How to setup your Data Science project in 2021

Vladislav Goncharenko,
Computer Vision Team Lead at Evocargo
Feb 2021

girafe ai

Foreword

- All conclusions are author's findings
  (personal opinion based on own experience, your may differ)
- Only bird's eye view - better dig yourself into links (plenty of them on slides)
- Additions, comments and discussions are welcome!

girafe ai

Outline

1. Version control system
    a. Data version control
2. Data annotation
3. Jupyter notebooks
4. Code style
5. Dependencies and packaging
6. Models training
7. *Testing (not included, external video)*
8. *Inference (not included, external video)*

Version Control System (VCS)

girafe
ai

01

Version Control System

- Source code - git
- Cloud remote - gitlab or github
- Best practice - merge requests (pull requests)

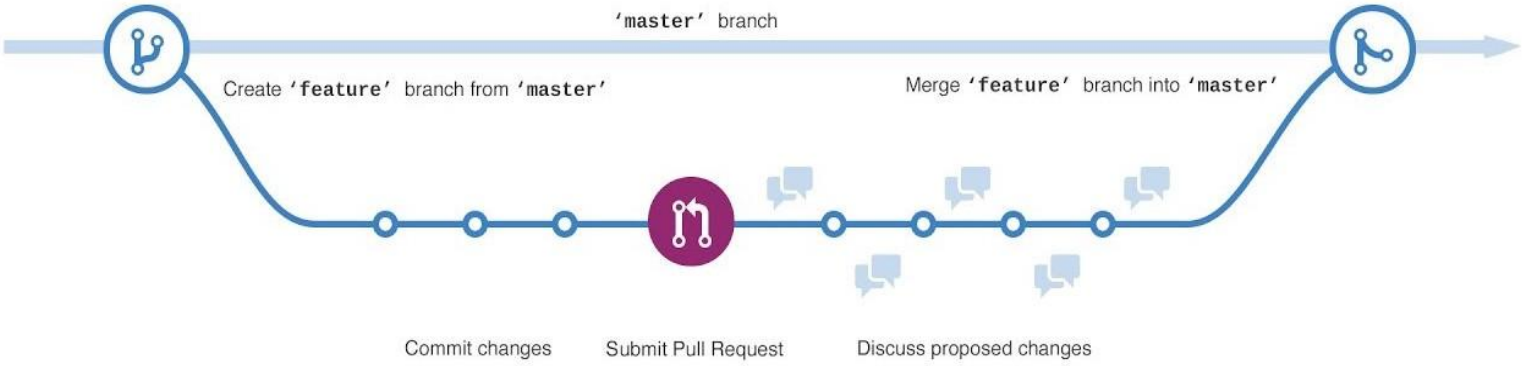
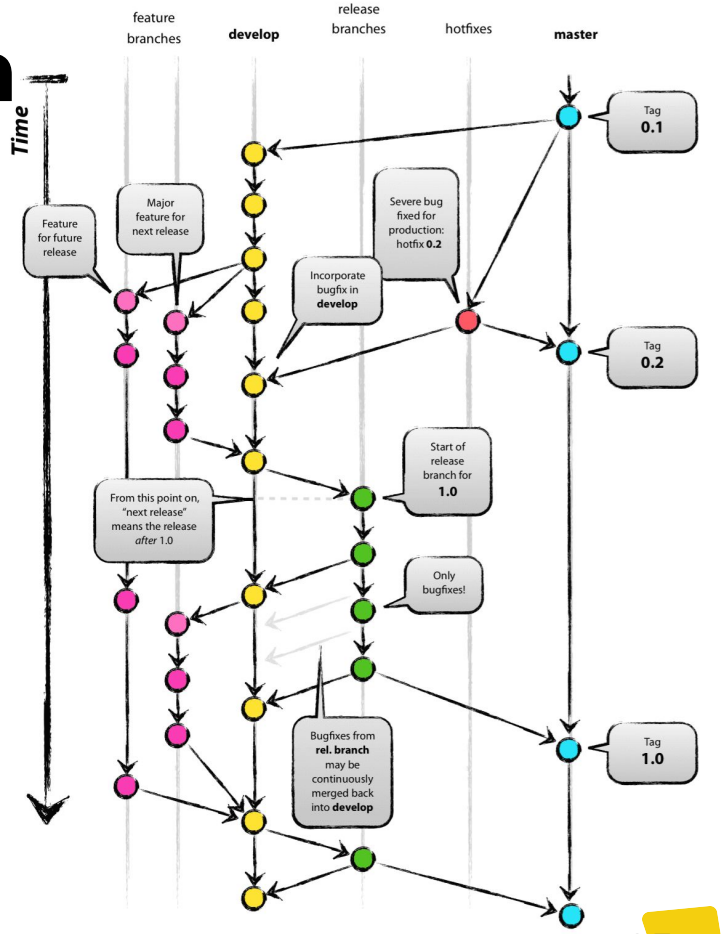


