Operaciones de Machine Learning

Proyecto 3
Nivel 3

Pontificia Universidad Javeriana
Cristian Javier Diaz Alvarez

23 de abril de 2025

1. Descripción

Este proyecto busca evaluar el proceso despliegue de MLOPS usando Kubernetes, empezando por la recolección y procesamiento de datos, para generar una fuente de información lista para el entrenamiento de modelos de Machine Learning. El modelo de mejor desempeño debe usarse para realizar inferencia mediante una API, la cual se consume mediante una interfaz gráfica. Dando seguimiento a las métricas de los sistemas desplegados y su infraestructura. Para esto se debe:

- Usando AirFlow cree los DAGs que le permitan recolectar, procesar y almacenar los datos.
- Para el registro de experimentos y modelos utilice MLflow
- Usando FastAPI cree un API que consuma el mejor modelo
- Cree una interfaz gráfica usando Streamlit que permita realizar inferencia.
- Usando Prometheus y Grafana recolecte información de uso de los sistemas para observabilidad.
- Usando Locust determinar la cantidad máxima de usuarios que la aplicación puede soportar

Para crear el servidor de AirFlow, depende del estudiante decidir donde se alojará, puede ser en un contenedor usando docker-compose o en kubernetes.

Este proyecto pretende encontrar el mejor modelo posible para el dataset propuesto, sin embargo, **se espera tener seguimiento del proceso de experimentación**, aprovechando de MLflow su seguimiento de metadatos y artefactos. Cada uno de los procesos de entrenamiento debe ser orquestado y ejecutado por AirFlow, realizándose de manera periódica. Adicionalmente, se deben publicar en MLflow los modelos y asignar automáticamente al Stage de **-production-** el que mejor desempeño presente, esto permitirá generar un despliegue que no requiere modificaciones de código para cambio en la versión de un modelo.

Para la inferencia se espera una interfaz gráfica que permita interactuar con el modelo de producción, esta debe permitir el ingreso de valores, tener un valor pre-definido e indicar cual fue el modelo que usó. (versión).

Usando los componentes anteriores, se deben desarrollar las funcionalidades descritas, siguiendo la arquitectura propuesta. La entrega, consiste en código fuente en repositorio publico, despliegue del sistema usando Kubernetes sobre la maquina proporcionada exponiendo la interfaz gráfica (20 %), seguimiento de experimentos mediante MLflow con bucket y base de datos (20 %). El proceso de inferencia debe tomar siempre el modelo definido en MLflow como producción, sin cambios en código (20 %). La recolección, procesamiento, almacenamiento de datos y entrenamiento de modelos debe realizarse usando AirFlow (20 %). La observabilidad debe hacerse como mínimo, sobre la API de inferencia, simulando los usuarios mediante Locust (10 %). Como sustentación del proyecto, se debe

entregar un vídeo en un canal de YouTube con una duración no mayor a 10 minutos. En donde explique, organización del proyecto, cambios en arquitectura, procesamiento y experimentación realizada, posteriormente mostrar/usar interfaz gráfica que permite realizar inferencia, por último explicar las métricas recolectadas y explicar el o los dashboard creados(10 %).

2. Descripción del dataset

El conjunto de datos representa 10 años (1999-2008) de atención clínica en 130 hospitales de EE.UU. y redes de entrega integradas. Incluye más de 50 características que representan los resultados del paciente y del hospital. Se extrajo información de la base de datos para encuentros que cumplían con los siguientes criterios.

1. Es un encuentro de paciente hospitalizado (un ingreso en el hospital).
2. Es un encuentro diabético, es decir, durante el cual se ingresó al sistema como diagnóstico cualquier tipo de diabetes.
3. La duración de la estadía fue de al menos 1 día y como máximo 14 días.
4. Se realizaron pruebas de laboratorio durante el encuentro.
5. Se administraron medicamentos durante el encuentro.

Los datos contienen atributos tales como número de paciente, raza, sexo, edad, tipo de ingreso, tiempo en el hospital, especialidad médica del médico que lo admitió, número de pruebas de laboratorio realizadas, resultado de la prueba de HbA1c, diagnóstico, número de medicamentos, medicamentos para la diabetes, número de pacientes ambulatorios. , hospitalización y visitas de emergencia en el año anterior a la hospitalización, etc. Diabetes 130-US hospitals for years 1999-2008

2.1. Cargar el Dataset

Utilice las siguientes celdas para cargar el conjunto de datos **modificado** en su espacio de trabajo.

