Proyecto 1 – Etapa 2

Daniel Triviño – 202113218

Jose Cristobal Arroyo – 202011404

Andres Cordero – 202011880

Desarrollo de la aplicación de analítica

Para el desarrollo del proyecto se optó por crear una aplicación web con las siguientes características

Característica	Valor
Lenguaje para el backend	Python
Frameworks para el backend	Uvicorn, FastApi
Modelo de aprendizaje utilizado	Multinomial Naive Bayes
Frameworks para el pipeline de aprendizaje	SKlearn, CloudPickle
Lenguaje para el frontend	Javascript
Frameworks para el frontend	ReactJS y Axios
Despliegue	Máquina Local

Características de la aplicación

- Predicción del ODS más relacionado con un conjunto de textos en español. La entrada puede ser un JSON o un CSV
- Reentrenamiento incremental del modelo de aprendizaje. La entrada puede ser un JSON o un CSV.
- Conserva una nueva versión del modelo de aprendizaje cada vez que es reentrenado y le asigna una versión. Estas versiones pueden ser revertidas para conservar el mejor modelo posible.
- Evaluación del modelo tras cada reentrenamiento con un subconjunto de datos de validación, el cual nunca es usado para entrenar el modelo.

Análisis de tipos de entrenamiento

Hay tres maneras diferentes de Re-entrenar un modelo de aprendizaje automático. Cada tipo tiene diferentes ventajas y desventajas, y debe ser escogido dependiendo de los requerimientos de la aplicación.

Reentrenamiento incremental

Algunos modelos de machine learning permiten realizar entrenamiento incremental. Esto significa que el modelo puede ser entrenado múltiples veces con diferentes conjuntos de datos. Tras cada

entrenamiento, el modelo no descarta el entrenamiento previo, sino que procura mejorarse cada vez. Esto es ideal para aplicaciones en las cuales los modelos no tienden a volverse obsoletos y en general pueden mejorar al utilizar mayores volúmenes de datos.

Reentrenamiento total

En otras aplicaciones es deseable entrenar un modelo completamente nuevo en base a un conjunto de datos. Esto puede ser útil para aplicaciones en las cuales el problema planteado cambia de manera importante. Por ejemplo, si en el contexto de este proyecto se cambia el requerimiento para reconocer todos los ODS en vez de solo tres, puede ser ideal entrenar un modelo completamente nuevo.

Reentrenamiento parcial

Otras veces, puede ser deseable que un modelo descarte una parte de los datos con los que se ha entrenado previamente, mas no todos. Esto es importante para problemas de series de tiempo en los cuales los modelos se vuelven obsoletos con el paso del tiempo. Un ejemplo de eso es un modelo predictivo para el mercado de acciones. No es práctico para el modelo hacer predicciones basadas en los datos de toda la historia, sino en una ventana de tiempo definida hacia el pasado desde el momento presente.

Endpoints de la aplicación

Healthcheck

Muestra el estado de salud del servidor

Path: /

Method: GET

Predict one

Recibe una lista de uno o mas textos en español para ser clasificados.

Path: /predict

Method: POST

Input: JSON:

{ Textos_espanol: List of strings }

Output: JSON:

{ predictions: [{sdg: int, prob: float}…]}

Predict Many

Un endpoint util para cargar un gran volumen de textos a través de un CSV.

Path: /predict/multiple

Method: POST

Input: FILE .CSV

Input Columns: Textos_espanol

Output: JSON:

{ predictions: [{sdg: int, prob: float}…]}

Train

Un endpoint para cargar features, la variable objetivo y reentrenar el modelo de manera incremental.

Path: /train

Method: POST

Input: .CSV FILE

Input Columns: Textos_espanol, sdg

Input: JSON

{Textos_espanol: List of strings, sdg: List if integers}

Output: JSON

{accuracy: float,

f1_score: float,

precision: float,

recall: float,

message: str}

Version

Un endpoint que permite ver la última versión semántica del modelo de aprendizaje.

Path: /versión

Method: GET

Response: JSON ->

{version: int}

Rollback

Un endpoint que descarta la última versión del modelo de aprendizaje.

Path: /rollback

Method: POST

Response: JSON ->

{message: “Model rolled back to version ‘version’”}

Reset

Un endpoint que descarta todas las versiones del modelo excepto la original.

Path: /reset

Method: DELETE

Response: JSON ->

{‘message’: ‘Model reset successfully’}

Trabajo en equipo

Líder del equipo:

Daniel Triviño

Ingeniero de datos y desarrollador Backend:

Cristobal Arroyo

Ingeniero de software responsable de Frontend:

Andrés Cordero

Trabajo de los integrantes

Cristobal Arroyo: Desarrollo de API en python

Andrés Cordero: Desarrollo de Frontend en React

Daniel Triviño: Planeación, planteamiento de método de reentrenamiento y documentación

Reparto de puntos

Cristobal Arroyo: 36

Andres Cordero: 35

Daniel Triviño: 29