Data Version Control (DVC):
Tutorial 1: Get Started

Mikhail Rozhkov

2019

What is DVC tool?

- ML project version control
- ML experiment management
- Deployment & Collaboration

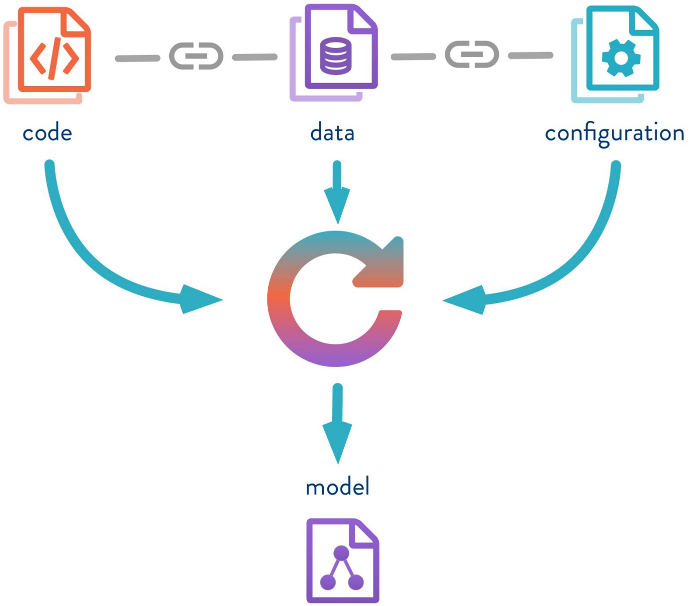


Image source: https://dvc.org/doc/tutorial

Use Case:

Iris Flowers Classification

- Task: classify Iris flowers
- Dataset: Iris dataset
- Metrics: F1

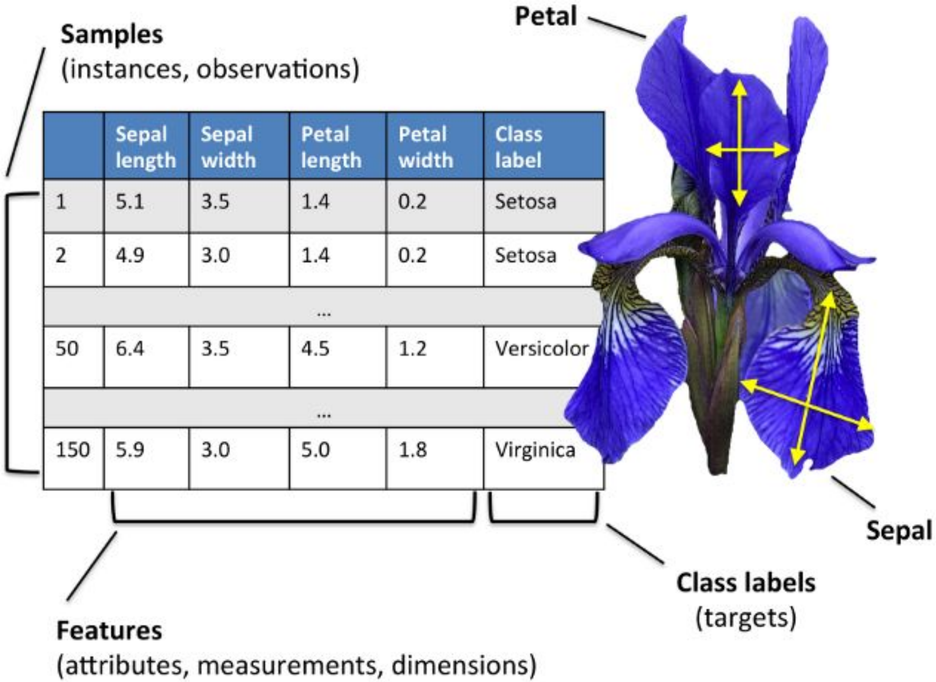
References:

- https://en.wikipedia.org/wiki/Iris_flower_data_set
- https://scikit-learn.org/stable/tutorial/statistical_inference/supervised_learning.html

Image source:
https://medium.com/@jebaseelanravi96/machine-learning-iris-classification-33aa18a4a983

Use Case:

Iris Flowers Classification



Samples
(instances, observations)
Petal
Sepal
Class labels
(targets)
Features
(attributes, measurements, dimensions)

