Mono-repositories in Python

What, When & How?

Avik Basu

About Me

- Staff Data Scientist at Intuit
- Engineering + Data Science
- Love PS5 games
- Soccer + Tennis
- Driving is therapy!

What is a mono-repo?

What is a mono-repository?

- a.k.a. Monolith
- A single code base for multiple projects
- Projects can be
    1. Libraries
    2. Applications
    3. Scripts and automation
    4. Explorations and experiments

What could go wrong with a mono-repo?

Quite a few things!

Pretty important ones too :)

- Versioning decisions
- CI/CD complexity
- Merging and conflicts
- Code ownership issues
- Need for specialized tools!!
- Unintended code breakage

Then why does it still matter??

Mono-repos

When does it make sense?

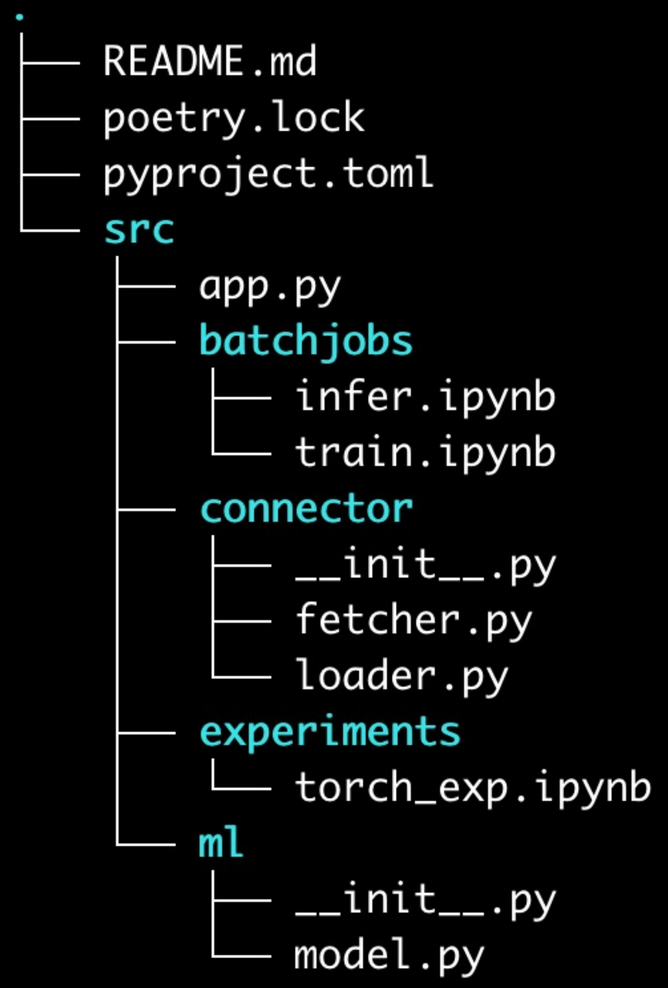
- Inter-related projects with a common goal
- Small to medium level code-base
- Early-stage projects and ideas
    - Better development velocity
    - Easier promotion
- Different components have similar change rate
- System-level extensions, e.g. C/C++/Rust

How about we look at a use-case?

Detecting Fraud in transactions

What do we need?

- Develop a web service to detect fraud transactions in real-time
- Needs connection with external storage systems
- Batch training and inference jobs
- Need some room for model experimentation



Someone from another team says…

We have two options

To be considered

A. Package libraries in separate repositories

- Multiple places to look for
- Harder if more libraries come out of the project
- What if there is an ask for a new UI component?

