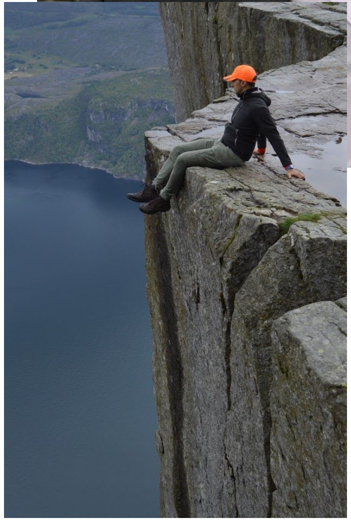
DataScienceSeed Online #10

Data Science, Machine Learning, Artificial Intelligence Meetup a Genova

MLOps, quando si smette di giocare

Model
Create
Plan
Release
Configure
Monitor

Simone Merello - Head of Deep AI, Perceptolab

Simone Merello - Head of Deep AI @ Perceptolab

Academic background:

- Computer scientist with focus on data science
- Researcher @ NTU university of Singapore

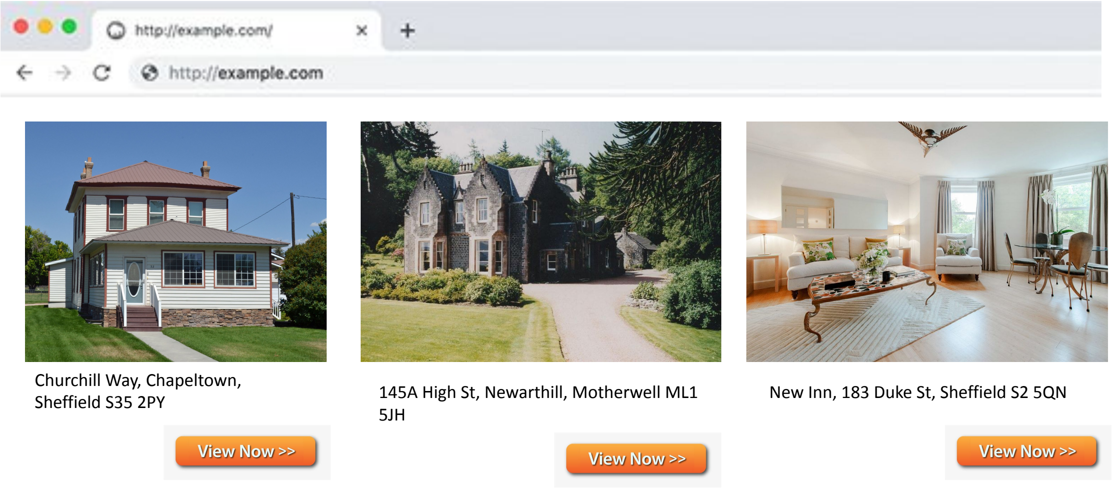
Now:

- Still ML Researcher with focus on Computer Vision: TensorFlow, Pytorch
- MLOps engineering to help team collaboration and automation of ML pipelines.

Leisure:

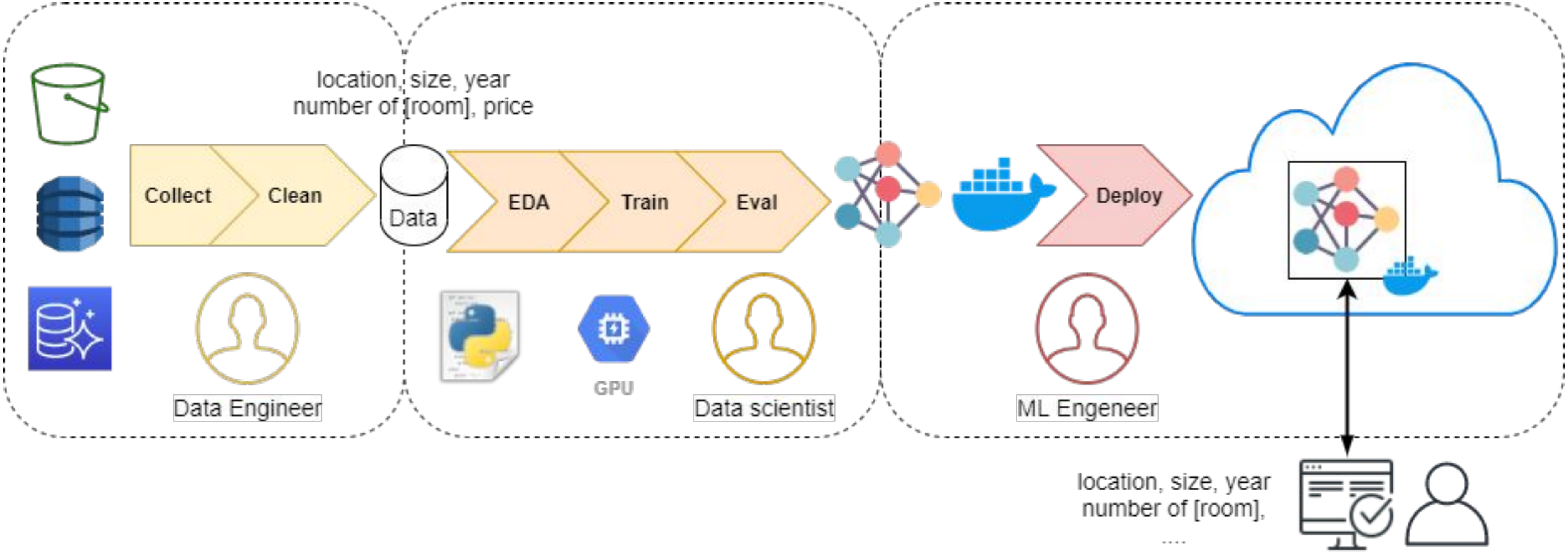
- Love traveling and love water sports!

