Operaciones de Machine Learning

Proyecto 2
Nivel 2
Orquestación, métricas y modelos

Pontificia Universidad Javeriana
Cristian Javier Diaz Alvarez

18 de marzo de 2025

1. Descripción

El proyecto tiene como objetivo que los estudiantes implementen un entorno de MLOps en una máquina virtual utilizando Docker Compose. Este entorno incluirá servicios esenciales como Airflow para la orquestación, MLflow para el registro de experimentos y modelos, Minio como almacenamiento de objetos para MLflow, y MySQL como base de datos para la metadata de MLflow. Además, los estudiantes trabajarán con una API externa para obtener datos aleatorios que serán utilizados para entrenar un modelo de inteligencia artificial.

Docker como herramienta para construir contenedores que soporten el servicio creado. Si bien Docker permite ejecutar las imágenes de los contenedores construidos, para levantar el sistema completo se requieren instrucciones para construcción de imágenes e implementación de contenedores. Por lo tanto, para probar los servicios creados en conjunto, se exige el uso de Docker Compose.

Docker Compose permite levantar, un conjunto de contenedores, definidos previamente. Esta herramienta, mediante un archivos de configuración tiene la capacidad de construir imágenes y/o levantarlas de manera unificada y controlada. Adicional a los contenedores, es posible establecer configuraciones de montajes de volúmenes, así como de redes de Docker según las necesidades.

Usando los componentes anteriores, se deben desarrollar las funcionalidades descritas, siguiendo la arquitectura propuesta. La entrega consiste en código fuente en repositorio publico (10 %), despliegue del sistema usando docker-compose (10 %) sobre la maquina virtual proporcionada exponiendo las interfaces gráficas, allí se revisará el resultado de las ejecuciones de DAG en airflow (30 %), métricas de entrenamientos y modelos para inferencia en mlflow (30 %), finalmente se realizará proceso de inferencia en la API(20 %). Nota: La arquitectura de referencia propuesta es un mínimo esperado, sin embargo, existe libertad para agregar nuevos servicios si los consideran necesarios

2. Descripción del dataset

Se propone utilizar una variante del conjunto de datos Tipo de Cubierta Forestal. Esto se puede utilizar para entrenar un modelo que predice el tipo de cobertura forestal en función de variables cartográficas. Puede leer más sobre el conjunto de datos **original** aquí y describimos las columnas de datos a continuación:

Column Name	Variable Type	Units / Range	Description
Elevation	quantitative	meters	Elevation in meters
Aspect	quantitative	azimuth	Aspect in degrees azimuth
Slope	quantitative	degrees	Slope in degrees
Horizontal_Distance _To_Hydrology	quantitative	meters	Horz Dist to nearest surface water features
Vertical_Distance _To_Hydrology	quantitative	meters	Vert Dist to nearest surface water features
Horizontal_Distance _To_Roadways	quantitative	meters	Horz Dist to nearest roadway
Hillshade_9am	quantitative	0 to 255 index	Hillshade index at 9am, summer solstice
Hillshade_Noon	quantitative	0 to 255 index	Hillshade index at noon, summer soltice
Hillshade_3pm	quantitative	0 to 255 index	Hillshade index at 3pm, summer solstice
Horizontal_Distance _To_Fire_Points	quantitative	meters	Horz Dist to nearest wildfire ignition points
Wilderness_Area (4 binary columns)	qualitative	0 (absence) or 1 (presence)	Wilderness area designation
Soil_Type (40 binary columns)	qualitative	0 (absence) or 1 (presence)	Soil Type designation
Cover_Type (7 types)	integer	1 to 7	Forest Cover Type designation

Tabla 1: Descripción de variables.

Como puede notar, los datos cualitativos ya han sido codificados usando representación One-hot (por ejemplo, **'Soil_Type'** tiene 40 columnas binarias donde un "1ïndica la presencia de una característica). Para este proyecto usaremos una versión modificada de este conjunto de datos que muestra un formato más crudo. Esto le permitirá practicar sus habilidades en el manejo de diferentes tipos de datos. Puede ver el código para preparar el conjunto de datos aquí