image source

Version Control System

- Source code - [git]
- Cloud remote - gitlab or github
- Best practice - [merge requests (pull requests)]
    - Advanced - [Git Flow (original article)]
    - [Flows comparison]



Version Control System

- Source code - git
- Cloud remote - gitlab or github
- Best practice - merge requests (pull requests)
    - Advanced - Git Flow (original article)
    - Flows comparison
- Further automation - CI/CD

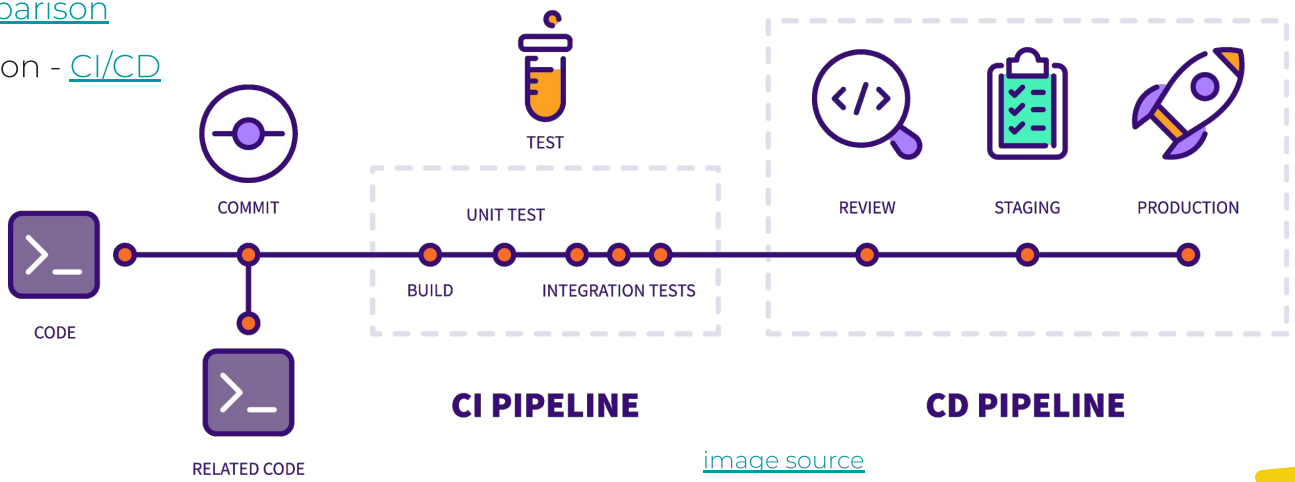
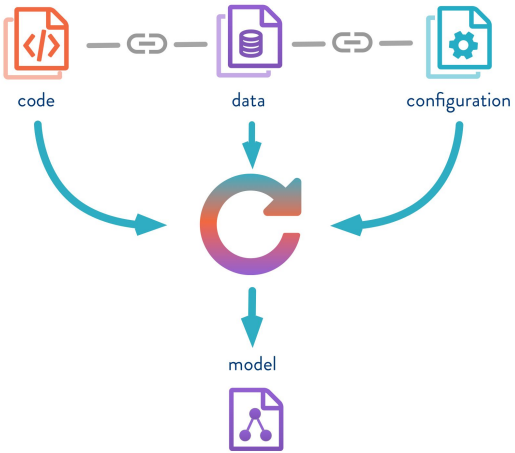