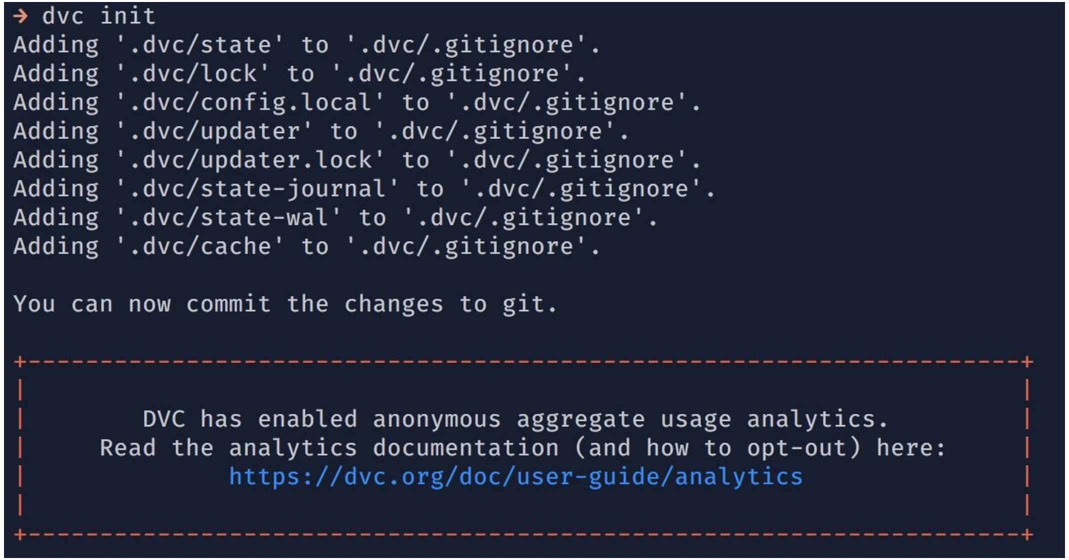
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Image source:
https://medium.com/@jebaseelanravi96/machine-learning-iris-classification-33aa18a4a983

Step 1:

Preparation

- clone repository
- create virtual environment
- install required python packages
- initialize DVC



```
→ dvc init
Adding '.dvc/state' to '.dvc/.gitignore'.
Adding '.dvc/lock' to '.dvc/.gitignore'.
Adding '.dvc/config.local' to '.dvc/.gitignore'.
Adding '.dvc/updater' to '.dvc/.gitignore'.
Adding '.dvc/updater.lock' to '.dvc/.gitignore'.
Adding '.dvc/state-journal' to '.dvc/.gitignore'.
Adding '.dvc/state-wal' to '.dvc/.gitignore'.
Adding '.dvc/cache' to '.dvc/.gitignore'.

You can now commit the changes to git.

+---------------------------------------------------------------------+
|                                                                     |
|        DVC has enabled anonymous aggregate usage analytics.        |
|     Read the analytics documentation (and how to opt-out) here:    |
|              https://dvc.org/doc/user-guide/analytics              |
|                                                                     |
+---------------------------------------------------------------------+
```

Initialize DVC

# run command

```
$ dvc init
```

# output

```
Adding '.dvc/state' to '.dvc/.gitignore'.
Adding '.dvc/lock' to '.dvc/.gitignore'.
Adding '.dvc/config.local' to '.dvc/.gitignore'.
Adding '.dvc/updater' to '.dvc/.gitignore'.
Adding '.dvc/updater.lock' to '.dvc/.gitignore'.
Adding '.dvc/state-journal' to '.dvc/.gitignore'.
Adding '.dvc/state-wal' to '.dvc/.gitignore'.
Adding '.dvc/cache' to '.dvc/.gitignore'.

You can now commit the changes to git.
```

Commit changes

# run command

```
$ git add .
$ git commit -m "Initialize DVC"
```

# output

```
Initialize DVC
 2 files changed, 8 insertions(+)
 create mode 100644 .dvc/.gitignore
 create mode 100644 .dvc/config
```

DVC Files and Directories

# run command

```
$ ls -a .dvc
```

# output

```
./
../
.gitignore
cache/
config
```

# run command

```
$ cat .dvc/.gitignore
```

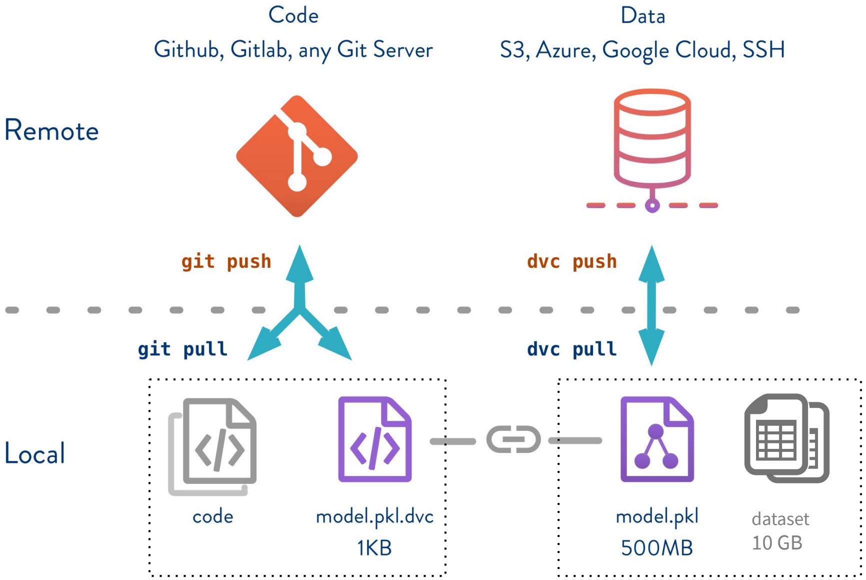
# output

```
/state
/lock
/config.local
/updater
/updater.lock
/state-journal
/state-wal
/cache
```

Step 2:

Data and Model Files Versioning

- add file to version control
- pull/push/checkout



Original image source: https://dvc.org/doc/use-cases/data-and-model-files-versioning

Add file under DVC control

# run command

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

# output

```
Adding 'data/iris.csv' to 'data/.gitignore'.
Saving 'data/iris.csv' to '.dvc/cache/57/fce90c81521889c736445f058c4838'.
Saving information to 'data/iris.csv.dvc'.
```

Add .dvc file to git

# run command

```
$ git status -s data/
```

# output

```
?? data/.gitignore
?? data/iris.csv.dvc
```

# run command

```
$ git add .
$ git commit -m "Add a source dataset"
```

# output

```
Add a source dataset
 2 files changed, 9 insertions(+)
 create mode 100644 data/.gitignore
 create mode 100644 data/iris.csv.dvc
```

What is DVC-file?

