Data Versioning Motivation

INTRODUCTION TO DATA VERSIONING WITH DVC



Ravi Bhadauria
Machine Learning Engineer

datacamp

What is Data Versioning?

- Definition
  - Monitors data changes over time
  - Snapshots data over iterations
  - Similar to code versioning
- Benefits
  - Retrieval and scrutiny
  - Data consistency, accountability, and lineage
- Applications
  - Data Science and Machine Learning
  - Data Engineering
  - Financial Analysis, Auditing and Compliance

Data vs Code Versioning

Code Versioning

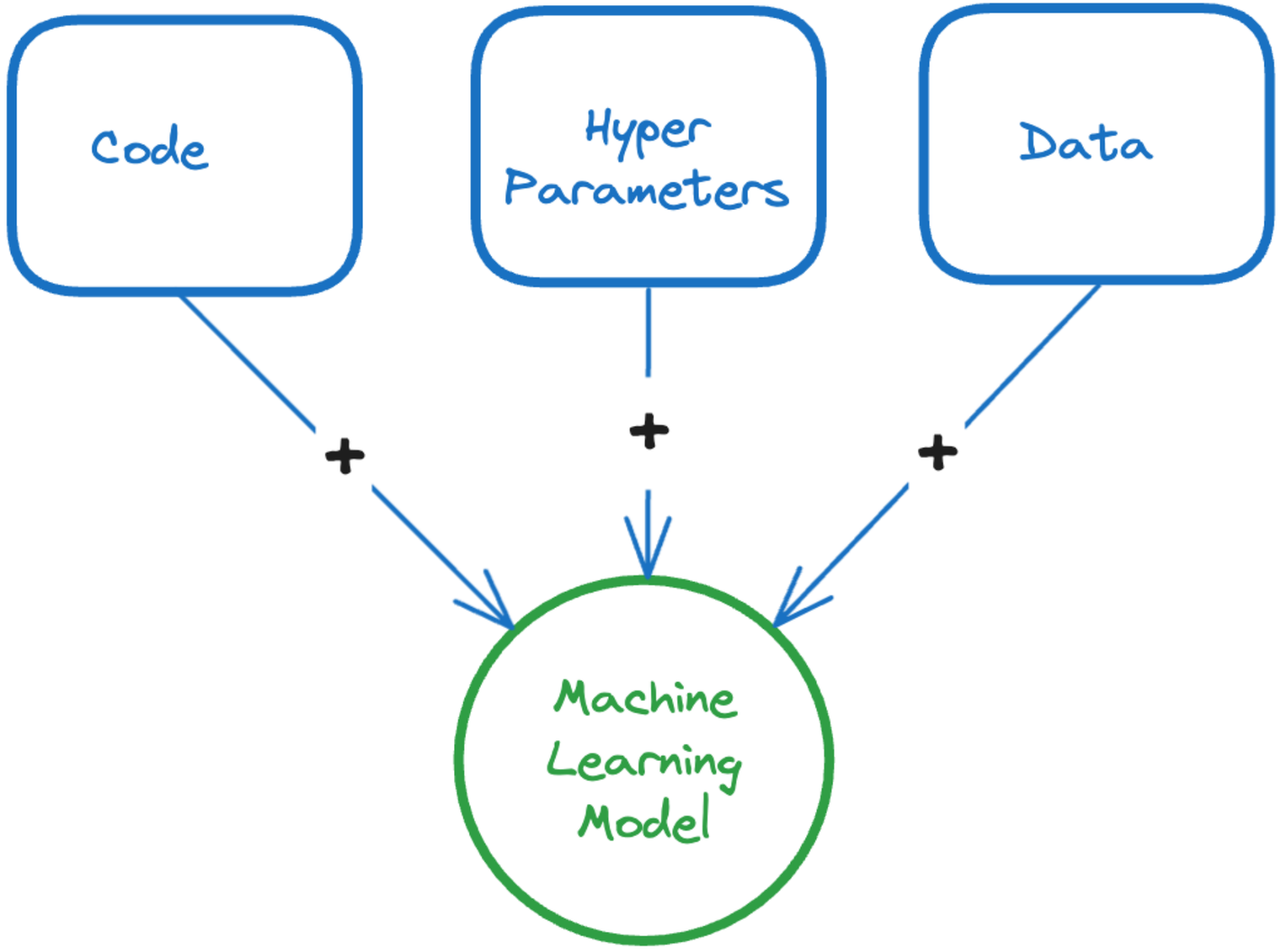
- Well known in software development
- Uses tools like **Git** to do decentralized version control
- Easier to manage as codebases are small

Data Versioning

- Relatively new (SciDB proposed in 2012)
- Toolchains like **DVC** are used in conjunction with Git
- Relatively difficult to manage due to large dataset size

[1] doi: 10.1109/ICDE.2012.102

Why Data Versioning in ML?