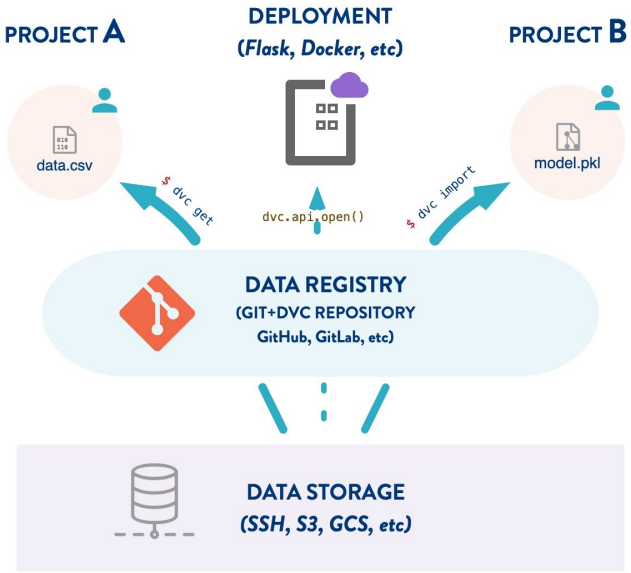
image source

Data Version Control (DVC)

- git for data is DVC (tutorials: one, two)
- use only Versioning and Access

  (Pipelines and Experiments are still raw)



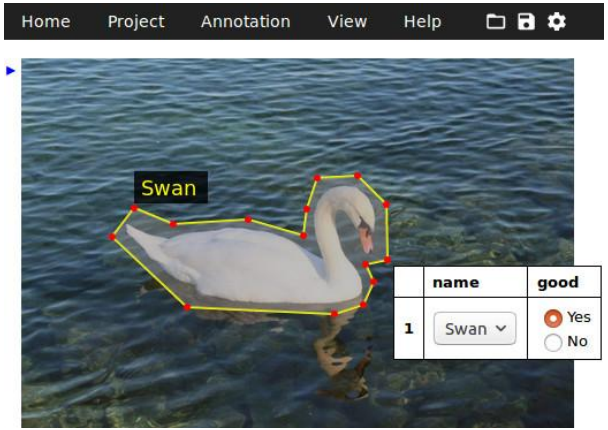


Data Annotation

girafe ai

02

Data Annotation

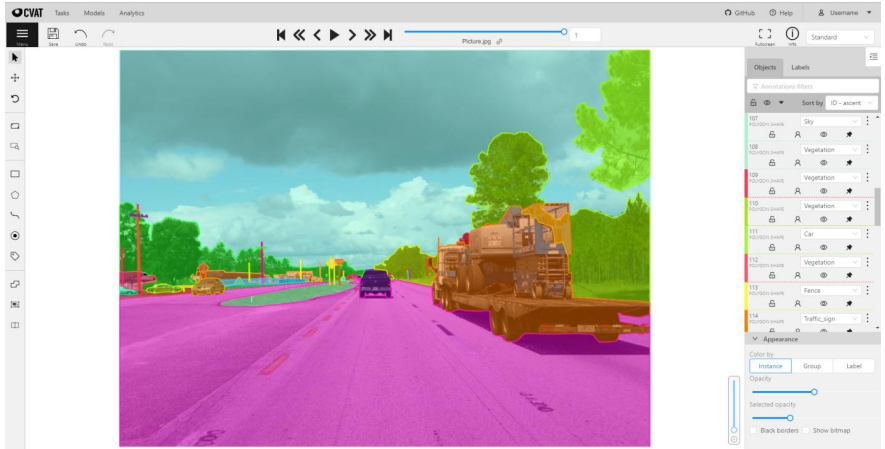


Solutions specific to Computer Vision

- simple cases - VIA (free software, standalone)
- scalable solution - CVAT (free software, server based)
- special cases - hasty.ai (proprietary, server based)