B. Package libraries inside the main project

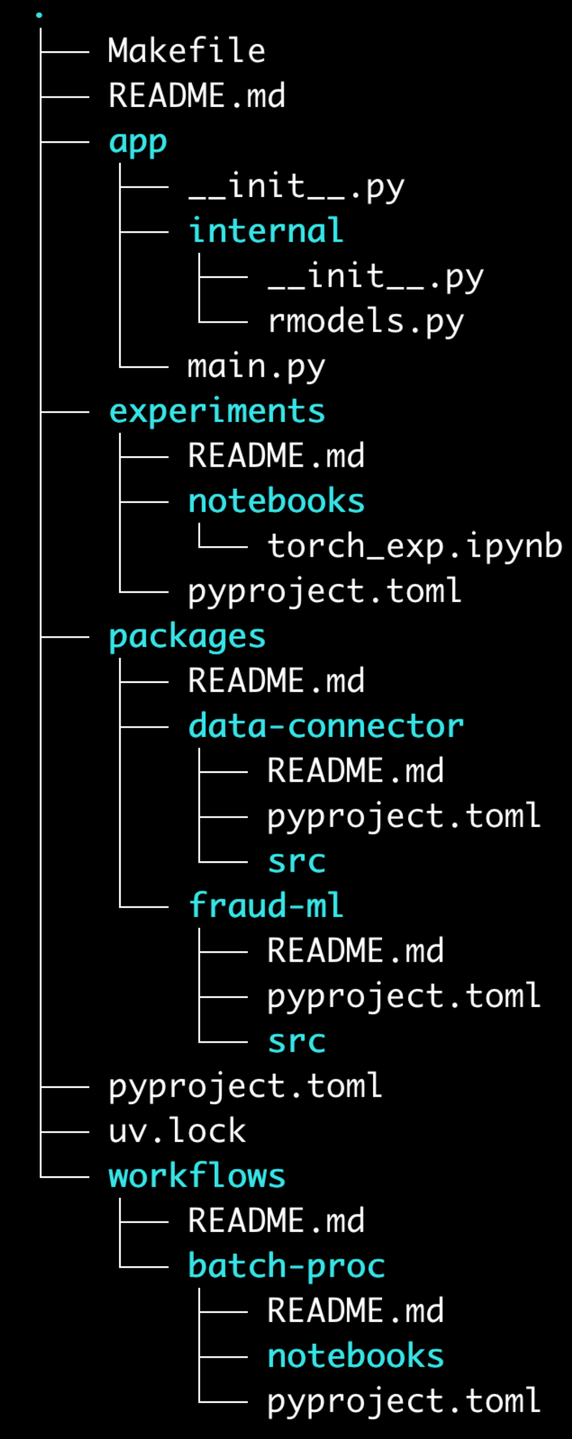
- Every connected component is in one place
- Adding a new project for the same common goal is not a problem

Small !PIVOT! is needed

- 2 libraries
  - 1 core library containing ML models for fraud detection (fraud-ml)
  - 1 supporting library for connecting to different data sources (data-connector)
- 1 microservice (fraud-detect)
  - serves real-time inference
- 1 workflow project
  - Contains batch training and inference jobs (batch-proc)
- 1 experimentation project (experiments)
  - For ad-hoc experiments and analysis

Structure

As a mono-repo



Version management

For multiple packages in the repo

A. One version for the whole repo

- Simpler to manage
- Easier to keep track
- Unnecessary updates for libraries with no changes

B. Each library has its own version

- No unnecessary updates
- Difficult to maintain
- Essential to have a compatibility matrix
- Preferred to use **git submodules**

One version

Example changelog

[0.1.1] - 2025-02-06

Features

- **fraud-ml:** Added a new ML model
- **fraud-detect app:** Working example of service

Changed

- **fraud-detect app:** Improved refactoring of modules.

Fixes

- **fraud-ml:** Add type hints

[0.1.0] - 2025-02-01

Features

- **fraud-ml:** Initial release with structure.
- **data-connector:** Introduced data fetching
- **fraud-detect:** First release, featuring integration with fraud-ml and data-connector.

Multiple versions

Example compatibility matrix

Application	fraud-ml	data-connector
1.0.x	<1.1	<0.3
1.1.x	>=1.1.x,<1.2	>=0.3,<0.4.x
2.0.x	>= 1.2.x	>=0.4.x

Python Mono-repos

In what ways they can be useful?

- One place to look through everything
- Can improve development velocity (sometimes)
- Easier promotion of open-source projects (think GitHub ⭐)
- Better management of other language extensions, e.g. C/C++/Rust
- Popular mono-repositories
    - pytorch-lightning
    - azure-sdk-for-python

Thank You! 🙏

Github Repo

Connect on LinkedIn