Proyecto Final de Bootcamp de MLOps

Por: Luis Carlos Sánchez Ruiz

Visión General y Planeación del Proyecto

Se abordará el problema de **Housing Prices**. Es un problema de regresión clásico que involucra predicciones numéricas (precios de casas) a partir de varias características. Se obtuvo este clásico problema desde la plataforma Kaggle, donde también se puede encontrar el dataset utilizado (link: https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview).

Fase 1: Definición del Problema y Objetivos

1.1 Descripción del problema:

Se trabaja con el conjunto de datos de **Housing Prices**, que es un clásico en competiciones de Kaggle. El objetivo es predecir el precio de una casa basada en ciertas características (por ejemplo, el número de habitaciones, el tamaño en pies cuadrados, la ubicación, etc.).

- **Objetivo**: Predecir el precio de una casa en función de sus características.
- **Datos**: Se usa el conjunto de datos de **Kaggle** de "House Prices: Advanced Regression Techniques". El dataset incluye columnas como el área, el número de habitaciones, la antigüedad, entre otras, y el precio de la casa.

1.2 Infraestructura necesaria:

Para este proyecto, vamos a usar:

- **Entorno de desarrollo**: Jupyter Notebook.
- **Bibliotecas**:
    - **pandas**, **numpy**: para el manejo de datos.
    - **scikit-learn**: para el preprocesamiento, la creación y evaluación de modelos.
    - **MLflow**: para el tracking de experimentos.

- **FastAPI** y **Uvicorn** (para un despliegue online del modelo).
- **Grafana/Prometheus**: para el monitoreo.

1.3 Diagramas de flujo de datos:

Descripción de un flujo general:

1. **Ingesta de Datos**: Cargar el dataset desde un archivo CSV.
2. **Preprocesamiento**: Limpiar los datos (tratar valores faltantes, convertir variables categóricas a numéricas, etc.).
3. **Entrenamiento**: Entrenamos el modelo de regresión (primero un modelo sencillo como **Linear Regression** y después algo más complejo como **RandomForestRegressor**).
4. **Evaluación**: Evaluamos el modelo con la métrica **RMSE**.
5. **Despliegue**: Desplegamos el modelo en un servicio de API.
6. **Monitoreo**: Monitoreamos el rendimiento del modelo en producción.

1.4 Justificación del Despliegue:

Para el despliegue del modelo se opta por un despliegue **online**, donde el modelo recibe las características de una nueva casa y devuelve su predicción del precio en tiempo real.

Fase 2: Gestión de Experimentos y Modelos

2.1 Sistema de tracking de experimentos:

Se usa **MLflow** para el seguimiento de experimentos. Con MLflow, podremos:

- Registrar los parámetros del modelo (como el tipo de modelo, hiperparámetros, etc.).
- Almacenar las métricas de evaluación (como RMSE).
- Versionar los modelos entrenados.

Pasos iniciales:

1. Instalar MLflow.
2. Configurar un servidor local de MLflow para registrar los experimentos.

2.2 Versionado de modelos:

Cada vez que se entrene un modelo, se le asignará una versión para poder comparar entre diferentes iteraciones.

Fase 3: Orquestación y Pipelines de ML

3.1 Diseño de pipelines reproducibles y escalables:

Se busca crear un pipeline que incluya los siguientes pasos:

- **Preprocesamiento**: Limpiar los datos, manejar valores faltantes, y codificar variables categóricas.
- **Entrenamiento**: Probar varios modelos, por ejemplo, **LinearRegression** y **RandomForestRegressor**.
- **Evaluación**: Medir el rendimiento de cada modelo con **RMSE**.

3.2 Automatización y paralelización:

El pipeline puede ser automatizado usando **MLflow**.

Fase 4: Integración Continua y Pruebas

4.1 Pipeline de CI/CD:

Se planea usar **GitHub Actions** o **GitLab CI** para crear un pipeline de CI/CD que:

- Ejecute pruebas unitarias de preprocesamiento y entrenamiento.
- Automatice el entrenamiento del modelo y su evaluación.
- Despliegue el modelo de forma automática.

Fase 5: Despliegue del Modelo

5.1 Despliegue:

Se planea optar por un **despliegue online** con **FastAPI**, ya que queremos que las predicciones sean en tiempo real.

Pasos:

1. Crear un servidor FastAPI que reciba los datos de entrada (características de la casa) y devuelva el precio predicho.
2. Desplegar el servidor con **Docker** para hacerlo portátil y fácil de ejecutar.

Fase 6: Monitoreo del Modelo

6.1 Logging y Alertas:

Para el monitoreo, se planea usar **Prometheus** y **Grafana**:

- **Prometheus**: Recogerá métricas de rendimiento (como el **RMSE**).
- **Grafana**: Visualizará las métricas y enviará alertas si detectamos que el modelo se está degradando con el tiempo.

6.2 Data Drift y Concept Drift:

Se planea implementar un sistema para detectar si las distribuciones de los datos de entrada han cambiado con el tiempo (lo que puede afectar al rendimiento del modelo).