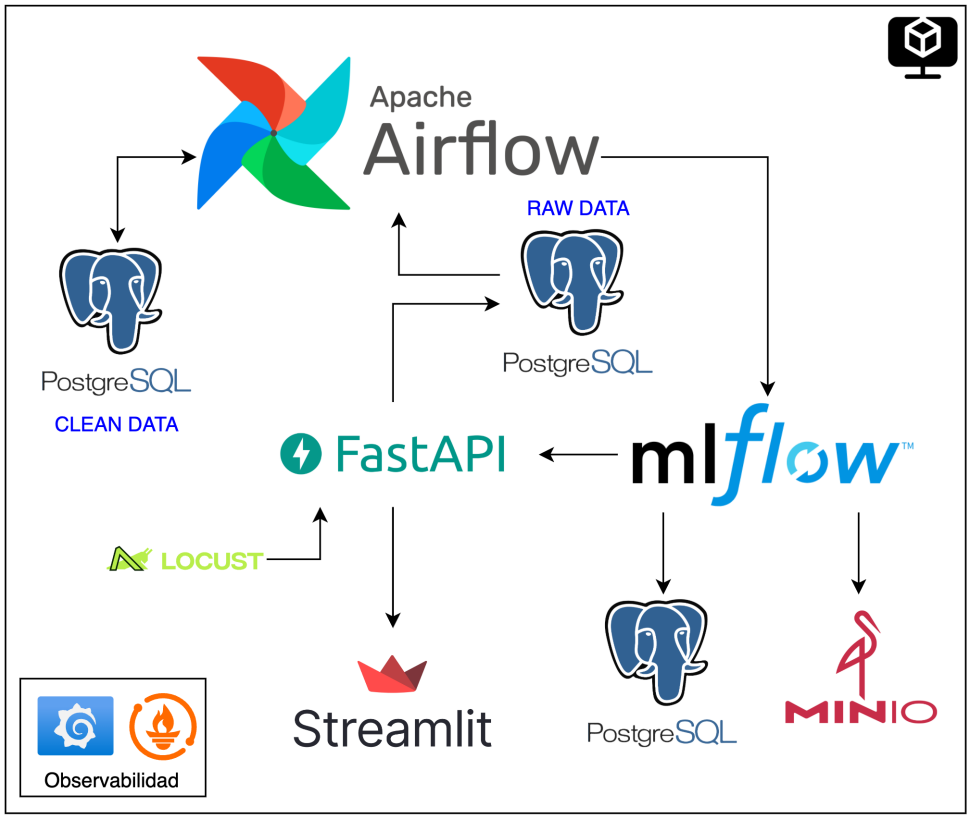
```
## download the dataset
# Directory of the raw data files
_data_root = './data/Diabetes'

# Path to the raw training data
_data_filepath = os.path.join(_data_root, 'Diabetes.csv')

# Download data
os.makedirs(_data_root, exist_ok=True)
if not os.path.isfile(_data_filepath):
    #https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/
    url = 'https://docs.google.com/uc?export= \
    download&confirm={{VALUE}}&id=1k5-1caezQ3zWJbKaiMULTGq-3sz6uThC'
    r = requests.get(url, allow_redirects=True, stream=True)
    open(_data_filepath, 'wb').write(r.content)
```

Este es todo el conjunto de datos, debe realizarse completo su procesamiento, desde análisis, limpieza, ingeniería de características y creación de subconjuntos de entrenamiento, validación y test. Este procesamiento se tendrá en cuenta para la asignación de la nota. Si desea descargar el archivo de manera directa puede encontrarlo Acá

3. Arquitectura

Este proyecto busca exponer al estudiante a una arquitectura típica en equipos que desarrollan/aplican inteligencia artificial. Estos equipos se enfrentan a tareas relacionadas a la recolección de datos, procesamiento de información, entrenamiento y puesta en producción de modelos de Inteligencia Artificial, así como el monitoreo de la infraestructura y los servicios que fueron desplegados. Para dar un adecuado manejo de este escenario, se propone una arquitectura que permite orquestar todo este proceso mediante AirFlow, realizar seguimiento de experimentos usando MLflow y distintos componentes de almacenamiento de datos crudos, información procesada, artefactos y meta-datos. Como puede observar, estos componentes de manera individual son el acumulado del desarrollo del curso por lo que su integración no debe suponer un nuevo reto.

El seguimiento de experimentos de Inteligencia Artificial, requiere el almacenamiento de dos conjuntos de datos, Artefactos y Metadatos. Los artefactos, hacen referencia al los elementos resultantes al proceso de entrenamiento, métricas, código, imágenes, modelo, entre otros. Los metadatos, es la información relativa a la ejecución, parámetros, tiempos, conjuntos de ejecución, entre otros. Para almacenar estos datos, se requieren dos artefactos, una base de datos SQL para los metadatos y un sistema de archivos para los artefactos, estos dos elementos dan soporte a MLflow como herramienta de seguimiento. Para la ejecución de experimentos se requiere otro sistema en donde se van a ejecutar los entrenamientos que se registrarán en MLflow. Por ultimo, el sistema de archivos, también funciona como registro de modelos, por lo cual, es posible mantener los modelos de producción o desarrollo disponibles para ser usados por las aplicaciones desplegadas.



