Deploying models in production

Giorgio Alfredo Spedicato, PhD FCAS CSPA C.Stat[1]

Leitha SRL, Unipol Group

2024-11-01

Disclaimer

Intro

- Deploying a model in production is a complex task that requires a deep understanding of the model, the data and the infrastructure.
- Actuaries were traditionally involved in the first two aspects, but the third one is becoming more and more important.
- Modern Python frameworks make the third aspect easier.

MLOps: Enhancing the ML Model Lifecycle

- **Definition**: MLOps integrates practices to automate and manage the entire machine learning model lifecycle, improving deployment efficiency and reliability in production.
- **Why It's Needed**: Bridges the gap between development and production, enabling better collaboration between data scientists and engineers for models that are more performant, scalable, and maintainable.
- **Key Benefits**: Unified workflow reduces errors, accelerates deployment, and provides continuous monitoring to keep models up-to-date and accurate.

MLOps: instruments

- **Model training**: Python pipelines with specified requirements.
- **Version control**: Git, GitHub, GitLab
- **Model deployment**: FastAPI, Streamlit, Docker

Project presentation

- **Objective**: Deploying an insurance quote model in production
- **Tools**:
  - **Python pipelines**: to train the frequency, severity and pure premium models
  - **Git**: to version control the code
  - **Docker**: to create a container with the models
  - **FastAPI**: to deploy the models
  - **Streamlit**: to create a user interface

Python pipelines

Structure

- Ingests the data, and clean
- Fits the models, saves them
- Assess the performance of the models, uses MLflow to log the results and artifacts

Requirements

- Python packages specified in a `requirements.txt` file
- The `requirements.txt` file is used to create a virtual environment
- The Dockerfile uses the virtual environment to create a container

MLflow

- MLflow is an open source platform for managing the end-to-end machine learning lifecycle.
- It is used to log the results and artifacts of the models
- It is organized in experiments and runs

Streamlit walkthrough

Why streamlit?

- Streamlit is an open-source app framework for Machine Learning and Data Science projects.
- It creates a user interface for the models
- It is easy to use and to deploy

Streamlit code

- the st. functions are used to create the user interface
- the session state is used to track user's choices
- the models are loaded, cached and used to make predictions

FastApi walkthrough

Why FastAPI?

- FastAPI is a modern framework for building APIs, e.g. for deploying machine learning models.
- It allows to expose the models as a REST API, so that they can be used by other applications e.g. by rest
- It uses Python type hints to validate the input and output of the API

Structure of a FastAPI app

- The app is defined in a Python file
- The models are loaded and used to make predictions
- The app is run with `uvicorn`

Docker

Why Docker?

- Docker is a platform for developing, shipping, and running applications.
- It allows to create containers with the models and the dependencies
- It is used to deploy the models in production

Dockerfile

- The Dockerfile is used to create the container
- Classical steps are:
    - load a base python image
    - copy models and code
    - install requirements
    - Run the app

Docker commands

- `docker build -t myimage .` to build the image
- `docker run -d --name mycontainer -p 8000:8000 myimage` to run the container that exposes the app on port 8000
- `docker stop mycontainer` to stop the container

References I