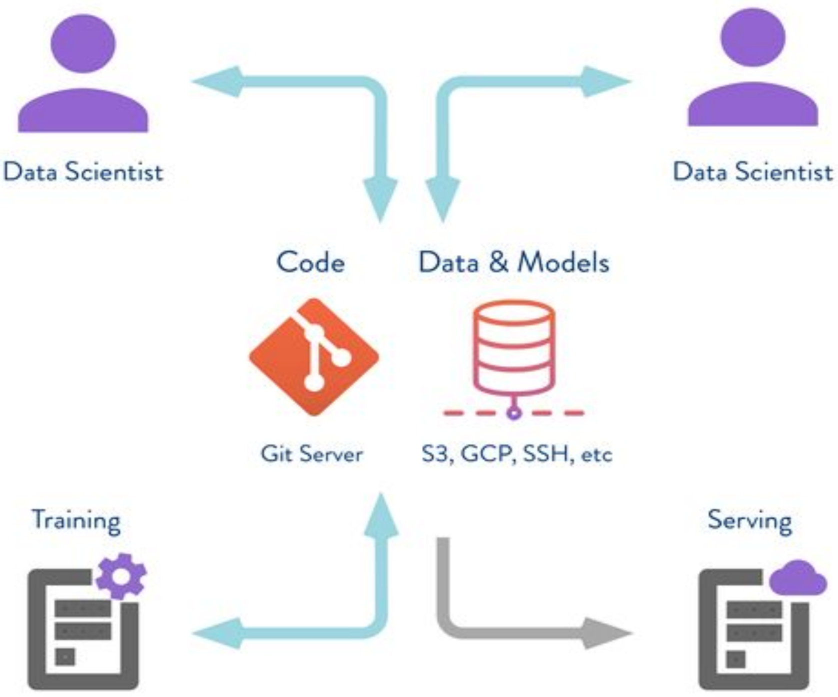
Data version control

Stanisław Giziński

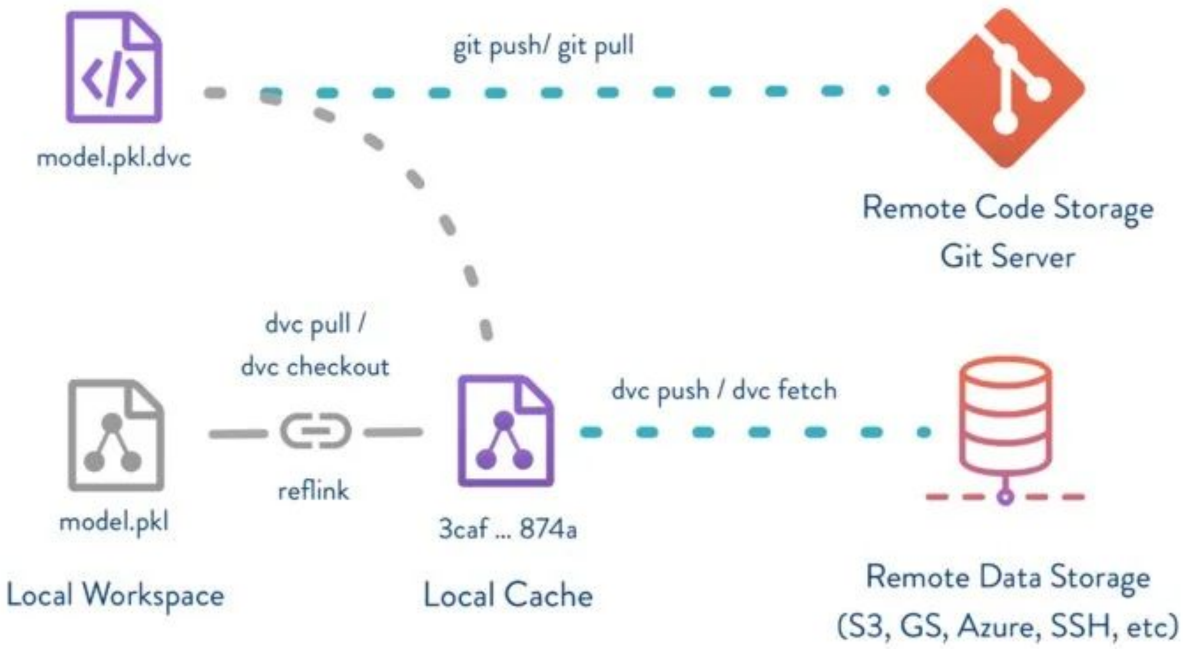
Problems

- Reproducibility and tracking dependencies between process steps
- Versioning data and artifacts
- Sharing the data and artifacts

