Text Summarizer API: A Full-Stack ML Deployment Project

Abrar Zahin

This project demonstrates the end-to-end deployment of a text summarization service using modern machine learning and DevOps tools. At its core, the application uses a pre-trained transformer-based model, specifically the `sshleifer/distilbart-cnn-12-6` model from Hugging Face, to generate concise summaries from longer text inputs. The goal of this project is not only to showcase the ability to build an ML application, but also to serve it as a scalable, production-ready API.

The application is built using FastAPI, a high-performance Python web framework ideal for serving machine learning models. The summarization pipeline is defined within a FastAPI app, with a single POST endpoint `/summarize` that takes raw text as input and returns a generated summary in JSON format. This simple interface allows seamless interaction from other services, scripts, or front-end applications.

To ensure portability and reproducibility, the entire app is containerized using Docker. The Dockerfile defines a minimal Python environment, installs the required dependencies, and launches the FastAPI app using Uvicorn. With Docker, the summarization API can be run on any system that supports containers, regardless of the host OS or Python setup.

In addition to containerization, the project is configured for Kubernetes deployment. Kubernetes manifests are included to define a deployment with two replicas of the API, a LoadBalancer service to expose it externally, and a Horizontal Pod Autoscaler (HPA) that adjusts the number of running pods based on CPU usage. These configurations simulate a production environment and demonstrate readiness for large-scale traffic and system health monitoring.

While the actual deployment and model used in this project are limited to a Colab environment and a lightweight model, the architecture is designed to be scalable to much larger models and cloud-native infrastructure. In a real-world setting, this same pipeline could serve 7B or even 70B parameter models hosted via vLLM or Hugging Face Text Generation Inference, assuming adequate GPU resources.

Overall, this project showcases practical skills in machine learning model serving, API design, containerization, and orchestration. It demonstrates not just the ability to build a model, but to deploy, expose, and scale it in a production-like setting using industry-standard tools such as Docker and Kubernetes. This kind of full-stack ML engineering project is representative of what modern MLOps looks like in practice.