Dataset influence

Dataset A

	Booking_ID	number of adults	number of children	number of weekend nights	...	booking status
0	INN32220	2	0	1	...	0
1	INN19707	2	0	0	...	0
2	INN36276	2	0	1	...	0
3	INN02246	2	0	0	...	1
4	INN27306	1	0	0	...	0

Dataset B

	Booking_ID	number of adults	number of children	number of weekend nights	...	booking status
0	INN33507	2	0	0	...	0
1	INN29646	1	0	0	...	0
2	INN34621	2	0	2	...	0
3	INN19236	2	0	0	...	0
4	INN02991	2	0	0	...	0

Dataset influence

Hyperparameters are kept consistent, dataset changed

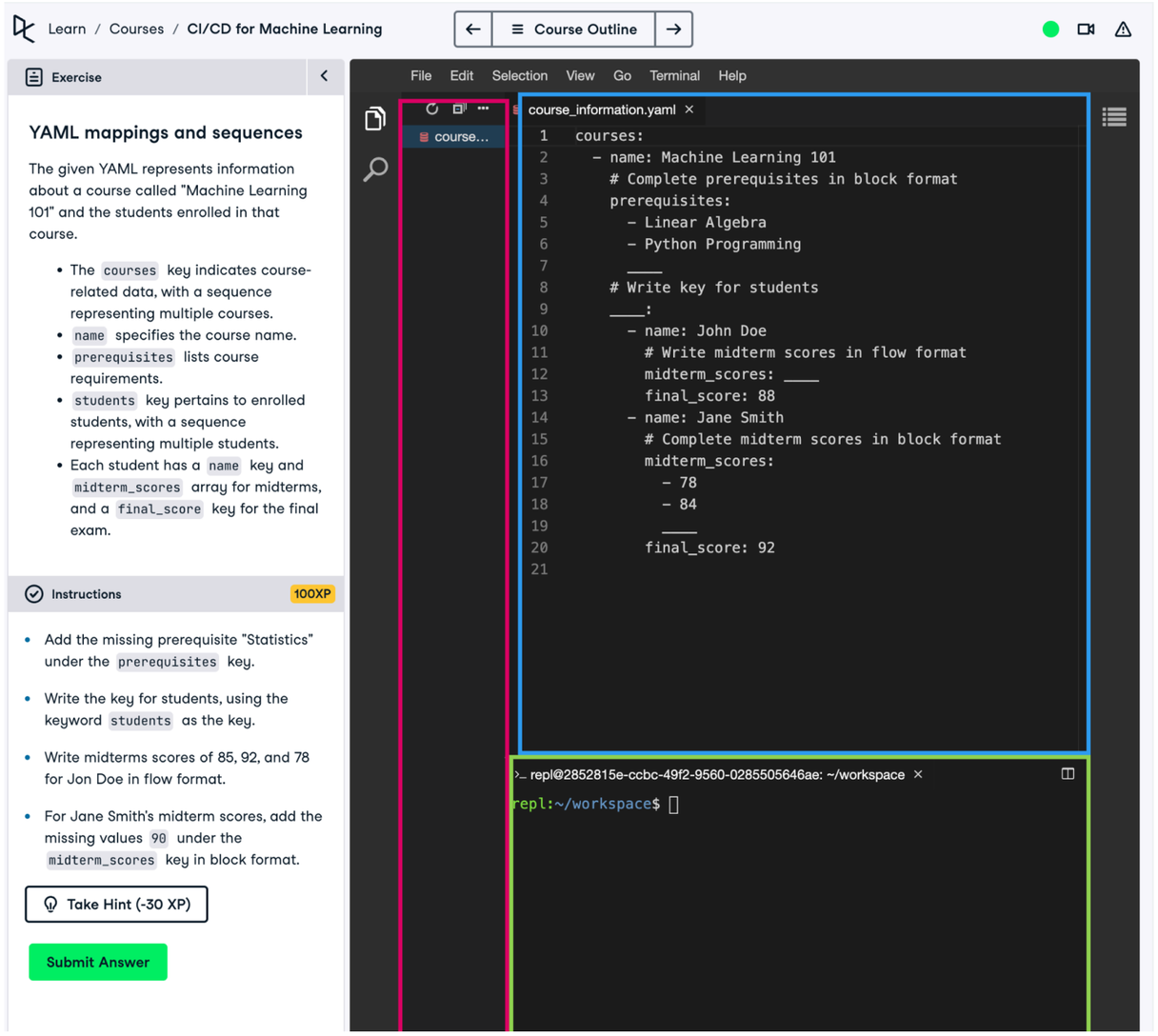
Metric	Dataset A	Dataset B
Precision	0.78	0.79
Recall	0.54	0.57
F1 Score	0.64	0.66
Accuracy	0.80	0.81