Utilizing git



Utilizing git



Usage example

Project structure

- data/
    - *raw.csv*
- scripts/
    - prep_data.py
    - split_data.py
    - prep_features.py
    - train.py
    - eval.py

Files written in *italics* are not tracked by git (added to .gitignore)

```
git init

dvc init
```

Dependencies between process steps

Project structure

- data/
  - ***raw.csv***
- scripts/
  - prep_data.py
  - split_data.py
  - prep_features.py
  - train.py
  - eval.py

```
dvc add data/raw.csv
```

Versioning the data

Project structure

- data/
  - *raw.csv*
  - **raw.csv.dvc**
- scripts/
  - prep_data.py
  - split_data.py
  - prep_features.py
  - train.py
  - eval.py

```
dvc add data/raw.csv
```

Creates

```
outs:
- md5: 576bf6473a980759a0d1ea7d44ffdd46
  size: 16942
  path: raw.csv
```

Dependencies between process steps

Project structure

- data/
  - *raw.csv*
  - raw.csv.dvc
- scripts/
  - **prep_data.py**
  - split_data.py
  - prep_features.py
  - train.py
  - eval.py

```
dvc run -n prepare \

-d scripts/prep_data.py -d data/raw.csv \

-o data/prepared.csv \

python scripts/prep_data.py data/raw.csv
data/prepared.csv
```

Dependencies between process steps

Project structure

- data/
  - *raw.csv*
  - raw.csv.dvc
  - ***prepared.csv***
- scripts/
  - prep_data.py
  - split_data.py
  - prep_features.py
  - train.py
  - eval.py
- **dvc.yaml**
- **dvc.lock**

```
prepare:
    cmd: python scripts/prep_data.py
    deps:
      - scripts/prep_data.py
      - data/raw.csv
    outs:
      - data/prepared.csv
```

Dependencies between process steps

Project structure

- data/
  - *raw.csv*
  - raw.csv.dvc
  - ***prepared.csv***
- scripts/
  - prep_data.py
  - **split_data.py**
  - prep_features.py
  - train.py
  - eval.py
- dvc.yaml
- dvc.lock

```
dvc run -n split_data \

-d scripts/split_data.py -d data/prepared.csv \

-o data/train.csv -o data/test.csv

python scripts/split_data.py data/prepared.csv
data/train.csv data/test.csv
```

Dependencies between process steps

Project structure

- data/
  - *raw.csv*
  - raw.csv.dvc
  - *prepared.csv*
  - *train.csv*
  - *test.csv*
- scripts/
  - prep_data.py
  - split_data.py
  - **prep_features.py**
  - train.py
  - eval.py
- dvc.yaml
- dvc.lock

```
dvc run -n prep_features_train \

-d scripts/prep_features.py -d data/train.csv \

-o data/train_features.npy \

python scripts/prep_features.py data/train.csv
data/train_features.npy
```

```
dvc run -n prep_features_test \

-d scripts/prep_features.py -d data/test.csv \

-o data/test_features.npy \

python scripts/prep_features.py data/test.csv
data/test_features.npy
```

Dependencies between process steps

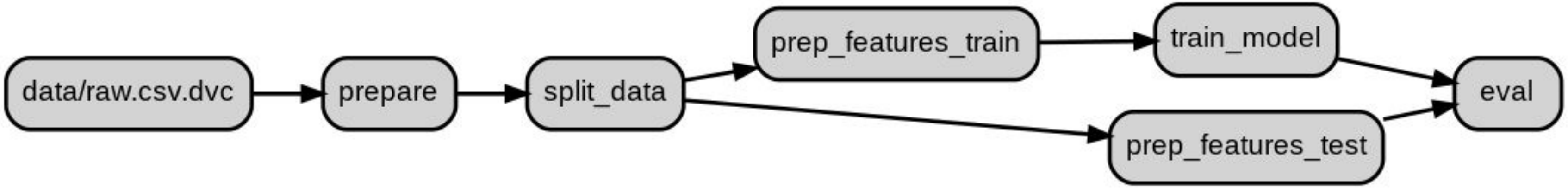
Project structure

- data/
    - *raw.csv*
    - raw.csv.dvc
    - *prepared.csv*
    - *train.csv*
    - *test.csv*
    - ***train_features.npy***
    - ***test_features.npy***