# run command

```
$ cat data/iris.csv.dvc
```

# output

```
md5: 1cff89878034249db68ba6046d5b49a9
wdir: ..
outs:
- md5: 57fce90c81521889c736445f058c4838
  path: data/iris.csv
  cache: true
  metric: false
  persist: false
```

# run command

```
$ du -sh .dvc/cache/*/*
```

# output

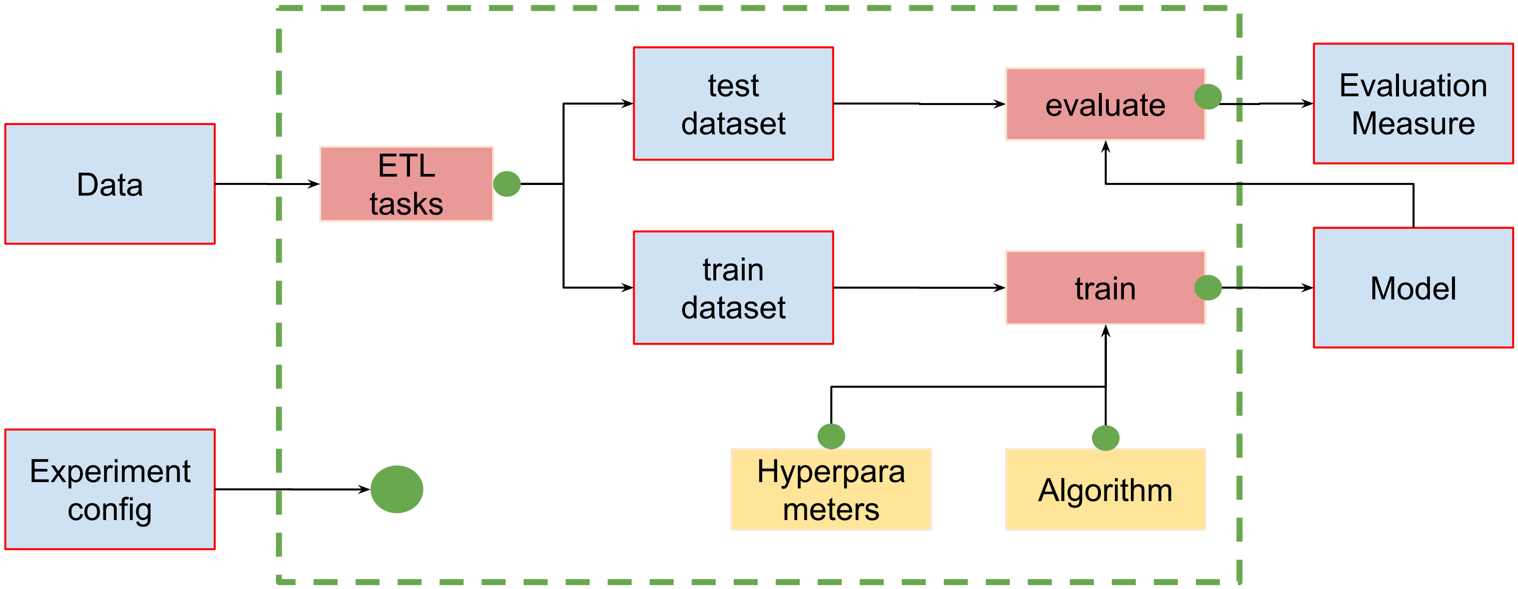
```
4.0K
.dvc/cache/57/fce90c81521889c736445f058c4838
```

Step 3:

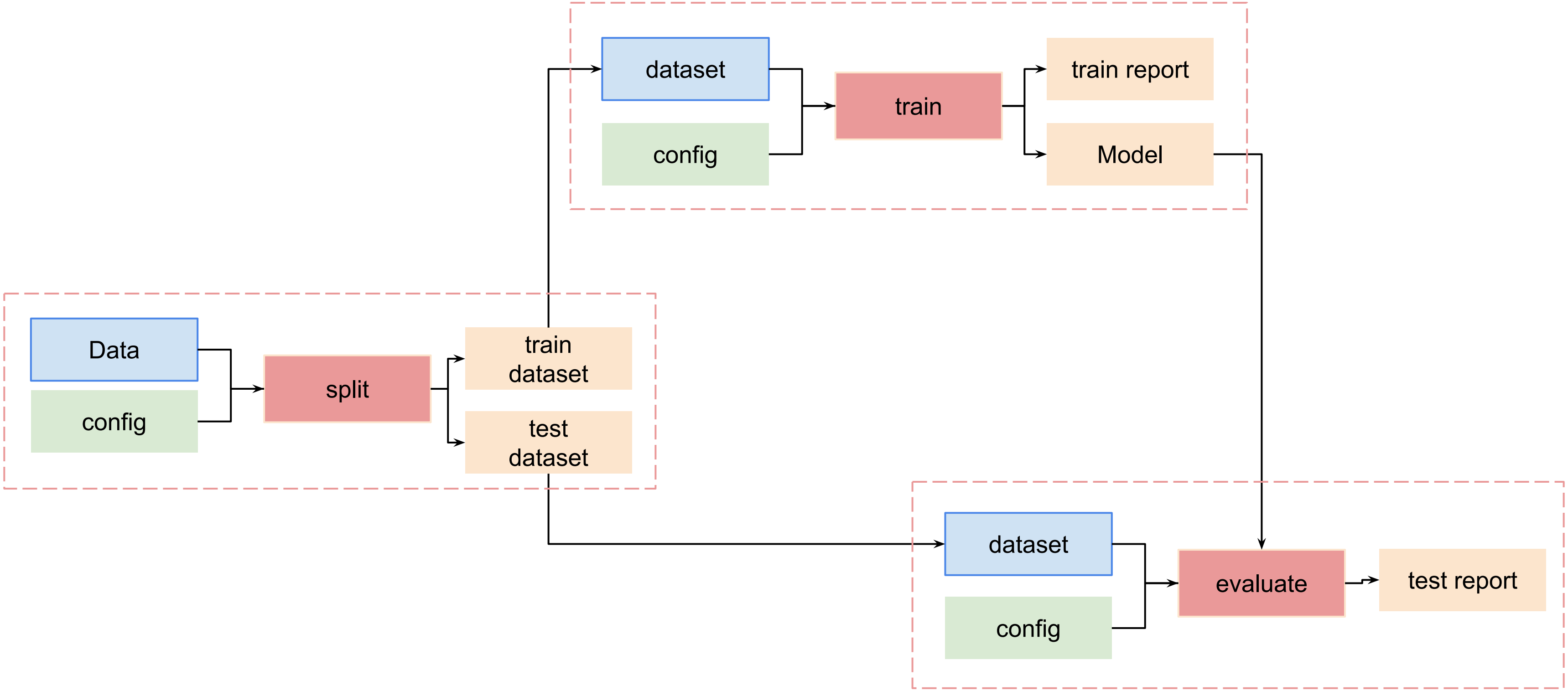
ML pipelines

- DVC pipeline concept
- **dvc run**
- params
- output .DVC file structure

Start with artifacts versioning!



ML pipelines



Add pipeline stages with dvc run

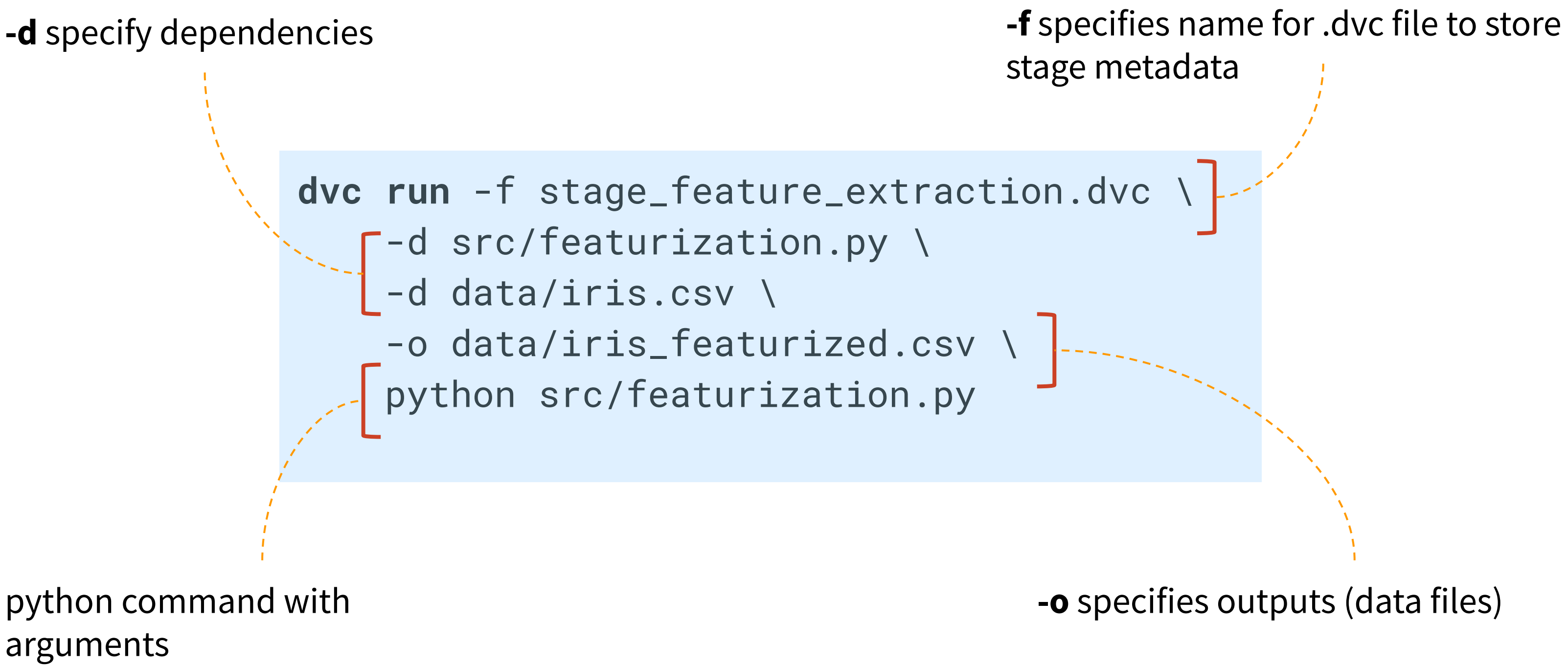
# run command

```
$ dvc run -f stage_feature_extraction.dvc \
   -d src/featurization.py \
   -d data/iris.csv \
   -o data/iris_featurized.csv \
   python src/featurization.py
```