A practical example



→ New feature: Real Estate Appraisal

ML Development cycle



But then something happens… [examples]

- New feature doesn’t work as expected → MONITORING
- More data comes in → FAST ITERATION
- New team members → NEED FOR COOPERATION
- Multiple projects / data / models / experiments → TRACKING
- Performance degradation with time → AUTOMATIC RE-TRAINING

ML Systems require organization!

[1] Some Issues of ML Systems

1. **Entanglement**: "Changing Anything Changes Everything”
2. **Tracking dependencies:** data, code, env, input models
3. **Cascading**: the output of a model A might affect input of an [undeclared] model B
4. **Feedback Loops**: models influencing each other if they update over time
5. **Staleness**: if the input changes during time, the model has to adapt

[2] ML Systems Best practices

1. **Data management:** Ensuring availability, accessibility, quality and versioning of data.
2. **Pipelines:** supporting data preprocessing, train, test and deployment
3. **Automation** of training and deployment pipelines allows fewer deployment issues

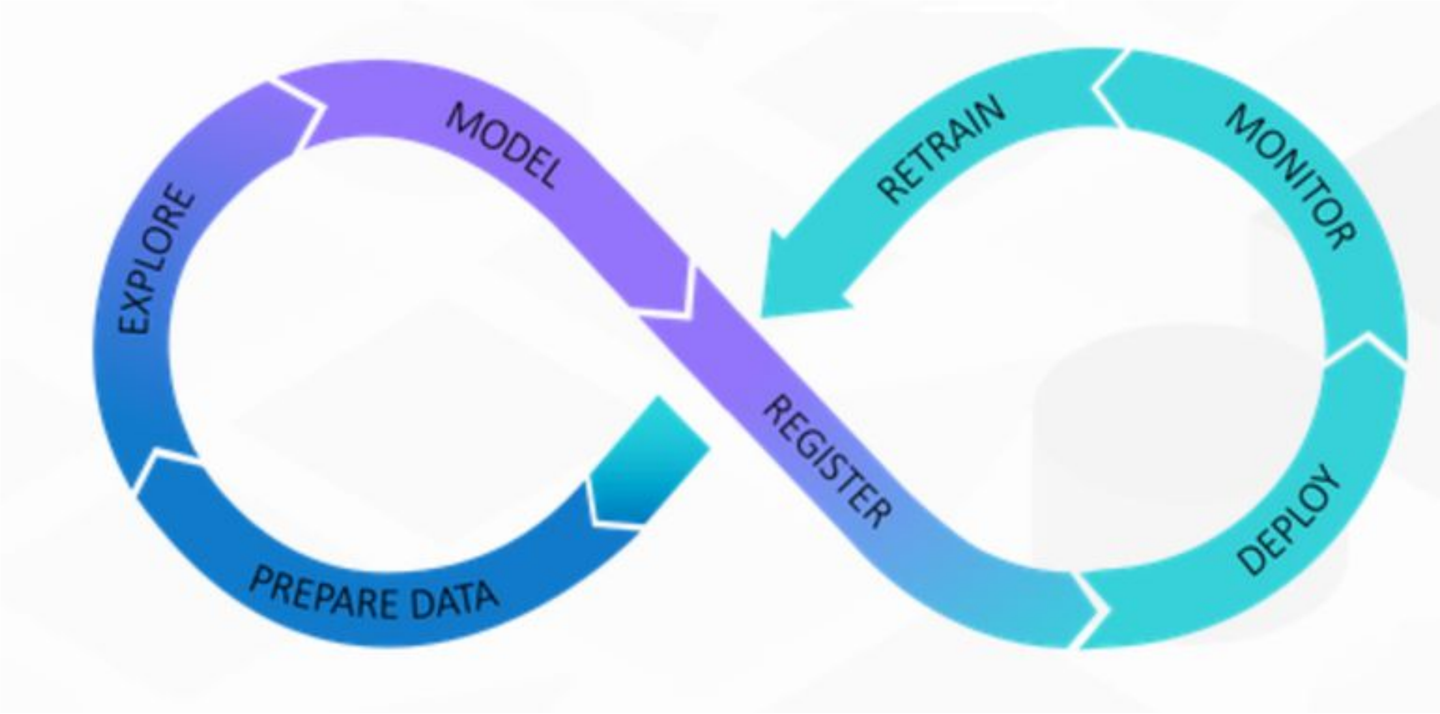
[3] ML Systems readiness

1. **Features and data:** assert expectations, cost/benefit tradeoff, fast addition of new features, tested features creation
2. **Model development:** versioning, evaluate {metrics = KPI, staleness, fairness}.
3. **Infrastructure:** reproducibility, integration tests, canary testing, quick rollback
4. **Monitoring:** monitor {changes in dependencies, input expectations, staleness}

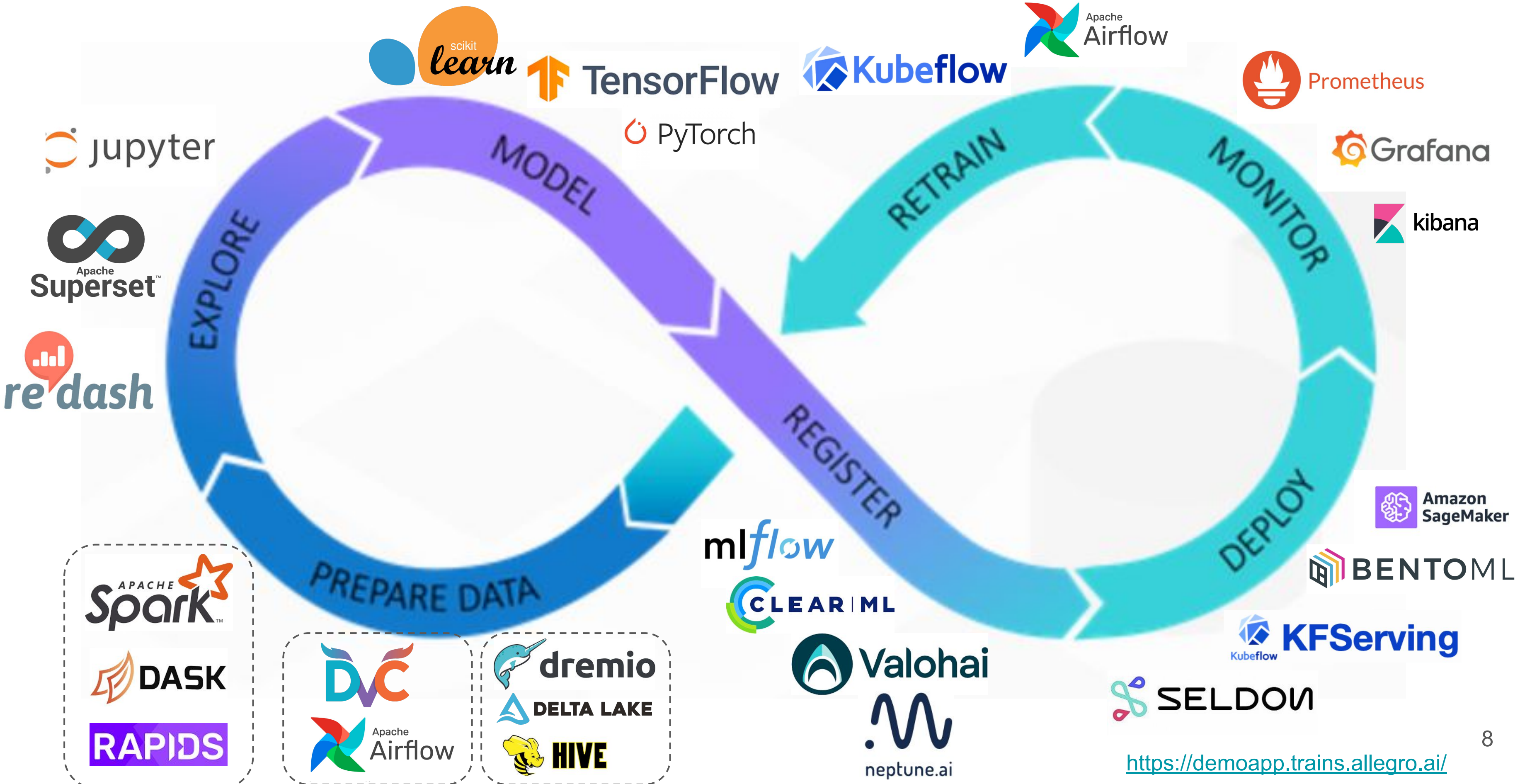
[1] Sculley, David, et al. "Hidden technical debt in machine learning systems” .2015
[2] Amershi, Saleema, et al. "Software engineering for machine learning: A case study." *2019*
[3] Breck, Eric, et al. "The ml test score: A rubric for ml production readiness and technical debt reduction." *2017*