Hyperparameters influence

Dataset kept consistent, hyperparameters changed

Metric	`n_estimators=5`	`n_estimators=10`
Precision	0.78	0.85
Recall	0.54	0.52
F1 Score	0.64	0.65
Accuracy	0.80	0.81

Editor Exercises Layout



Let's practice!

INTRODUCTION TO DATA VERSIONING WITH DVC

Introduction to DVC

INTRODUCTION TO DATA VERSIONING WITH DVC

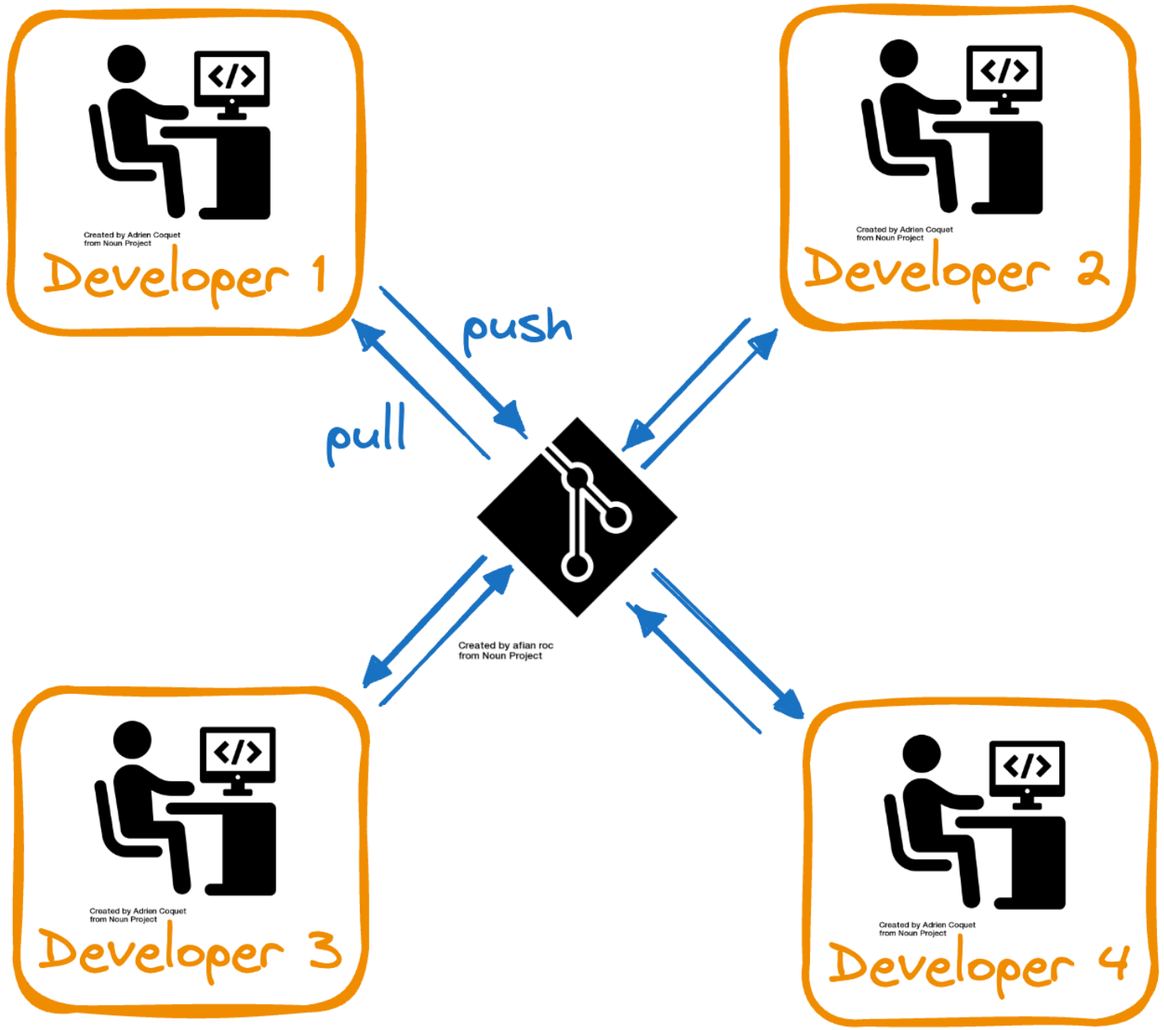


Ravi Bhadauria
Machine Learning Engineer

datacamp

Git as Version Control

- Code version control system
- Independent local development
  - Branch and merge
  - Version history management
- Enables collaboration



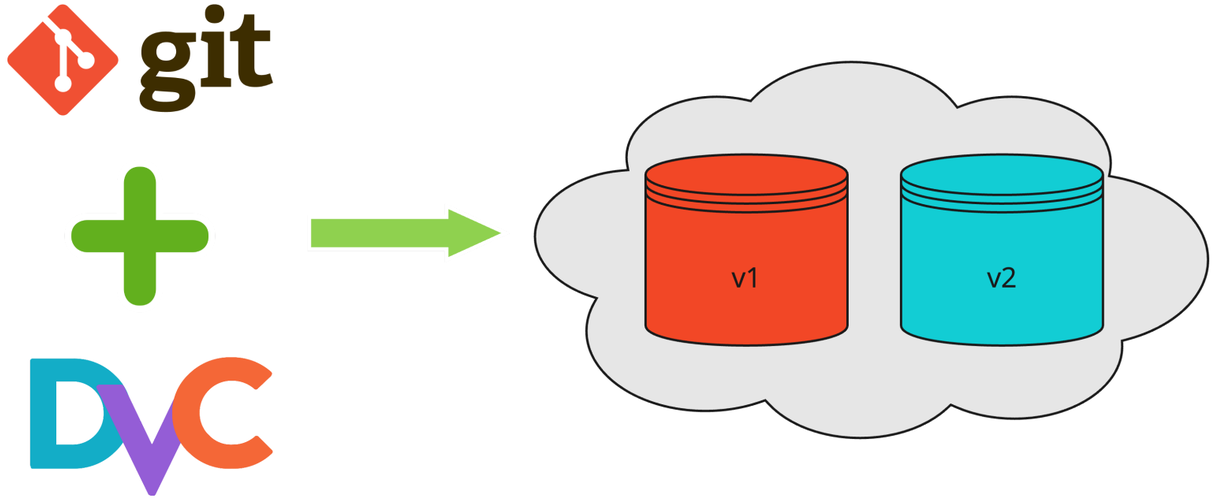
Git as Version Control

- CLI based interaction
- Run on *terminal*, aka *shell*
- Git tracks contents via a repository
  - Actual files/folders to be tracked
  - Git metadata (in `.git` folder)



Data Version Control (DVC)

- DVC: Data Version Control tool
  - Manages data and experiments
  - Similar to Git



- Git tracks metadata, DVC handles data versioning

Git vs DVC CLI

Git

- Initialize repository in working folder

```
$ git init
```

- Adding files to repository (staging changes)

```
$ git add code.py
```

- Commit changes (in version history)

```
$ git commit -m "adding first file"
```

DVC

- Initialize DVC repository in working folder

```
$ dvc init
```

- Adding data files to DVC

```
$ dvc add data/mydata.csv
```

- Updating all tracked data files

```
$ dvc commit
```

Git vs DVC CLI

Git

- Push code changes to remote server

```
$ git push
```

- Pulling changes from remote

```
$ git pull
```

- Cloning an existing repository from remote (Github)

```
$ git clone \
https://github.com/username/repository-name.git
```

DVC

- Push data changes to remote data server

```
$ dvc push
```

- Synchronizing your DVC project

```
$ dvc pull
```

- Download a file or directory tracked by DVC

```
$ dvc get \
https://github.com/username/repo-name model.pkl
```

Let's practice!