# output

```
Running command:
     python src/featurization.py
Adding 'data/iris_featurized.csv' to 'data/.gitignore'.
Saving 'data/iris_featurized.csv' to '.dvc/cache/04/ed69383af337e9dabf934cbc8abc11'.
Saving information to 'stage_feature_extraction.dvc'.
To track the changes with git run:
     git add data/.gitignore stage_feature_extraction.dvc
```

Add pipeline stages with dvc run



-d specify dependencies

-f specifies name for .dvc file to store stage metadata

```
dvc run -f stage_feature_extraction.dvc \
    -d src/featurization.py \
    -d data/iris.csv \
    -o data/iris_featurized.csv \
    python src/featurization.py
```

python command with arguments

-o specifies outputs (data files)

stage_feature_extraction.dvc

```
md5: eec5e74d81a441ff02716cadd3779961
cmd: python src/featurization.py
wdir: .
deps:
- md5: 5bce3d2f01813491283efeb24789f97a
  path: src/featurization.py
- md5: 57fce90c81521889c736445f058c4838
  path: data/iris.csv
outs:
- md5: cd9e208c0232da2fb80b4c927da35dbb
  path: data/iris_featurized.csv
  cache: true
  metric: false
  persist: false
```

stage_split_dataset.dvc

```
md5: 2c0cd9e4926980b60a70eb58bc123727
cmd: python src/split_dataset.py 0.4
wdir: .
deps:
- md5: e111aa0fa66588bf06c5f716d11bcff5
  path: src/split_dataset.py
- md5: cd9e208c0232da2fb80b4c927da35dbb
  path: data/iris_featurized.csv
outs:
- md5: 8743ef62798f623fbaae4401f4aab654
  path: data/train.csv
  cache: true
  ...
- md5: 3d40f0c85187dda2cd9bf58b3e916630
  path: data/test.csv
  cache: true
  ...
```

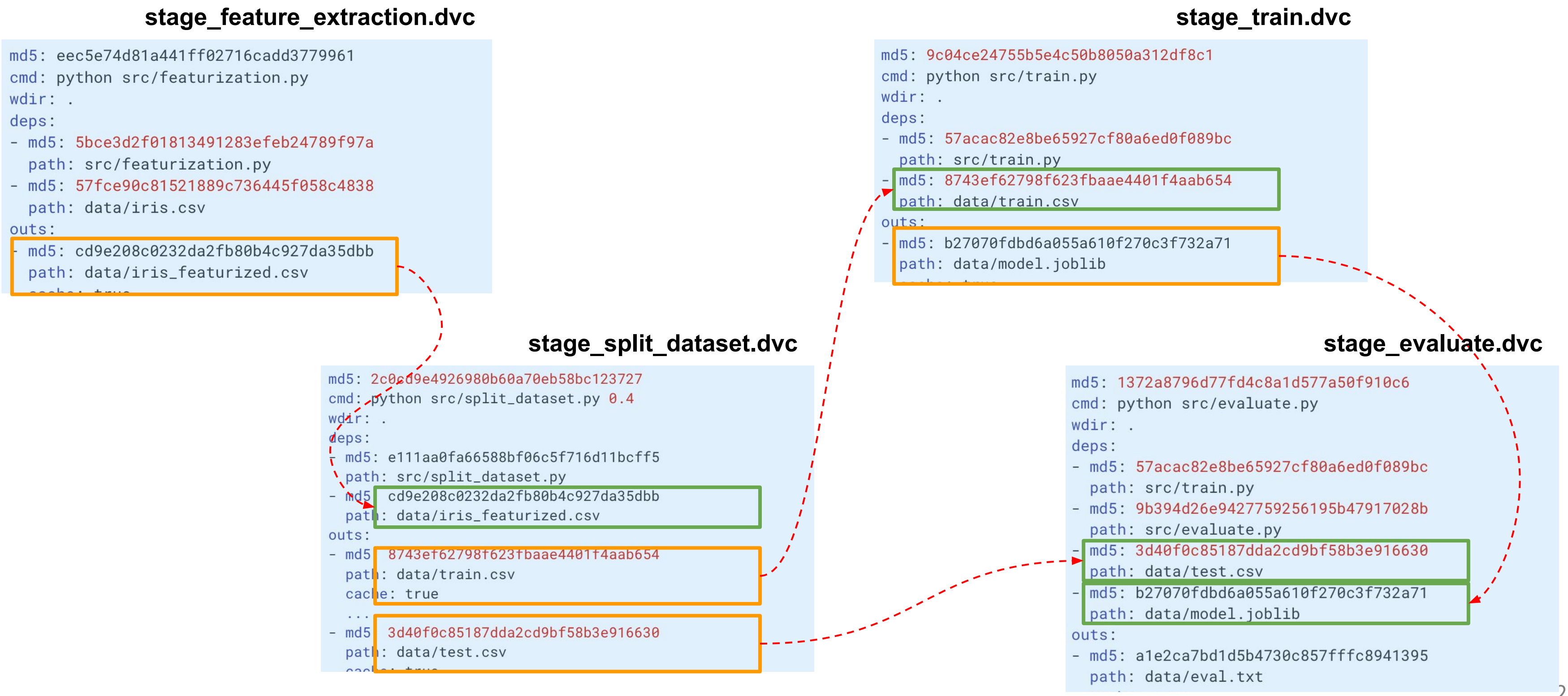
stage_train.dvc

```
md5: 9c04ce24755b5e4c50b8050a312df8c1
cmd: python src/train.py
wdir: .
deps:
- md5: 57acac82e8be65927cf80a6ed0f089bc
  path: src/train.py
- md5: 8743ef62798f623fbaae4401f4aab654
  path: data/train.csv
outs:
- md5: b27070fdbd6a055a610f270c3f732a71
  path: data/model.joblib
  cache: true
  metric: false
  persist: false
```