MLOps

Management of ML Systems: operationalize the steps than produce, serve and improve ML models



Some tools [divided by their native usage]

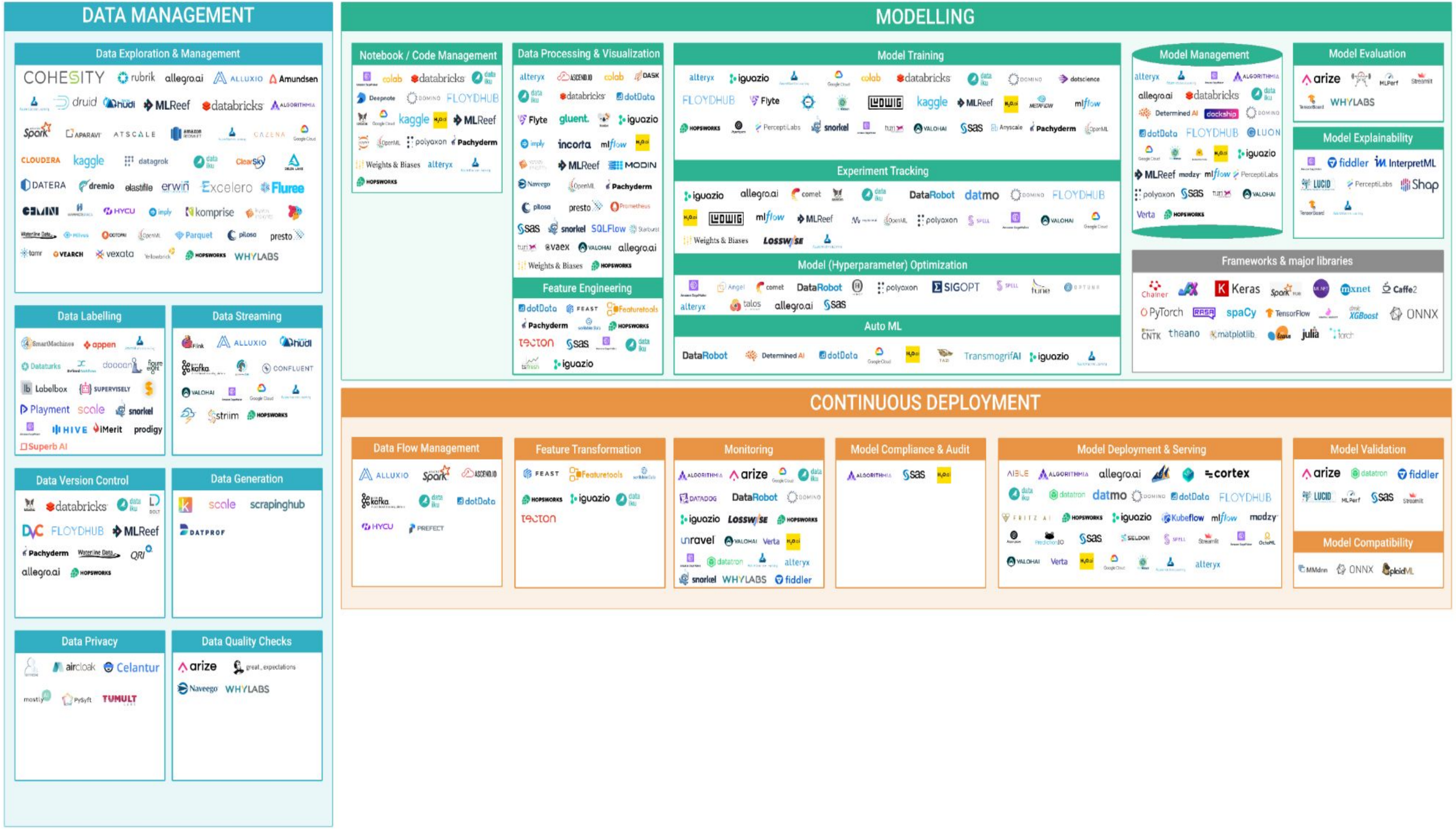


https://demoapp.trains.allegro.ai/

Tools landscape

https://mattturck.com/data2020/

ML tools



~300

	A	B	C
1	Name	Cat	SubCat
2	Abacus AI	All-in-one	AutoML
3	Accord	Modeling & Training	Framework
4	Aible	All-in-one	Serving
5	AIMET	Modeling & Training	Model compression
6	Aircloak	Data pipeline	Privacy
7	Airflow	Infrastructure	Workflow orchestration
8	Alectio	Modeling & Training	Active learning
9	Algorithmia	Serving	Serving
10	Alink	Modeling & Training	Framework
11	Allegro AI/TRAIN	Modeling & Training	Experiment tracking
12	AllenNLP	Modeling & Training	NLP
13	Alluxio	Data pipeline	Data management
14	Alteryx	Data pipeline	Data management
15	Amazon Redshift	Data pipeline	Data warehouse
16	Amundsen	Data pipeline	Database/Query
17	Angel ML	Modeling & Training	Distributed
18	Anodot	Data pipeline	Data monitoring
19	Anyscale	Infrastructure	Cloud management
20	Apache Druid	Data pipeline	Database/Query
21	Apache Flink	Serving	Stream processing
22	Apache Hudi	Data pipeline	Data warehouse
23	Apache Kafka	Serving	Stream storage
24	Apache Mahout	Modeling & Training	Framework
25	Apache MXNet	Modeling & Training	Framework
26	Apache ORC	Data pipeline	File format