INTRODUCTION TO DATA VERSIONING WITH DVC

DVC features and use cases

INTRODUCTION TO DATA VERSIONING WITH DVC



Ravi Bhadauria
Machine Learning Engineer

datacamp

DVC features and use cases

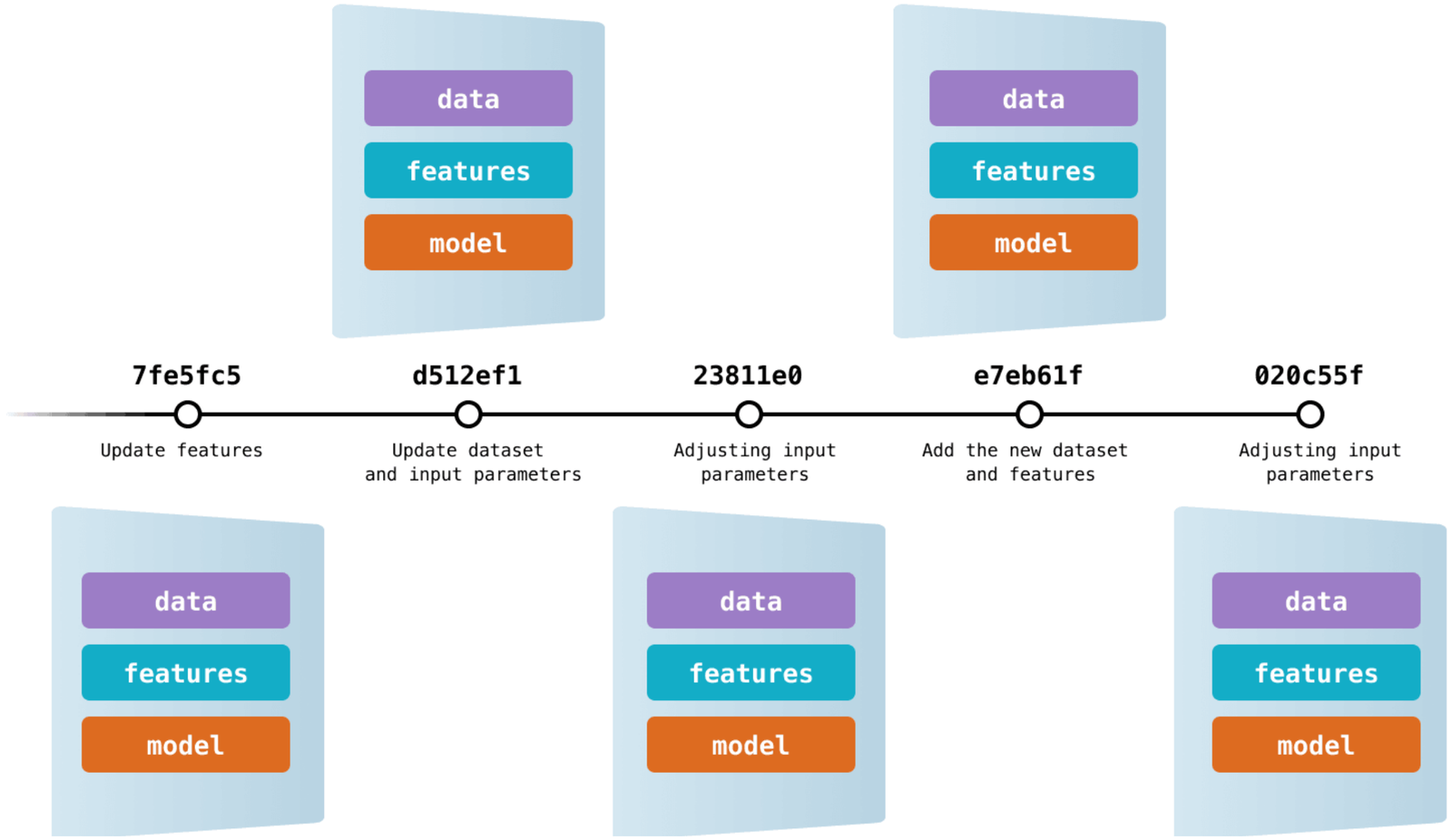
Covered topics

- Versioning data and models
- DVC Pipelines
- Metrics and plots tracking

Advanced topics (not covered)