stage_evaluate.dvc

```
md5: 1372a8796d77fd4c8a1d577a50f910c6
cmd: python src/evaluate.py
wdir: .
deps:
- md5: 57acac82e8be65927cf80a6ed0f089bc
  path: src/train.py
- md5: 9b394d26e9427759256195b47917028b
  path: src/evaluate.py
- md5: 3d40f0c85187dda2cd9bf58b3e916630
  path: data/test.csv
- md5: b27070fdbd6a055a610f270c3f732a71
  path: data/model.joblib
outs:
- md5: a1e2ca7bd1d5b4730c857fffc8941395
  path: data/eval.txt
  cache: true
  metric: true
```

DVC resolves dependencies in ML pipeline



stage_feature_extraction.dvc

```
md5: eec5e74d81a441ff02716cadd3779961
cmd: python src/featurization.py
wdir: .
deps:
- md5: 5bce3d2f01813491283efeb24789f97a
  path: src/featurization.py
- md5: 57fce90c81521889c736445f058c4838
  path: data/iris.csv
outs:
- md5: cd9e208c0232da2fb80b4c927da35dbb
  path: data/iris_featurized.csv
  cache: true
```

stage_split_dataset.dvc

```
md5: 2c0cd9e4926980b60a70eb58bc123727
cmd: python src/split_dataset.py 0.4
wdir: .
deps:
- md5: e111aa0fa66588bf06c5f716d11bcff5
  path: src/split_dataset.py
- md5: cd9e208c0232da2fb80b4c927da35dbb
  path: data/iris_featurized.csv
outs:
- md5: 8743ef62798f623fbaae4401f4aab654
  path: data/train.csv
  cache: true
  ...
- md5: 3d40f0c85187dda2cd9bf58b3e916630
  path: data/test.csv
  cache: true
```

stage_train.dvc

```
md5: 9c04ce24755b5e4c50b8050a312df8c1
cmd: python src/train.py
wdir: .
deps:
- md5: 57acac82e8be65927cf80a6ed0f089bc
  path: src/train.py
- md5: 8743ef62798f623fbaae4401f4aab654
  path: data/train.csv
outs:
- md5: b27070fdbd6a055a610f270c3f732a71
  path: data/model.joblib
  cache: true
```

stage_evaluate.dvc

```
md5: 1372a8796d77fd4c8a1d577a50f910c6
cmd: python src/evaluate.py
wdir: .
deps:
- md5: 57acac82e8be65927cf80a6ed0f089bc
  path: src/train.py
- md5: 9b394d26e9427759256195b47917028b
  path: src/evaluate.py
- md5: 3d40f0c85187dda2cd9bf58b3e916630
  path: data/test.csv
- md5: b27070fdbd6a055a610f270c3f732a71
  path: data/model.joblib
outs:
- md5: a1e2ca7bd1d5b4730c857fffc8941395
  path: data/eval.txt
```

Step 4:

Metrics tracking

- specify metrics file with **-m**
- **dvc show metrics**
- **dvc show metrics -a**

Add stage with specified metrics (-m)

# run command

```
$ dvc run -f stage_evaluate.dvc \
    -d src/train.py \
    -d src/evaluate.py \
    -d data/test.csv \
    -d data/model.joblib \
    -m data/eval.txt \
    python src/evaluate.py
```

# run command

```
$ cat stage_evaluate.dvc
```

# output

```
md5: 2c5f02b139310b839b97f2a093b802b9
cmd: python src/evaluate.py
wdir: .
deps:
- md5: 025acbe1552887fab33f5314d036e907
  path: src/train.py
- ...
outs:
- md5: 1f7764d988d8d251dc3e9b1c5419f58b
  path: data/eval.txt
  cache: true
  metric: true
  persist: false
```

Metrics tracking

# run command

```
$ dvc metrics show
```

# output

```
data/eval.txt:
{"f1_score": 0.7861833464670345,
"confusion_matrix":
{"classes":
["setosa", "versicolor", "virginica"],
"matrix":
     [[23, 0, 0],
      [0, 8, 0],
      [0, 11, 18]]}}
```

Step 5:

Reproducibility

- how does it work?
- one command: **dvc repro**
- how to force reproducing the pipeline

How to reproduce a pipeline?

