmente cuando sea necesario:

* Guardado del modelo: El archivo modelo\_nb.joblib contiene el modelo entrenado y se guarda automáticamente después de cada reentrenamiento.
* Cargar el modelo: En cada llamada de predicción o reentrenamiento, el modelo se carga desde este archivo, lo que asegura que las mejoras acumuladas se mantengan.

1. Acceso a través de la API:

El acceso a las funcionalidades del modelo se proporciona a través de una API basada en FastAPI. A través de esta API, se han implementado dos puntos principales de acceso:

* Predicción (/predict):

Recibe un conjunto de textos en formato JSON. Realiza la predicción utilizando el modelo, devolviendo tanto la clase predicha como las probabilidades para cada clase.

Ejemplo de solicitud:

{

"texts": ["La salud mental es importante", "La igualdad de género es fundamental"]

}

* Reentrenamiento (/retrain):

Recibe un conjunto de textos y sus etiquetas correspondientes en formato JSON. Actualiza el modelo mediante reentrenamiento incremental utilizando los nuevos datos.

Devuelve métricas como precisión, recall y F1-score para evaluar el rendimiento del modelo después del reentrenamiento.

Ejemplo de solicitud:

{

"texts": ["La educación es esencial para el desarrollo", "La salud pública es una prioridad"],

"labels": [4, 3]

}

1. Registro de Logs y Seguimiento:

Para facilitar el monitoreo y el diagnóstico de posibles errores, se implementó un sistema de logging que registra cada solicitud enviada al API. Esto incluye:

* Datos de entrada: Los textos y etiquetas enviados en cada solicitud de predicción o reentrenamiento.
* Resultados: Las predicciones generadas o las métricas de rendimiento obtenidas después de un reentrenamiento.

1. Planteamiento definiciones reentrenamiento:

Para cumplir con este requerimiento, se plantearon tres enfoques distintos para el reentrenamiento del modelo. Estas son las tres definiciones con sus descripciones, ventajas y desventajas:

* Reentrenamiento Completo desde Cero

Descripción: El reentrenamiento completo implica volver a entrenar el modelo desde el principio, utilizando todos los datos disponibles, tanto antiguos como nuevos. En este enfoque, el modelo se reinicia y aprende nuevamente desde cero utilizando el conjunto de datos completo, actualizándose con los datos nuevos que han sido añadidos.

Ventaja: Mayor precisión global: Al entrenar desde cero con todos los datos, el modelo tiene la oportunidad de aprender patrones más complejos y ajustar mejor los parámetros con todos los datos disponibles, lo que puede generar un modelo con mejor rendimiento general.

Desventaja: Requiere mucho tiempo y recursos: El reentrenamiento completo puede ser muy costoso computacionalmente, especialmente si el conjunto de datos es grande, ya que el modelo debe procesar y ajustar a todos los datos desde el principio cada vez que se entrena.

* Reentrenamiento Incremental (Entrenamiento con Nuevos Datos)

Descripción: El reentrenamiento incremental implica actualizar el modelo solo con los nuevos datos disponibles, sin volver a procesar el conjunto de datos anterior. Este enfoque se logra mediante técnicas como el uso de partial\_fit, que permite a ciertos modelos como Naive Bayes continuar el entrenamiento con los datos adicionales sin perder lo que ya han aprendido.

Ventaja: Es mucho más rápido y menos costoso computacionalmente que el reentrenamiento completo, ya que solo se actualiza el modelo con los nuevos datos, lo que ahorra tiempo y recursos, especialmente cuando el conjunto de datos original es grande.

Desventaja: Posible pérdida de rendimiento: Como el modelo solo está aprendiendo de los nuevos datos, puede no tener la oportunidad de ajustar completamente los parámetros con respecto a los datos anteriores, lo que podría llevar a un modelo que no generalice tan bien a los datos viejos.

* Reentrenamiento Periódico

Descripción: Este enfoque combina ambos métodos anteriores. El modelo se entrena de manera incremental a medida que llegan nuevos datos, pero también se reentrena desde cero de forma periódica para incorporar de manera integral tanto los datos antiguos como los nuevos. Esto podría realizarse, por ejemplo, cada cierto número de iteraciones, o cuando se alcanza un umbral en la cantidad de nuevos datos.

Ventaja: Este enfoque equilibra la necesidad de mantener el rendimiento global al reentrenar el modelo desde cero de forma periódica, mientras que el entrenamiento incremental entre reentrenamientos completos mantiene el modelo actualizado sin usar demasiados recursos.

Desventaja: La planificación de cuándo y cómo se debe realizar el reentrenamiento completo añade complejidad adicional, ya que se debe establecer un sistema que determine cuándo es necesario reentrenar desde cero y cuándo actualizar incrementalmente.

* Enfoque usado

En este proyecto, se optó por implementar el enfoque de reentrenamiento incremental, utilizando el método partial\_fit del modelo Naive Bayes. Este enfoque fue elegido debido a su eficiencia computacional y flexibilidad al permitir que el modelo se actualice rápidamente con los nuevos datos sin tener que reentrenarlo completamente desde cero cada vez. Aunque puede haber una posible pérdida de rendimiento en comparación con un reentrenamiento completo, este enfoque se adapta mejor a las necesidades del proyecto en términos de velocidad y capacidad para procesar nuevos datos de manera continua.

Sección 2. Desarrollo de la aplicación y justificación