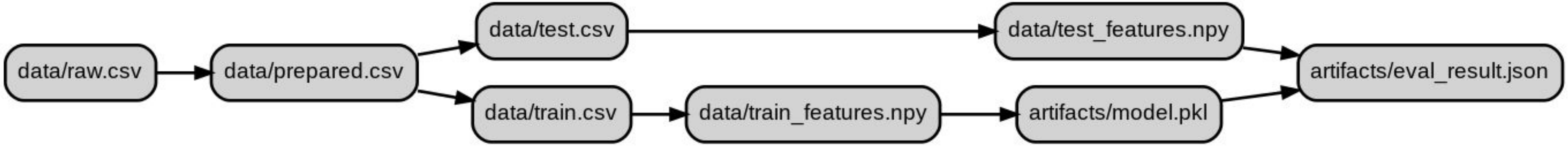
```
stages:
 prepare:
   cmd: python scripts/prep_data.py data/raw.csv
data/prepared.csv
   deps:
     - data/raw.csv
     - scripts/prep_data.py
   outs:
     - data/prepared.csv
 split_data:
   cmd: python scripts/split_data.py data/prepared.csv
data/train.csv data/test.csv
   deps:
     - data/prepared.csv
     - scripts/split_data.py
   outs:
     - data/test.csv
     - data/train.csv
 prep_features_train:
   cmd: python scripts/prep_features.py data/train.csv
data/train_features.npy
   deps:
     - data/train.csv
     - scripts/prep_features.py
   outs:
     - data/train_features.npy
 prep_features_test:
   cmd: python scripts/prep_features.py data/test.csv
data/test_features.npy
   deps:
     - data/test.csv
     - scripts/prep_features.py
   outs:
     - data/test_features.npy
 train_model:
   cmd: python scripts/train.py data/train_features.npy
artifacts/model.pkl
   deps:
     - scripts/train.py
     - data/train_features.npy
   outs:
     - artifacts/model.pkl
 eval:
   cmd: python scripts/eval.py artifacts/model.pkl
data/test_features.npy artifacts/eval_result.json
   deps:
     - scripts/eval.py
     - artifacts/model.pkl
     - data/test_features.npy
   outs:
     - artifacts/eval_result.json
```