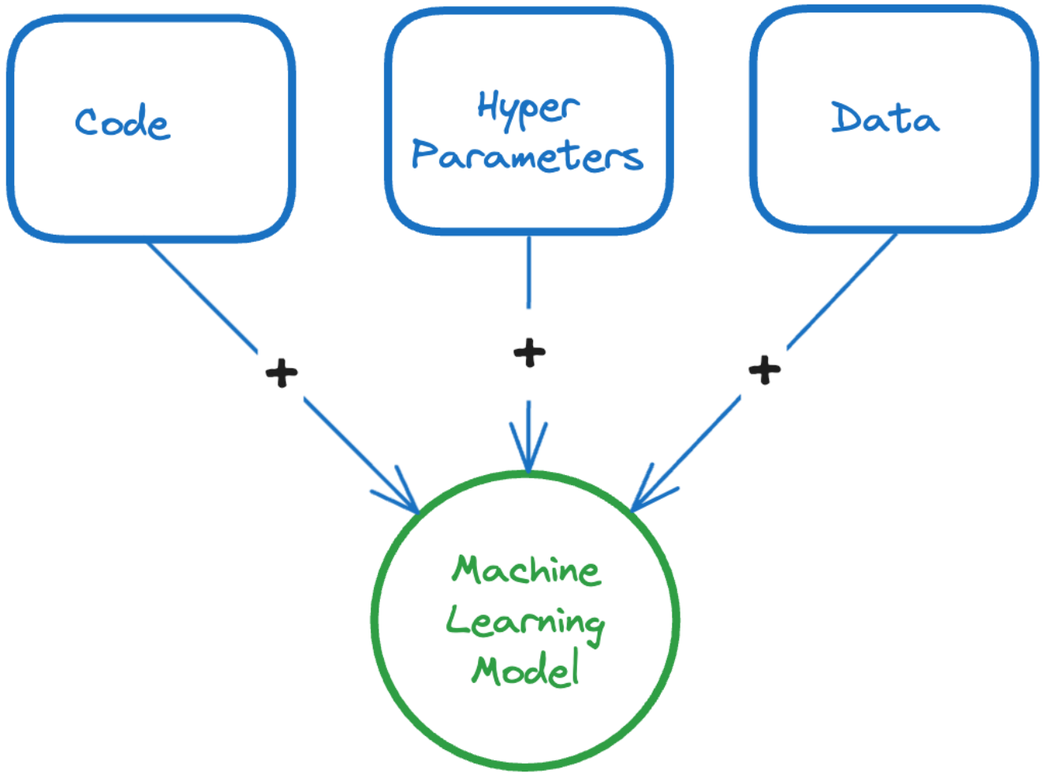
- Experiment tracking
- CI/CD for machine learning
- Data registry

