ML Deployment Architecture Document

1. Overview of the Deployment Architecture

This document describes the modern three-tier architecture used to deploy our Machine Learning model, transitioning from a local Python script to a scalable, containerized web application accessible via Hugging Face Spaces.

Core Components and Responsibilities

Component | Technology | Role

Frontend/Client | Streamlit (or Gradio) | Handles User Input, formats requests, and displays results. Runs on port 7860 (exposed).

Backend/Server | FastAPI | Acts as the API Gateway. Loads the model, handles data validation, executes prediction, and formats the final response. Runs internally on port 8001.

Execution Environment | Docker | Encapsulates the entire application.

Model Artifact | Scikit-learn/Joblib | The trained, serialized model.

2. Communication Flow in a Real-World Application

The entire architecture is deployed within a single Docker container on Hugging Face Spaces. Communication relies on internal networking.

Steps:

1. Startup → start.sh launches FastAPI in background.

2. Readiness Check → start.sh uses curl to check /health.

3. Launch UI → start.sh launches Streamlit on port 7860.

4. User Request → Streamlit sends POST to 127.0.0.1:8001.

5. Prediction → FastAPI loads model & predicts.

6. Response → FastAPI returns JSON.

7. Display → Streamlit shows output.

Note: Use explicit loopback 127.0.0.1.

3. FastAPI: Key Features and Importance

- Asynchronous (ASGI)

- Data validation (Pydantic)

- Automatic docs

- Efficient model loading

Interview Q&A;:

Q: Main advantage of FastAPI?

A: High performance + validation.

Q: Pydantic purpose?

A: Data schemas and validation.

Q: What is Uvicorn?

A: ASGI server hosting FastAPI.

Q: How is model loading handled?

A: Loaded once at startup.

4. Docker: Key Features and Importance

- Reproducibility

- Isolation

- Portability

Interview Q&A;:

Q: Image vs Container?

A: Image is blueprint; container is running instance.

Q: Purpose of EXPOSE?

A: Documents listening ports.

Q: Why custom start.sh?

A: Manage multiple processes and readiness checks.

Q: What does CMD do?

A: Default command on container start.

Q: Why slim base image?

A: Smaller, faster deployments.