https://about.mlreef.com/images/blog/ml_landscape.png

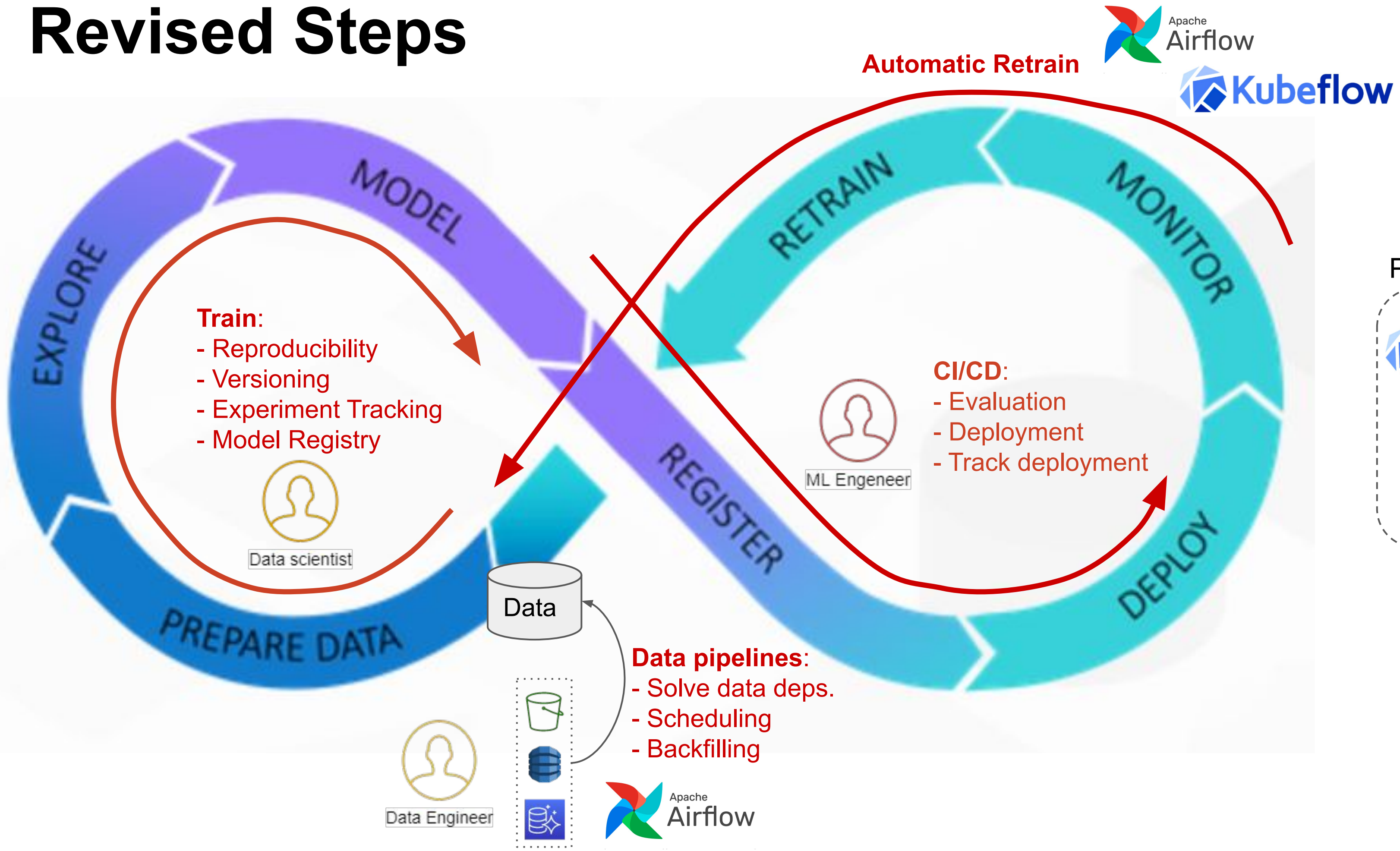
https://docs.google.com/spreadsheets/

[SOME] Features you might look for

ML PERSPECTIVE FOR BATCH LEARNING TASKS		
DATA	MODELLING	DEPLOY [1]
Versioning	Track experiments	Automated CI/CD
Availability	Reproducibility (dependencies)	Track Deployment
Training - Serving consistency	Track outputs (models / performances)	Canary / Shadow testing
Schedule jobs	Compare experiments	Automatic Retraining
Exploration	Hyperparams Optimization	Monitoring data (outliers / dist. shift)
Data quality checks	Infrastructure handling	Monitoring model performances
Cataloging	Peer reviewing	Explaining predictions
Labelling		Automatic Scalability
Handle Real time data		
Infrastructure		
Storage scalability		

[1] Klaise, Janis, et al. "Monitoring and explainability of models in production." arXiv preprint arXiv:2007.06299 (2020).

