

Software Engineering Practices for Data Scientists

Module 4: Software Engineering, Reproducibility, and Deployment Basics

Dr. Yves J. Hilpisch¹

March 27, 2026



**THE DATA
SCIENTIST**

¹Get in touch: <https://linktr.ee/dyjh>. Web page: <https://thedata scientist.dev>. Research, structuring, drafting, and visualizations were assisted by GPT 5.x as a co-writing tool under human direction. Comments and feedback are welcome.

Preface

This preface explains how Module 4 completes the core arc of *The Data Scientist*. The earlier books — *Python Programming Foundations for Data Science* [2], *Python Data Analysis, Visualization, and Storytelling* [3], and *Statistics, Machine Learning, and Model Evaluation* [4] — teach Python, data analysis, and baseline modeling. This final module, *Software Engineering, Reproducibility, and Deployment Basics* [5], turns those separate skills into cleaner, more reproducible, and more visible project work.

Many beginners can already write notebooks that answer a question. Fewer can turn those notebooks into a small project that someone else can run, review, and understand. Module 4 focuses on that gap.

The goal is not to turn the reader into a full-time software engineer. It is to build working habits that make data-science projects easier to trust, share, and present:

- organizing folders and files with intention,
- using Git and GitHub calmly rather than fearfully,
- separating exploration from reusable code,
- checking data and logic with small tests and assertions,
- and packaging the result as a portfolio artifact.

This module also introduces lightweight dashboards and deployment awareness. The scope remains practical. The reader does not need to become a cloud specialist. They do need to understand why environments, app entry points, and clear documentation matter in real projects.

The study rhythm remains familiar:

- read a short section,
- run or adapt a small example,
- reflect on what would make the work easier to reuse,
- and save the result in a project folder that could be shown publicly.

By the end of this book, the target outcome is simple. The reader should be able to say: this project is not only interesting; it is also organized, documented, reproducible, and ready to share.

Contents

Preface	i
I Reproducible Project Work	1
1 Project Structure for Data Science Work	3
1.1 Why Structure Matters Early	4
1.2 A Minimal Project Skeleton	4
1.3 Naming and Parameters	5
1.4 Paths With <code>pathlib</code>	5
1.5 Raw vs Derived Data	5
1.6 Quick Start Routine	5
1.7 Reproducibility as a Habit	6
1.8 Where We Are Heading Next	6
2 Version Control with Git and GitHub	7
2.1 Why Commits Matter	7
2.2 One-Time Setup	8
2.3 The Daily Loop	8
2.4 Writing Helpful Messages	8
2.5 A Small Working Command Set	8
2.6 Ignore What Should Not Be Tracked	9
2.7 GitHub as a Portfolio Surface	9
2.8 First Push to GitHub	9
2.9 Branches Without Drama	9
2.10 Where We Are Heading Next	10
3 From Notebook to Reusable Project	11
3.1 When a Notebook Is Enough	11
3.2 Think in Three Layers	12
3.3 A Small Helper Module	12
3.4 Refactor One Cell at a Time	12
3.5 Keeping Responsibilities Clear	13
3.6 Execution Paths	13
3.7 Notebook Hygiene Tips	13
3.8 Where We Are Heading Next	13
4 Testing, Assertions, and Reliability	14
4.1 Assertions as Local Guards	14
4.2 What to Test First	15
4.3 Data Validation Mindset	15
4.4 Tiny, Runnable Tests	15

4.5	Checks Before Models	16
4.6	Small Tests, Real Value	16
4.7	Safety Nets for Dashboards	16
4.8	Where We Are Heading Next	16
II Delivery and Professional Packaging		17
5	Environments and Dependency Hygiene	19
5.1	Why Environments Matter	20
5.2	One Visible Install Path	20
5.3	A Minimal Local Workflow	20
5.4	Dependency Hygiene	20
5.5	What to Put in <code>requirements.txt</code>	20
5.6	Refreshing the Environment	21
5.7	Managing Differences Across Machines	21
5.8	Local and Colab Thinking	21
5.9	Where We Are Heading Next	21
6	Interactive Dashboards for Data Work	22
6.1	What a Small Dashboard Is For	23
6.2	Plan the Page Before Coding	23
6.3	A Minimal Streamlit Example	23
6.4	Keep It Fast With Caching	24
6.5	Running the App	24
6.6	Handling Empty Results Gracefully	24
6.7	Choosing Good Interactions	24
6.8	Document the App in the README	24
6.9	Where We Are Heading Next	24
7	Deployment Awareness and Practical Boundaries	26
7.1	What Deployment Means	27
7.2	One-Step Sharing Options	27
7.3	A Tiny Dockerfile Sketch	27
7.4	Why Containers Matter	27
7.5	Practical Boundaries	27
7.6	Talking About Environment Variables	28
7.7	Talking About Deployment Honestly	28
7.8	Where We Are Heading Next	28
8	Capstone: Interactive Data Science Portfolio Project	29
8.1	Capstone Dataset and Task	30
8.2	Required Deliverables	30
8.3	Suggested Repository Shape	30
8.4	A Manageable Capstone Path	31
8.5	Dashboard Scope	31
8.6	Professional Summary	31
8.7	Where We Are Heading Next	32
Glossary		34
Epilogue		36

List of Figures

1.1	A small project layout separates data, reusable code, figures, and documentation instead of storing everything in one notebook.	4
3.1	A healthy transition path: explore ideas in a notebook, then extract stable logic into reusable modules.	12
6.1	A lightweight dashboard creates a short loop from data to filtering, visual inspection, and action.	23
8.1	Professional packaging means more than code: the environment, documentation, narrative, and demo all support the same project story.	31

Contact

Software Engineering Practices for Data Scientists

Module 4: Software Engineering, Reproducibility, and Deployment Basics



Get in touch:

linktr.ee/dyjh

thedata scientist.dev

© 2026 Dr. Yves J. Hilpisch — All rights reserved.