Versioning data and models



[1] https://dvc.org/doc/use-cases/versioning-data-and-models

Pipelines



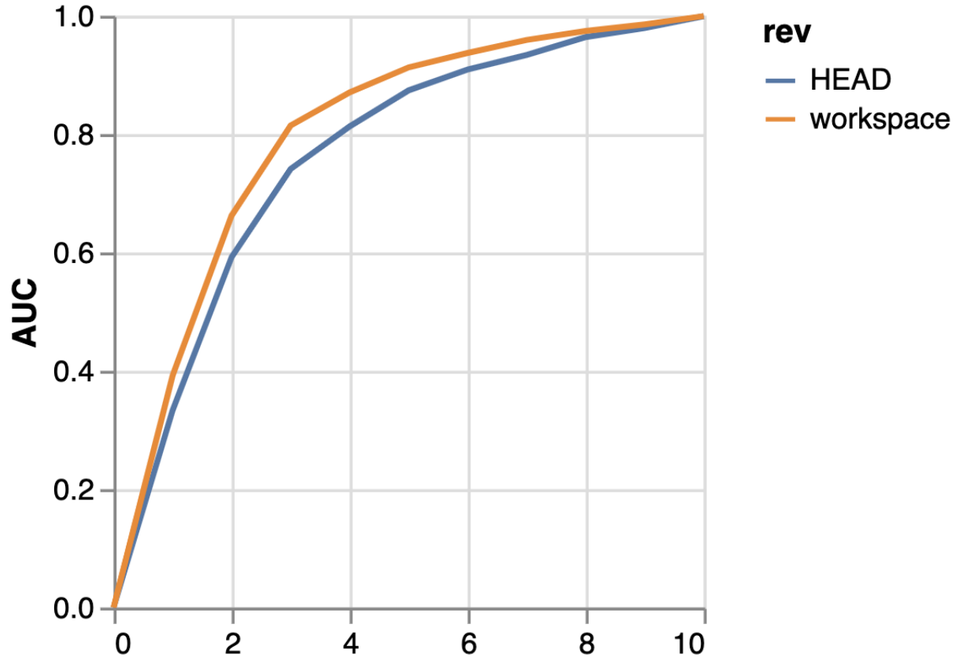
- Define pipeline in `dvc.yaml`

```
stages:
  train:
    cmd: python train.py
    deps:
      - code/train.py
      - data/input_data.csv
      - params/params.json
    outs:
      - model_output/model.pkl
```

- Run with `dvc repro`.

Tracking metrics and plots

```
$ dvc metrics diff
Path                    Metric    HEAD      workspace    Change
dvclive/metrics.json    AUC       0.78912   0.18114      -0.60798
dvclive/metrics.json    TP        215       768          553
```

