Stage 13: Productization — Homework Sheet

Assignment Overview

In this homework, you will prepare a final repo folder, as if it were your final project, for **reuse, clarity, and handoff**.
You will apply the productization concepts from the lecture and reading text to:

- Organize your repository
- Clean and modularize code
- Create final documentation
- Demonstrate basic interactive and API access to your analysis.

In the lecture, we learned how to structure projects, pickle models, expose endpoints via Flask, build dashboards with Streamlit/Dash, and prepare code for handoff.

Now, you will adapt these concepts to finalize your own project. In this homework you will practice doing thia before doing this in your actual final project.

Tasks

1. Project Organization

- Ensure your folder/ uses a clean folder structure:

```
data/
notebooks/
src/
reports/
model/
README.md
```

- Move reusable functions from notebooks into `/src/`.
- Remove exploratory cells and add comments to final notebooks.
- For API or dashboard, a minimal example set of files might include:

```
app.py
model.pkl
requirements.txt
README.md
```

---

2. README and Documentation

- Write a clear `README.md` including:
    - Project overview and objectives
    - How to rerun scripts/notebooks
    - Assumptions, risks, and lifecycle mapping
    - Instructions for using APIs or dashboards
- Include a stakeholder-ready summary (PDF or slide) describing results, assumptions, and next steps.

---

3. Model Persistence

- Pickle or save your final model in `/model/`.
- Include a notebook cell that demonstrates reloading the model and using it for a test prediction.

---

4. API Demonstration

- Create a **Flask endpoint** to serve predictions:
    - `POST /predict` with JSON features
    - `GET /predict/<input1>` for single feature input
    - `GET /predict/<input1>/<input2>` for two features
    - `GET /plot` to return a simple chart or image
- **Add error handling** for invalid inputs (e.g., missing features, wrong types)
- Include a `requirements.txt` for reproducibility
- Demonstrate calling the API from the notebook using `requests`
- Provide testing evidence (screenshots, curl logs, or notebook output) to show it works

---

5. Optional Dash / Streamlit Dashboard

- Build a small interactive dashboard with either Streamlit or Dash:
    - Allow users to input features
    - Display model predictions
    - Display simple charts or tables
    - Launch the dashboard from an external terminal window
    - Include error handling for invalid inputs
    - Include `requirements.txt` for reproducibility
    - Demonstrate locally with testing evidence

---

6. Handoff Readiness

- Check that the repository can be cloned and run on a fresh environment.
- Ensure all scripts, notebooks, models, and outputs are reproducible.
- Verify that another student can follow your README to run the project end-to-end.

Submission Guidelines

- Commit all files to your GitHub repo with clear commit messages.
- Include all final notebooks, scripts in `/src/`, pickled models in `/model/`, and outputs in `/reports/`.
- Ensure your README and stakeholder summary are complete.
- Optional: Include automated .bat scripts to run your flask app or dashboard app and/or to rerun the full pipeline.

Grading Rubric

Task	Points
Project folder structure	10
Code modularization and notebook cleanup	15
README completeness	15
Pickled model demonstration	10
Flask API functionality	15
Optional dashboard (Streamlit/Dash)	10
Error handling in API/dashboard	5
Testing evidence (screenshots, logs)	5
Handoff readiness and reproducibility	15

Total: 100 points

Stretch Goals (Optional)

- Add authentication to your API.
- Add logging to your pipeline for traceability.
- Include automated batch scripts to generate reports or predictions periodically.
- Prototype a more complex interactive dashboard with multiple inputs and visualizations.