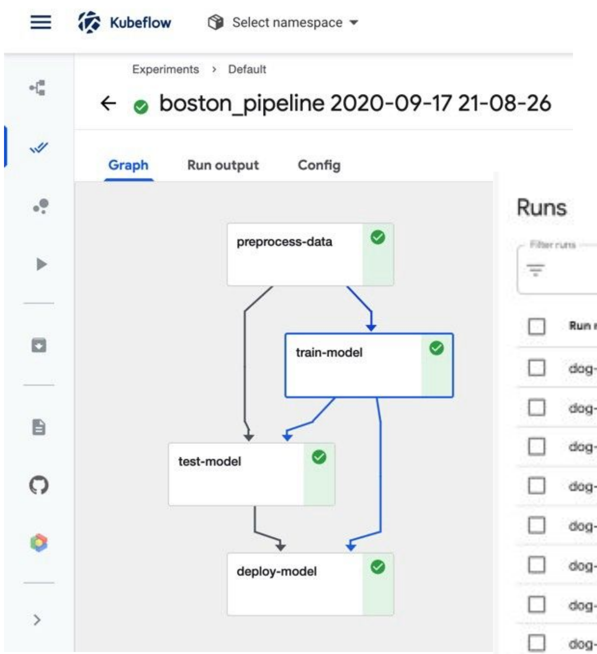
Revised Steps

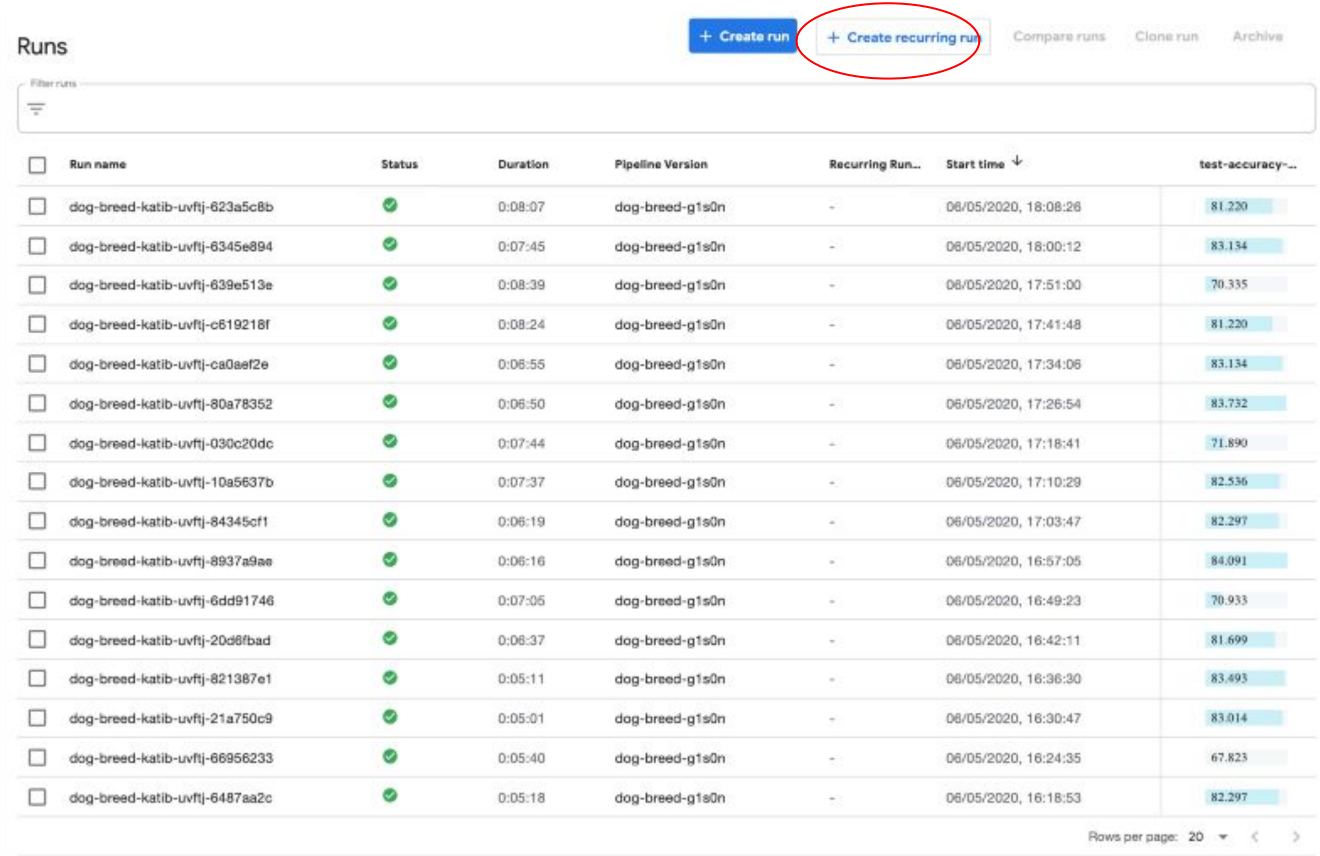


Pipeline tools

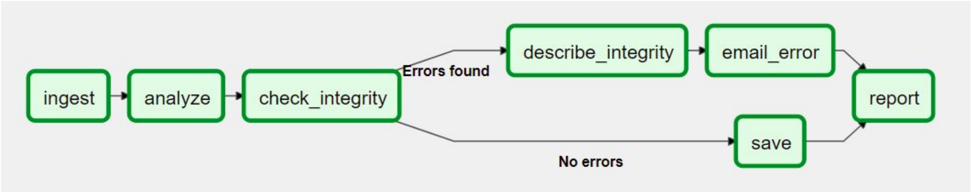
Kubeflow
Apache Airflow
DVC

KubeFlow: Pipelines, **Notebook Servers**, **Katib** (hyperparameter tuning)**, Artifact Store**, **KFServing**