# run command

```
$ dvc repro stage_evaluate.dvc
```

# output

```
Stage 'data/iris.csv.dvc' didn't change.
Stage 'stage_feature_extraction.dvc' didn't change.
Stage 'stage_split_dataset.dvc' didn't change.
Stage 'stage_train.dvc' didn't change.
Stage 'stage_evaluate.dvc' didn't change.
Pipeline is up to date. Nothing to reproduce.
```

Step 6:

Checkout

- get into previous state
- start over a new experiment

Checkout into previous experiment state

# run command

```
$ git checkout dvc-tutorial
$ dvc checkout
```

# output

```
WARNING: data 'data/eval.txt' exists. Removing before checkout.
WARNING: data 'data/train.csv' exists. Removing before checkout.
WARNING: data 'data/test.csv' exists. Removing before checkout.
WARNING: data 'data/model.joblib' exists. Removing before checkout.
WARNING: data 'data/iris_featurized.csv' exists. Removing before checkout.
[##############################] 100% Checkout finished!
```

Step 7:

Share Data and Model Files

- use local/cloud remote storage
- push
- pull

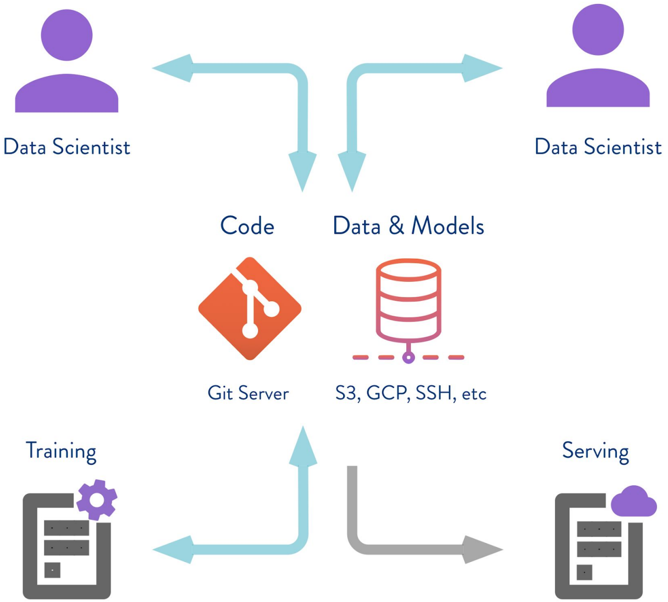


Image source: https://dvc.org/doc/use-cases/data-and-model-files-versioning

Push data to remote

/tmp/dvc used as a local ‘remote storage’ in this example

# run command

```
$ dvc push
```

# output

```
Preparing to upload data to '/tmp/dvc'
Preparing to collect status from /tmp/dvc
[##############################] 100% Collecting information
[##############################] 100% Analysing status.
(1/5): [##############################] 100% data/train.csv
(2/5): [##############################] 100% data/eval.txtturized.csv
(3/5): [##############################] 100% data/iris_featurized.csv
(4/5): [##############################] 100% data/test.csv
(5/5): [##############################] 100% data/model.joblib
```

Pull data from remote

/tmp/dvc used as a local ‘remote storage’ in this example

# run command

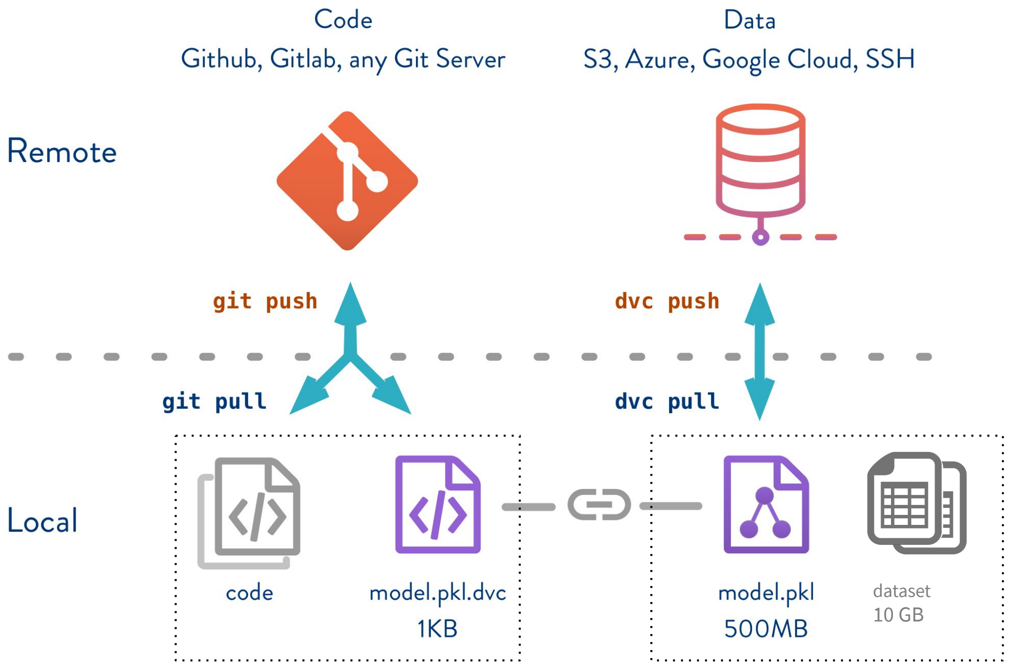
```
$ dvc pull
```

# output

```
Preparing to download data from '/tmp/dvc'
Preparing to collect status from /tmp/dvc
[##############################] 100% Collecting information
[##############################] 100% Analysing status.
[##############################] 100% Checkout finished!
```