Dependencies between process steps

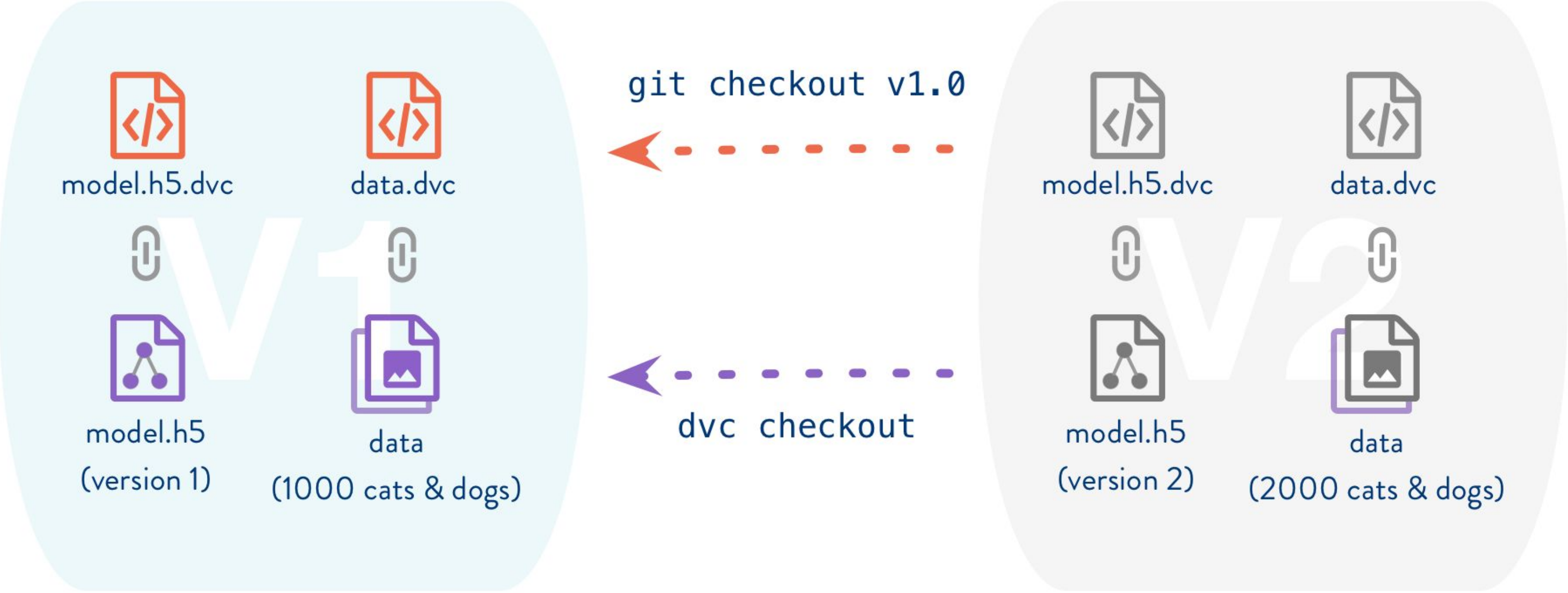
```
dvc dag
```



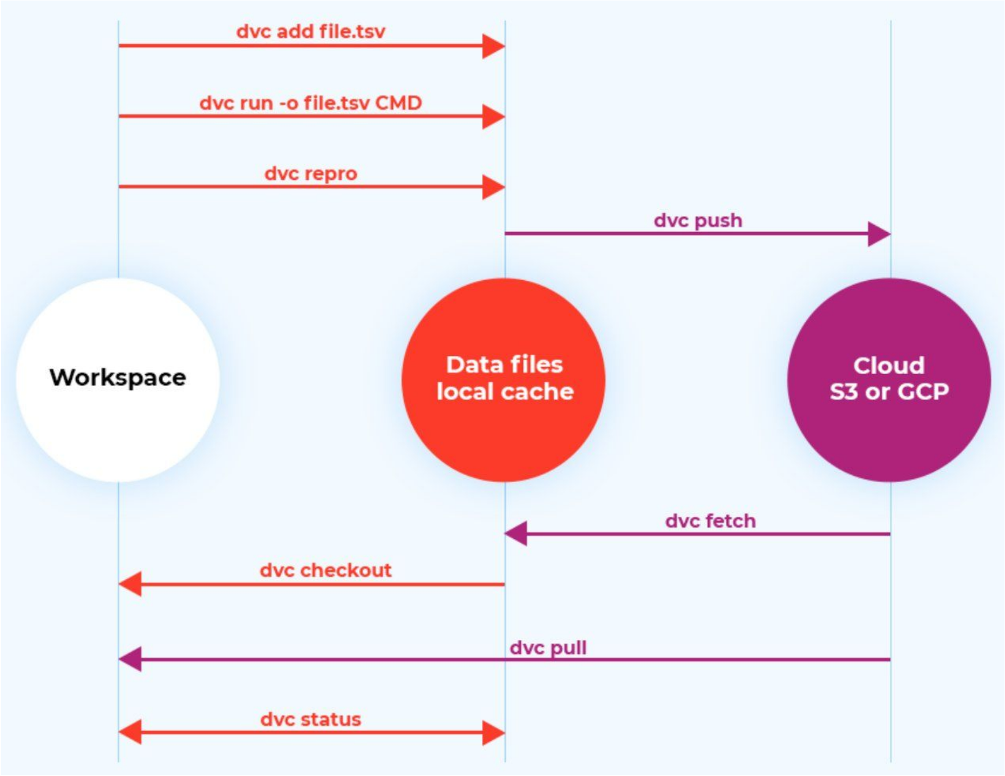
```
dvc dag -o
```



Changing versions



How does it work



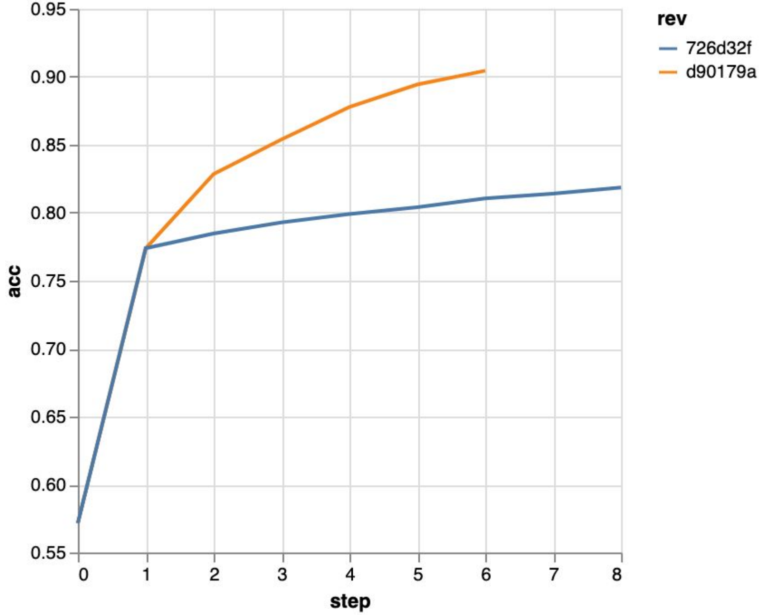
Experiments comparison

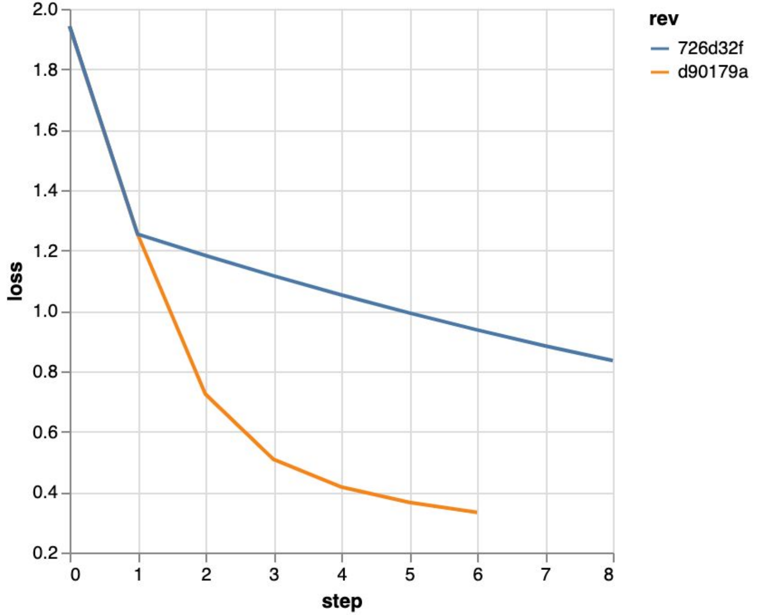
```
dvc metrics diff d90179a 726d32f
```

```
Path          Metric    Old      New      Change
metrics.json  acc       0.9044   0.8185   -0.0859
metrics.json  loss      0.33246  0.83515  0.50269
metrics.json  step      6        8        2
```

Experiments comparison