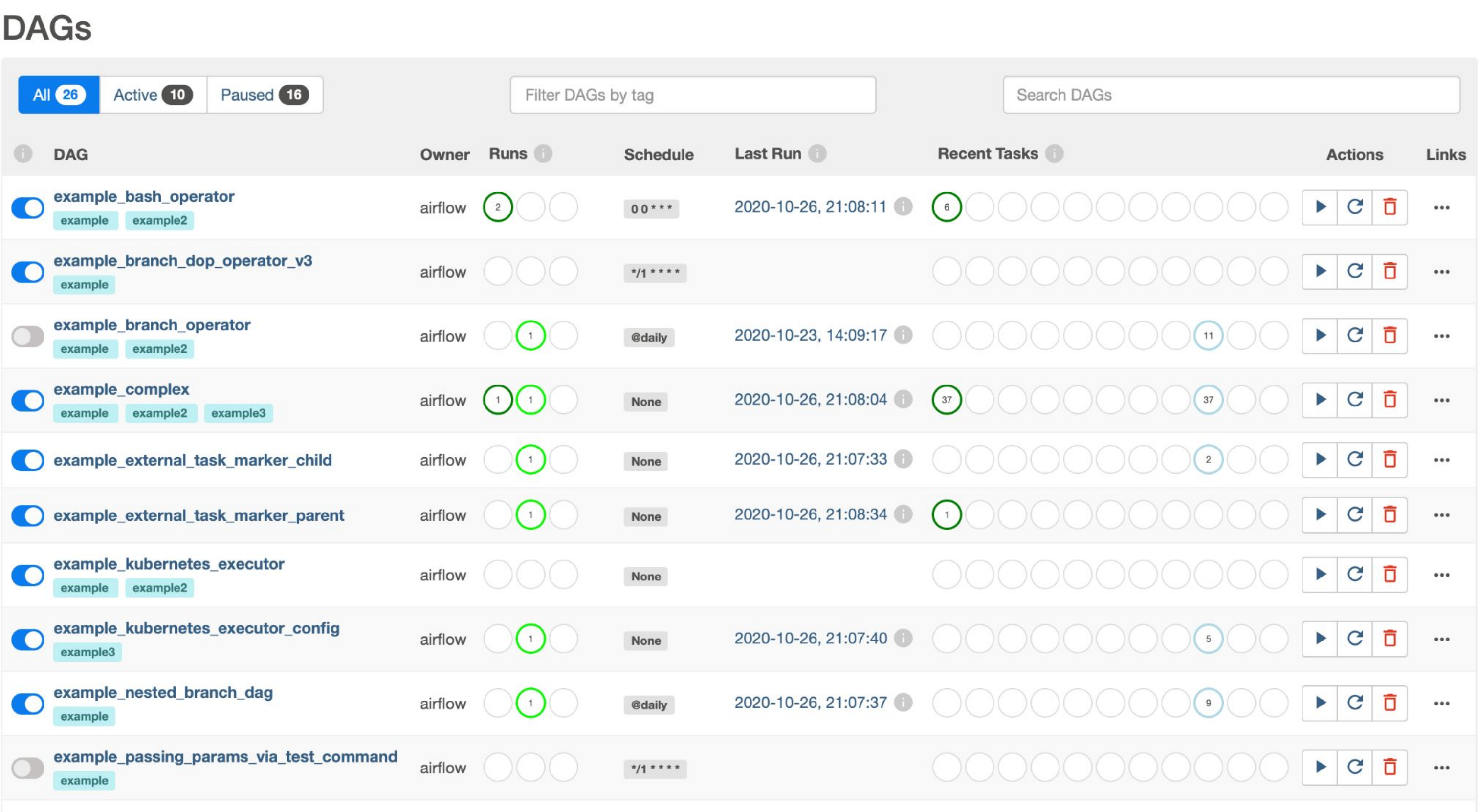




Run name	Status	Duration	Pipeline Version	Recurring Run...	Start time	test-accuracy-...
dog-breed-katib-uvftj-623a5c8b		0:08:07	dog-breed-g1s0n	-	06/05/2020, 18:08:26	81.220
dog-breed-katib-uvftj-6345e894		0:07:45	dog-breed-g1s0n	-	06/05/2020, 18:00:12	83.134
dog-breed-katib-uvftj-639e513e		0:08:39	dog-breed-g1s0n	-	06/05/2020, 17:51:00	70.335
dog-breed-katib-uvftj-c619218f		0:08:24	dog-breed-g1s0n	-	06/05/2020, 17:41:48	81.220
dog-breed-katib-uvftj-ca0aef2e		0:06:55	dog-breed-g1s0n	-	06/05/2020, 17:34:06	83.134
dog-breed-katib-uvftj-80a78352		0:06:50	dog-breed-g1s0n	-	06/05/2020, 17:26:54	83.732
dog-breed-katib-uvftj-030c20dc		0:07:44	dog-breed-g1s0n	-	06/05/2020, 17:18:41	71.890
dog-breed-katib-uvftj-10a5637b		0:07:37	dog-breed-g1s0n	-	06/05/2020, 17:10:29	82.536
