INNOVATION AI LAB

INO AI LAB

[AI EDUCATION • COURSE MATERIAL]

Data Science with Python & AI

Pandas, scikit-learn, modern ML workflows

LEVEL	DURATION	LESSONS
Intermediate	**8 hours**	**15**

[00]

Table of Contents

[LESSON 01]

The Modern Python Data Stack

The 2026 baseline stack is Python 3.12+, uv for package management, Polars or Pandas 2.x for dataframes, DuckDB for in-process SQL, scikit-learn for classical ML, PyTorch 2.5 for deep learning, and Marimo or Jupyter for notebooks. Polars is now the default for new projects — it's 5–10x faster than Pandas on most operations and has a cleaner API. Keep Pandas for legacy notebooks and tutorials.

DuckDB has quietly become the most important new tool in the stack. It runs SQL on Parquet, CSV, or in-memory dataframes without a server, handles datasets larger than RAM, and integrates seamlessly with Polars and Pandas. Replace your 'read CSV into memory then filter' pattern with DuckDB queries — your scripts will run on datasets 100x larger with no code changes.

// KEY TAKEAWAYS

- Polars > Pandas for new projects.
- DuckDB enables out-of-core SQL with zero setup.
- Use uv instead of pip/conda for env speed.

[LESSON 02]

From CSV to Insight in 10 Minutes

A productive first pass on any dataset follows the same script: load with `pl.read_csv` or `duckdb.read_csv`, inspect `df.describe()` and `df.null_count()`, plot histograms of numeric columns and bar charts of categoricals, look for outliers, and write down five questions you'd want answered. Spend 80% of the first session understanding the data, not modeling it. Skipping this step is the #1 source of broken models downstream.

Use a notebook with Marimo or Jupyter, but immediately refactor reusable code into `.py` modules. Notebooks rot fast — they break when packages update, they hide state bugs, and they don't diff cleanly in git. The teams that ship reliable data science write 10% of code in notebooks (exploration) and 90% in modules and tests (production).

// KEY TAKEAWAYS

- 80% of first session: understand the data.
- Five written questions beat ten plots.
- Move code from notebook to module fast.

[LESSON 03]

Feature Engineering Still Wins

For tabular data, feature engineering beats model choice almost every time. Start with domain-driven features: ratios between columns, time-since-last-event, rolling aggregates, target encoding for high-cardinality categoricals. Tools like Featuretools and tsfresh can auto-generate hundreds of features, but hand-crafted features informed by a subject expert still win on real-world datasets.

Avoid the classic leakage traps: don't compute features using future data, don't fit encoders on the full dataset before splitting, and don't include the target indirectly via a derived column. Use scikit-learn Pipelines to bind preprocessing and modeling together so the same transformations apply to train, validation, and inference data — this single discipline prevents most production surprises.

// KEY TAKEAWAYS

- Feature engineering > model choice on tabular data.
- Talk to a domain expert for the best features.
- Use Pipelines to prevent leakage and drift.

[LESSON 04]

From Notebook to Production

A model is not done when accuracy looks good — it's done when it's deployed, monitored, and rolled back safely. Wrap your trained model with FastAPI or LitServe, containerize with a slim Python base image, and deploy on Fly.io, Modal, or your cloud of choice. Add request logging, latency metrics, and a simple drift detector (e.g., compare incoming feature distributions to training).

Plan for model decay from day one. Tabular models often need retraining every 3–6 months as data drifts. Build retraining as a scheduled job with the same code path as training, gate deployment on eval performance, and keep the previous version warm for instant rollback. The hardest production lesson: champion-challenger comparisons in shadow mode catch 90% of regressions before users do.

// KEY TAKEAWAYS

- A model isn't done until it's deployed and monitored.
- Plan retraining and rollback from day one.
- Shadow-mode challengers catch most regressions.

[NEXT]

Keep Going

You've completed this course material. The real learning starts when you apply what you've read. Pick one idea from this PDF and run a small experiment this week. Document what worked and what didn't. Share your findings with the community.

Explore more free courses, daily AI tips, and curated tools at:

innovationailab.com

Have feedback or want to suggest a topic? We read every message.

hello@innovationailab.com

— Innovation AI Lab Team —

// part of LumiLife Tech