Week 10 Lab: Building ML APIs with FastAPI

CS 203: Software Tools and Techniques for AI

Duration: 3 hours

Lab Objectives

1. **FastAPI Basics**: Build your first Hello World API.
2. **Data Validation**: Use Pydantic to secure inputs.
3. **Model Serving**: Wrap a scikit-learn model in a REST API.
4. **Testing**: Verify your API works correctly.

Prerequisites:

- `pip install fastapi[standard] scikit-learn joblib pytest httpx`

Exercise 1: Hello FastAPI (30 min)

Goal: Get a server running and explore Swagger UI.

1. Create `main.py`.
2. Define a root endpoint `GET /` returning `{"message": "ML API is running"}`.
3. Define an endpoint `GET /add/{a}/{b}` that returns the sum of two integers.
4. Run with `fastapi dev main.py`.
5. Open `http://127.0.0.1:8000/docs`.
    - Try the "Execute" button for both endpoints.

Exercise 2: Pydantic Validation (45 min)

Goal: Define a robust schema for an ML model input.

Imagine a "House Price Prediction" model.
Inputs: `square_feet` (int), `bedrooms` (int), `zipcode` (str).

1. Define a `HouseFeatures` Pydantic model.
   - `square_feet`: Must be > 0.
   - `bedrooms`: Must be >= 0.
2. Create a `POST /predict_price` endpoint.
   - Accept `HouseFeatures` as JSON body.
   - Return a dummy prediction (e.g., `square_feet * 100`).

Test: Send invalid data (negative size) via Swagger UI and observe the 422 Error.

Exercise 3: Serving a Real Model (60 min)

Goal: The core MLOps task.

1. **Train**: Create `train_model.py`.
   - Train a `RandomForestClassifier` on the **Iris dataset**.
   - Save it using `joblib.dump(model, "iris_model.pkl")`.
   - Run this script once.
2. **Serve**: Update `main.py`.
   - Load `iris_model.pkl` on startup (global variable).
   - Create `POST /predict_species`.
   - Input: `sepal_length`, `sepal_width`, etc. (Use Pydantic!).
   - Output: The predicted class name (Setosa/Versicolor/Virginica).

Exercise 4: Testing (30 min)

Goal: Automate API verification.

1. Create `test_api.py`.
2. Use `fastapi.testclient.TestClient`.
3. Write a test `test_predict_species()`:
   - Send a sample JSON payload.
   - Assert status code is 200.
   - Assert "class" is in the response body.
4. Run `pytest`.

Submission

Deliverables:

1. `main.py` : Your FastAPI application.
2. `train_model.py` : The training script.
3. `test_api.py` : Your tests.
4. `requirements.txt` : Dependencies.

Challenge (Bonus):

Add a `/health` endpoint that checks if the model is loaded correctly and returns 200 OK or 500 Error.