All of them are web based

Suggest your favorite tools in comments! (especially for other tasks)



girafe ai

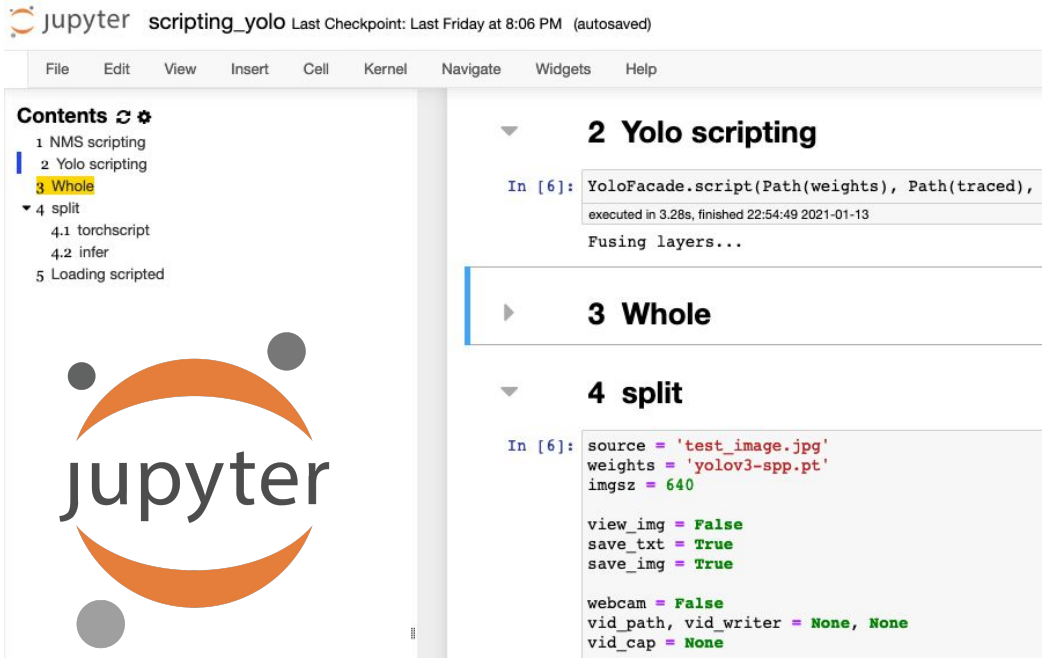
Jupyter Notebooks

girafe ai

03

Jupyter Notebooks

- One server for all kernels with [nb_conda_kernels]
- Advanced features functionality with [extensions] and [configurator]
- Notebooks are not added to git
- Used only for prototyping, transfer to .py files (repo) as soon as possible!!! (if you hit one cell at least twice)



girafe ai

Code Style

girafe
ai

04

Code Style

- pre-commit

  uses git hooks to run common utilities before git commands processing

```
$ pre-commit install
pre-commit installed at /home/asottile/workspace/pytest/.git/hooks/pre-commit
$ git commit -m "Add super awesome feature"
black.....................................................................Passed
blacken-docs.........................................(no files to check)Skipped
Trim Trailing Whitespace.................................................Passed
Fix End of Files.........................................................Passed
Check Yaml...........................................(no files to check)Skipped
Debug Statements (Python)................................................Passed
Flake8...................................................................Passed
Reorder python imports...................................................Passed
pyupgrade................................................................Passed
rst ``code`` is two backticks........................(no files to check)Skipped
rst..................................................(no files to check)Skipped
changelog filenames..................................(no files to check)Skipped
[master 146c6c2c] Add super awesome feature
 1 file changed, 1 insertion(+)
```

Code Style

- pre-commit
- black (PyCon talk)

  The uncompromising code formatter

  Other solutions:

  - yapf
  - autopep8

```
class Foo  (     object  ):
  def f    (self
  ):
    return       37*-2
  def g(self, x,y=42):
      return y
def f  (   a: List[ int ]) :
  return      37-a[42-u :  y**3]
```

```
class Foo(object):
    def f(self):
        return 37 * -2

    def g(self, x, y=42):
        return y


def f(a: List[int]):
    return 37 - a[42 - u : y ** 3]
```

Code Style

- pre-commit
- black
- isort

  Configurable

  Sorts import to 3 categories:

  - standard library
  - external (third party)
  - internal (this project and associatives)

isort your imports, so you don't have to.

```
import os
from math import ceil
import myproject.test

import .utils
from math import sqrt
import django.settings
```

```
from math import ceil, sqrt
import os

import django.settings

import myproject.test
```

girafe ai

Code Style

- pre-commit
- black
- isort
- flake8

  List of rules explained

Example of rules:

- F401: Module imported but unused
- E402: Module level import not at top of file
- F811: Redefinition of unused name from line n

More solutions from Python Code Quality Authority:

- bandit

Code Style

- pre-commit
- black
- isort
- flake8
- mypy

  Best for code editors,

  not pre-commit

: my[py]

Seamless dynamic and static typing

From Python...

```
def fib(n):
    a, b = 0, 1
    while a < n:
        yield a
        a, b = b, a+b
```

...to statically typed Python

```
def fib(n: int) -> Iterator[int]:
    a, b = 0, 1
    while a < n:
        yield a
        a, b = b, a+b
```

Code Style

- pre-commit
- black
- isort
- flake8
- mypy
- nbQA

  То же самое для Jupyter Notebooks!

  Поддерживает pre-commit

nbQA

Quality Assurance For Jupyter Notebooks

Code Style

- pre-commit
- black
- isort
- flake8
- mypy
- nbQA
- Documentation Driven Development (docstrings)
- Sphinx
- annotations, typing module
- pathlib

Dependencies and Packaging

girafe ai

05

Dependencies and Packaging

First - follow python package structure

- common extensible format
- able to be packaged

Second - dependencies management and packaging

- requirements.txt
- pip-tools
- pipx
- Pipfile and pipenv
- **poetry**

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

Dependencies and Packaging

One tool for all - poetry

- dares to install ONE tool per OS
- better conflicts resolving

Creates two files:

- `pyproject.toml` - initial dependencies, config for all tools
- `poetry.lock` - all concrete dependencies (platform agnostic)

If using conda environments:

```
poetry config virtualenvs.create false
```

For installing dependencies:

```
poetry install --no-root --remove-untracked
```

Also it is able to build and publish to PyPI or your own package registry

```
poetry build && poetry publish
```

Models Training

girafe ai

06

Models Training Framework

- Deep Learning
  - Pytorch
  - Tensorflow
- Gradient boosting
  - LightGBM
  - XGboost
  - CatBoost
- Others
  - Sklearn
  - Vowpal Wabbit
  - MLlib
  - .....

Experiments Runner

- Pytorch
  - Ignite
  - **Catalyst**
  - Lightning
- DVC Experiments
- MLFlow Experiments
- pachyderm
- .....

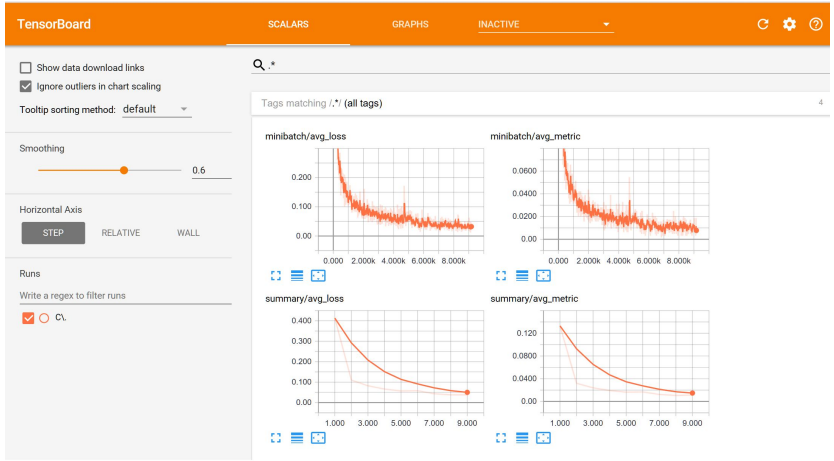
Catalyst

PyTorch Ignite

PyTorch Lightning

Experiments Tracker

- Tensorboard
- **MLFlow**
- sacred
- Kubeflow
- neptune.ai
- weights and biases
- .....



Summary

1. Version control system
    a. Data version control
2. Data annotation
3. Jupyter notebooks
4. Code style
5. Dependencies and packaging
6. Models training
7. *Testing (not included, external video)*
8. *Inference (not included, external video)*

Thanks for attention!

Questions? Additions? Welcome!

girafe ai