El proceso de orquestación se realiza mediante AirFlow, esta herramienta ya contiene bases de datos, cache, logs, plugins, todo lo necesario para funcionar, mediante la creación de DAGs se definen los flujos de ejecución. Para el proceso de experimentación se deben presentar los argumentos por los cuales, se modifican y separan los datos, se seleccionan los modelos y se definen métricas. Para entrenar el modelo se debe crear el pipeline de procesamiento y entrenamiento que debe ejecutar AirFlow, para los dos escenarios anteriores se debe registrar en MLflow la experimentación realizada, adicionalmente

registrar los modelos. Por ultimo, para el proceso de inferencia se debe utilizar los modelos registrados en MLflow, los modelos deben estar publicados y asociados al stage de producción, lo que permite cambiar el modelo seleccionado, sin realizar cambios en el código desplegado.

Como observará en la descripción de componentes, en la mayoría de casos solo se especifica, su función y relación. Es trabajo del estudiante definir como va a implementar el sistema, recuerde, este ejercicio busca acercarlo a un escenario de producción. Es decir, la experimentación genera modelos y despliegues de clientes consumen esos modelos. Es importante la disponibilidad de cada elemento para garantizar que el equipo de científicos de datos pueda registrar sus modelos y el despliegue de los sistemas de los clientes estén disponibles.

4. Componentes

Para construir este sistema, cada componente debe tener una función clara y definida, así mismo las interacciones entre los componentes debe ser explicita, es importante prever escenarios de fallo y tomar medidas o elevar los errores necesarios. Cada componente debe existir dentro de un contenedor y estar interconectado en red o archivos según la necesidad, es importante recordar que la comunicación entre servicios debe ser ligera. La elección técnica de cada componente es trabajo del estudiante, basado en la necesidad del micro servicio, cada componente debe ser justificado bajo un criterio técnico o cuanto menos lógico. No existe una única respuesta correcta, la argumentación define la calificación.

4.1. MLflow

Este servidor debe estar funcionando constantemente, en caso de caída debe iniciarse automáticamente. Debe tener conexión entre bucket y base de datos de metadata. Acá debe registrar la experimentación que realice buscando el mejor modelo para el conjunto de datos propuesto. El que considere como mejor versión, deberá estar marcado como modelo de producción.

4.2. AirFLow

Esta herramienta es la encargada de orquestar el procesamiento de datos, debe tomar los datos de la base de datos RAW DATAz aplicar el procesamiento definido para utilizar la información en proceso de entrenamiento, almacenando los resultados en ÇLEAN DATA", adicionalmente una vez se define un proceso de entrenamiento, debe ejecutar este proceso, registrando los resultados en MLflow

4.3. Sistema de Archivos - Bucket

Para el registro de artefactos, tal como se vio en clase, propone el uso de un bucket y acceder a el mediante el uso de MLflow. Configure el contenedor o si lo desea use otro método de almacenamiento, puede ser con o sin proveedor cloud.

4.4. Base de Datos - Metadata

Esta base de datos contiene lo relativo a ejecuciones registradas en MLflow; a diferencia del escenario presentado en clase, debe crear una base de datos en un contenedor y no puede ser sqlite. Sugerencia: PostgreSQL

4.5. JupyterLab

Este contenedor es opcional, es el encargado de ejecutar el código de experimentos, por lo cual consume la información existente en la base de datos -CLEAN DATA- y registra los resultados en MLflow. Para la condición de cargue en batch, este contenedor le puede ayudar a seccionar los documentos de ingreso de datos en las cantidades necesarias si así lo desea.

4.6. Base de Datos - CLEAN DATA

Este componente contiene la información que se usará como insumo para la experimentación y entrenamiento de modelos.

4.7. Base de Datos - RAW DATA

Este componente contiene la información sin modificaciones. En este componente se establece la siguiente condición: Se debe establecer una forma de separar los datos en 3 subconjuntos (train, validation, test), sin embargo, el subconjunto "train"**no se debe agregar en un solo cargue de información**. Se debe agregar la información en batch, es decir, **por lotes de datos, en este caso cada 15.000 registros.**

4.8. Inferencia UI

Este componente permitirá usar el mejor modelo de la experimentación, es decir, el que esté configurado como 'Production' en MLflow. Para esto se debe crear un contenedor que permita consumir el modelo registrado en MLflow, solicitar al usuario datos para realizar inferencia y, al presentar resultados, indicar cuál es el modelo que está usando. Para la creación de esta interfaz, no se espera un desarrollo web; se propone al estudiante revisar el framework "Streamlit"(también puede usar Gradio o alguna opción similar), el cual está diseñado para crear interfaces para aplicaciones de inteligencia artificial o ciencia de datos en minutos.

4.9. Observabilidad

Este bloque es agnóstico al resto del sistema y sus conexiones serán determinadas por el estudiante; como mínimo, se espera que se recolecte información de la API que permite realizar inferencia al modelo de mejor desempeño. Este bloque está compuesto por dos servicios, Grafana para la visualización de datos y Prometheus para la recolección de métricas.

4.10. API

Este componente permite realizar inferencia con el modelo de mejor resultado; este elemento debe permitir la recolección de métricas en el endpoint **/metrics** para que Prometheus pueda recolectarlas. Esta API debe consultar el mejor modelo a MLflow, sin cambios en el código