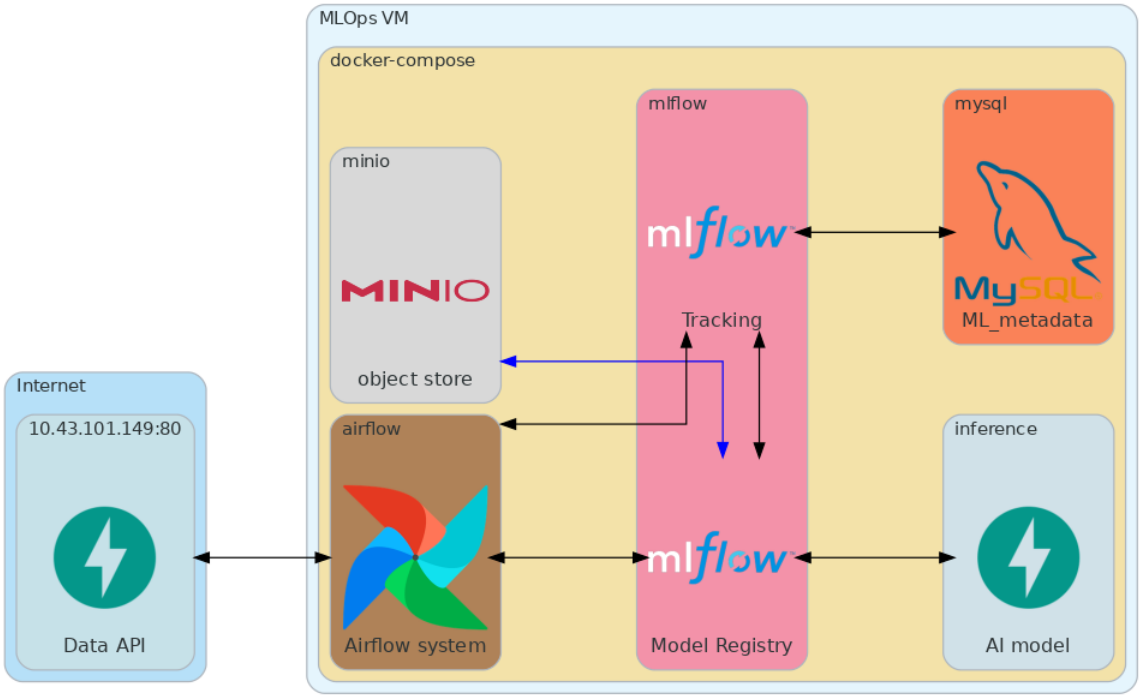
2.1. Cargar el Dataset

Los datos serán obtenidos a través de una API externa expuesta en la maquina virtual asignada al profesor alojada en la dirección IP **IP http://10.43.101.108:80**. Esta API proporcionará un conjunto de datos aleatorios que cambiarán cada 5 minutos. Los estudiantes deberán implementar un mecanismo para recolectar estos datos usando Airflow y utilizarlos para entrenar un modelo de IA en el entorno de MLflow.

El conjunto de datos fue divido en 10 partes (batch), por lo tanto para obtener cada una de esas 10 partes se debe hacer una petición, a la cual solo se solicitará el numero del grupo asignado en la petición. Esta petición retornará una porción aleatoria de los datos del batch actual. A partir de ese momento y durante los 5 minutos siguientes a la petición, se entregará información una porción aleatoria de el batch de información actual, después de esos 5 minutos se cambiará el batch y se repetirá la condición de los 5 minutos nuevamente. Tenga en cuenta que para lograr tener una muestra mínima de los datos debe extraer al menos, una porción de datos de cada batch. Adicionalmente el código de esta API se encuentra disponible en Data P2 está disponible si desean desplegarla y hacer pruebas reduciendo el tiempo entre cambios de batch de data, Adicionalmente si el servicio no estuviera disponible en la máquina especificada, es responsabilidad de los estudiantes desplegar bajo las condiciones originales y realizar conexión a esta API como si fuera una máquina externa

3. Arquitectura

Dentro de los paradigmas de programación, se tiene una intención en común, segmentar los procesos para poder organizar, reutilizar y cambiar cada una de las partes en distintos escenarios. Como resultado, hoy en día es popular la arquitectura de micro servicios, que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros. Esta Arquitectura busca dejar de lado el paradigma monolítico, al no tener componentes que realicen mas de una tarea.



Project 2

La arquitectura del sistema consistirá en la implementación de los siguientes servicios en una máquina virtual utilizando Docker Compose:

- **Airflow**: Se utilizará como orquestador para gestionar y programar los flujos de trabajo relacionados con la recolección de datos, entrenamiento de modelos y registro de experimentos en **MLflow**.
- **MLflow**: Se utilizará para el registro de experimentos, seguimiento de métricas y almacenamiento de modelos. MLflow estará configurado para utilizar **Minio** como almacenamiento de objetos y **MySQL** como base de datos para la metadata.
- **Minio**: Se utilizará como almacenamiento de objetos para **MLflow**. Minio proporcionará un entorno de almacenamiento escalable y distribuido para almacenar los artefactos generados durante los experimentos y entrenamientos de modelos en **MLflow**.
- **MySQL**: Se utilizará como base de datos para la metadata de **MLflow**. **MySQL** almacenará información sobre los experimentos, modelos, métricas y parámetros registrados en **MLflow**, permitiendo un seguimiento y gestión eficiente de los mismos.
- **Inference**: Se utilizará FastAPI para la construcción del API que consume el modelo entrenado en **MLflow**.

Los estudiantes deberán configurar adecuadamente los servicios de Docker Compose para que interactúen entre sí y con la API externa para recolectar datos, entrenar modelos y registrar experimentos

en MLflow. Para finalmente realizar la inferencia usando la API que consume los modelos de MLflow

Este proyecto proporciona una oportunidad para que los estudiantes adquieran experiencia práctica en la implementación de un entorno de MLOps completo, desde la recolección de datos hasta el registro de modelos y experimentos, utilizando herramientas y servicios populares en la industria.

3.1. Bono - Interfaz gráfica

Este componente es el encargado de la interacción con el usuario, solicita y permite el ingreso de información para realizar inferencia para finalmente presentar resultados. Adicionalmente, permite ejecutar cada API anteriormente mencionado. Para la creación de esta interfaz, no se espera un desarrollo web, se propone al estudiante revisar el framework **Streamlit o Gradio**, el cual está diseñado para crear interfaces para aplicaciones de inteligencia artificial o ciencia de datos, en minutos. Es decisión del estudiante definir como sera la interacción entre usuario e interfaz gráfica. Recuerde entregar información de lo que sucede, falla o se realiza de manera exitosa al usuario. Esta API debe estar expuesta en el puerto 8503.