```
dvc plots diff d90179a 726d32f
```

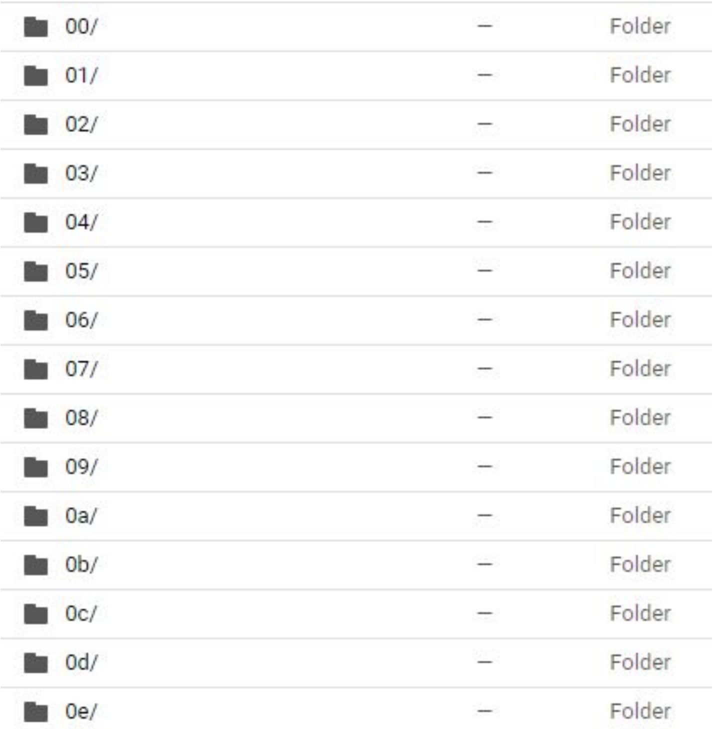




Drawbacks

Data stored on remote is stored in internal dvc format, not readable directly.

User has to manually specify outputs for every step.



00/	—	Folder
01/	—	Folder
02/	—	Folder
03/	—	Folder
04/	—	Folder
05/	—	Folder
06/	—	Folder
07/	—	Folder
08/	—	Folder
09/	—	Folder
0a/	—	Folder
0b/	—	Folder
0c/	—	Folder
0d/	—	Folder
0e/	—	Folder

Sources

Tutorial

https://dvc.org/doc/use-cases/versioning-data-and-model-files/tutorial

Documentation

https://dvc.org/doc