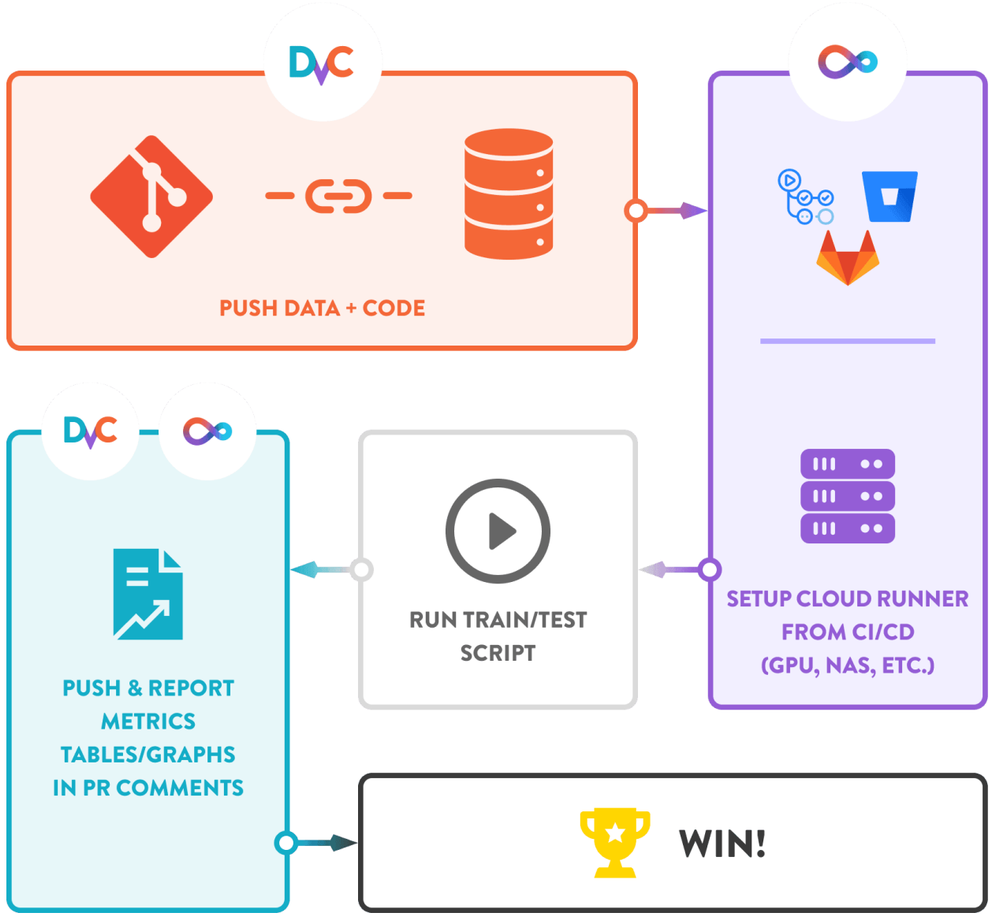


[1] https://dvc.org/doc/command-reference/plots/diff

Experiment tracking

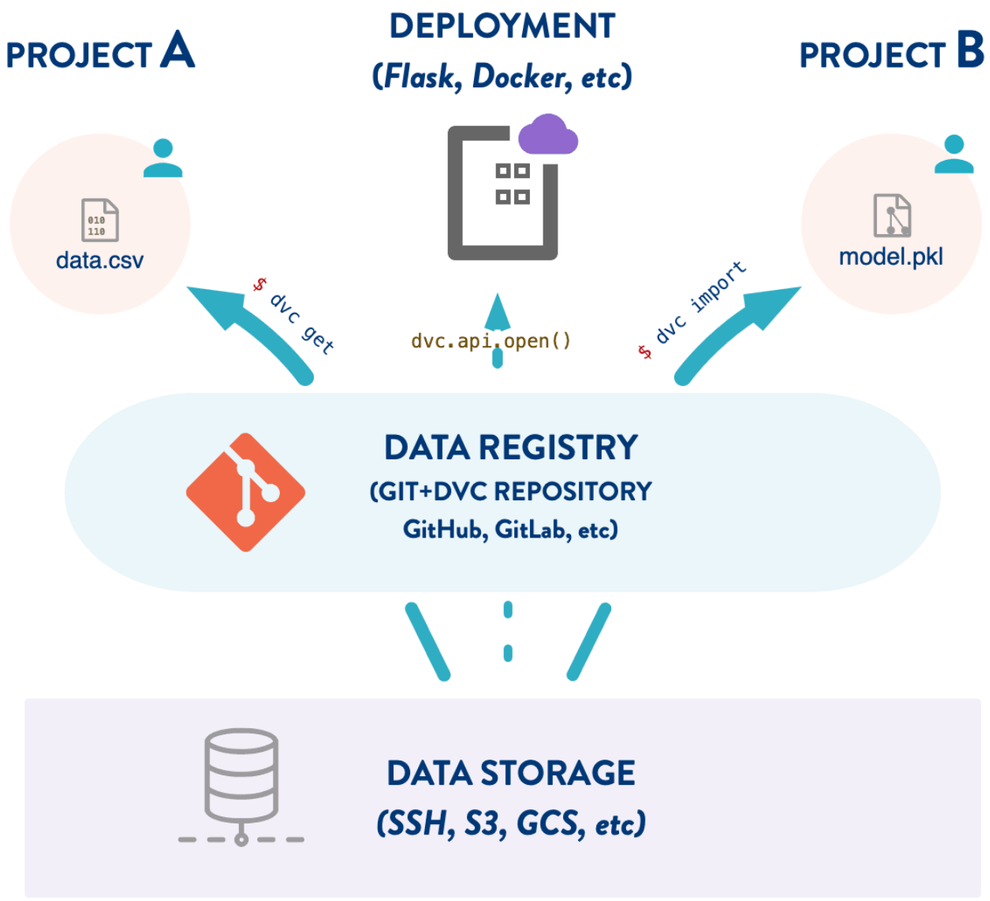
- Run experiment and log metrics
  - `dvc repro`
  - `dvc exp save`
- Alternatively, combine two steps `dvc exp run`
- Experiments are custom Git references
  - Prevent bloating up Git commits
  - Explicit saves can be made with `dvc exp save`
- Visualize using `dvc exp show`

CI/CD for Machine Learning



[1] Picture credits: https://dvc.org/doc/use-cases/ci-cd-for-machine-learning

Data registry



[1] Picture credits: https://dvc.org/doc/use-cases/data-registry

Let's practice!

INTRODUCTION TO DATA VERSIONING WITH DVC