Use case 1:

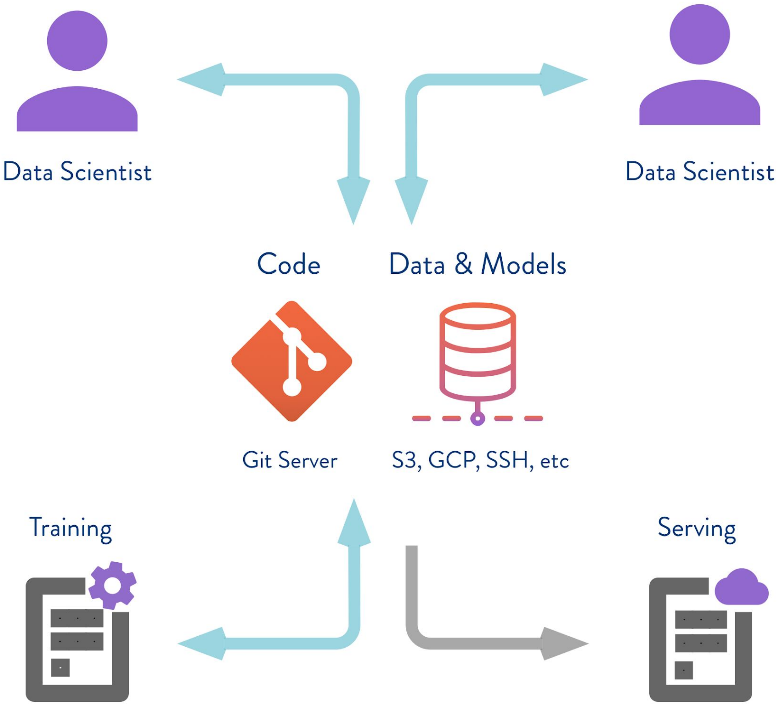
Data and Model Files Versioning



Original image source: https://dvc.org/doc/use-cases/data-and-model-files-versioning

Use case 2:

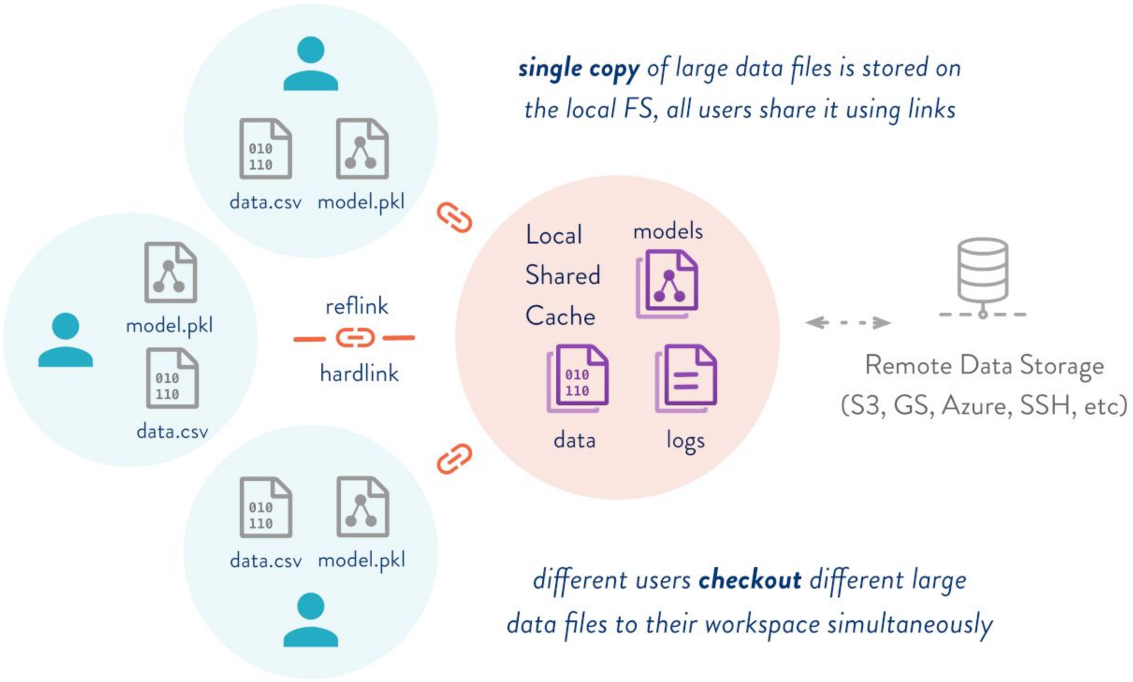
Share Data and Model Files



Original image source: https://dvc.org/doc/use-cases/share-data-and-model-files

Use case 3:

Teamwork with a Shared Development Server



single copy *of large data files is stored on the local FS, all users share it using links*

different users ***checkout*** *different large data files to their workspace simultaneously*

Image source: https://dvc.org/doc/use-cases/multiple-data-scientists-on-a-single-machine