# Step 3: Building & Serving ML Models with FastAPI

Tutor: Haythem Ghazouani

## Contents

# 1 Overview

FastAPI is a modern, high-performance framework for building APIs with Python. In this module, we use it to turn our trained Machine Learning pipeline into a usable service.

**Implementation Note**

Reference implementation: `code/app.py`

# 2 Step 1: Installation

To run the API, you need the framework and an ASGI server (Uvicorn). We also need `python-multipart` for file uploads.

```
pip install fastapi uvicorn python-multipart
```

Ensure these are in your `requirements.txt`.

# 3 Step 2: Understanding the Logic (app.py)

Our application follows a standard pattern:

## 3.1 1. Model Loading

We load the pickle file only **once** when the application starts, not for every request.

```
MODEL_PATH = "data/best_model_pipeline.pkl"
if os.path.exists(MODEL_PATH):
    model = joblib.load(MODEL_PATH)
```

## 3.2 2. Input Validation (Pydantic)

We define a Class `CustomerData` that inherits from `BaseModel`. This acts as a strict contract. If a user sends "Male" as an integer, FastAPI will raise a `422 Validation Error`.

## 3.3 3. Endpoints

- **/health (GET):** Returns status 200 if the model is loaded. Used by load balancers (like Kubernetes) to check if the pod is alive.
- **/predict (POST):** Accepts JSON data for a single customer. Returns prediction and probability.
- **/predict_batch (POST):** Accepts a `.csv` file using `UploadFile`, processes it with Pandas, and returns the result.

# 4 Step 3: Running the Server

Execute the following command from the root directory:

```
uvicorn code.app:app --reload
```

- `code.app`: Refers to the module `app.py` inside `code/` folder.
- `:app`: The instance of `FastAPI` created in the code.
- `-reload`: Automatically restarts the server when code changes (Dev mode only).

# 5 Step 4: Testing with Swagger UI

FastAPI automatically generates interactive documentation.

1. Open `http://localhost:8000/docs` in your browser.
2. Find the **POST /predict** endpoint.
3. Click **Try it out**.
4. Modfy the JSON body and click **Execute**.

**Common Pitfall**

If you see "Internal Server Error", check your terminal. It usually means the input data format didn't match what the model pipeline expects (e.g., "Gender" vs "gender").

# 6 Step 5: Testing with Code (Client-Side)

You can verify your API programmatically:

```python
import requests

url = "http://localhost:8000/predict"
payload = {
    "Geography": "France",
    "Gender": "Male",
    "CreditScore": 600,
    "Age": 40,
    "Tenure": 3,
    "Balance": 60000.0,
    "NumOfProducts": 2,
    "HasCrCard": 1,
    "IsActiveMember": 1,
    "EstimatedSalary": 50000.0
}
response = requests.post(url, json=payload)
print(response.json())
```