dog-breed-katib-uvftj-84345cf1		0:06:19	dog-breed-g1s0n	-	06/05/2020, 17:03:47	82.297
dog-breed-katib-uvftj-8937a9ae		0:06:16	dog-breed-g1s0n	-	06/05/2020, 16:57:05	84.091
dog-breed-katib-uvftj-6dd91746		0:07:05	dog-breed-g1s0n	-	06/05/2020, 16:49:23	70.933
dog-breed-katib-uvftj-20d6fbad		0:06:37	dog-breed-g1s0n	-	06/05/2020, 16:42:11	81.699
dog-breed-katib-uvftj-821387e1		0:05:11	dog-breed-g1s0n	-	06/05/2020, 16:36:30	83.493
dog-breed-katib-uvftj-21a750c9		0:05:01	dog-breed-g1s0n	-	06/05/2020, 16:30:47	83.014
dog-breed-katib-uvftj-66956233		0:05:40	dog-breed-g1s0n	-	06/05/2020, 16:24:35	67.823
dog-breed-katib-uvftj-6487aa2c		0:05:18	dog-breed-g1s0n	-	06/05/2020, 16:18:53	82.297



Airflow: DAG (chained Operators), **Scheduler, Executor**



DAGs

All 26 | Active 10 | Paused 16

Filter DAGs by tag | Search DAGs

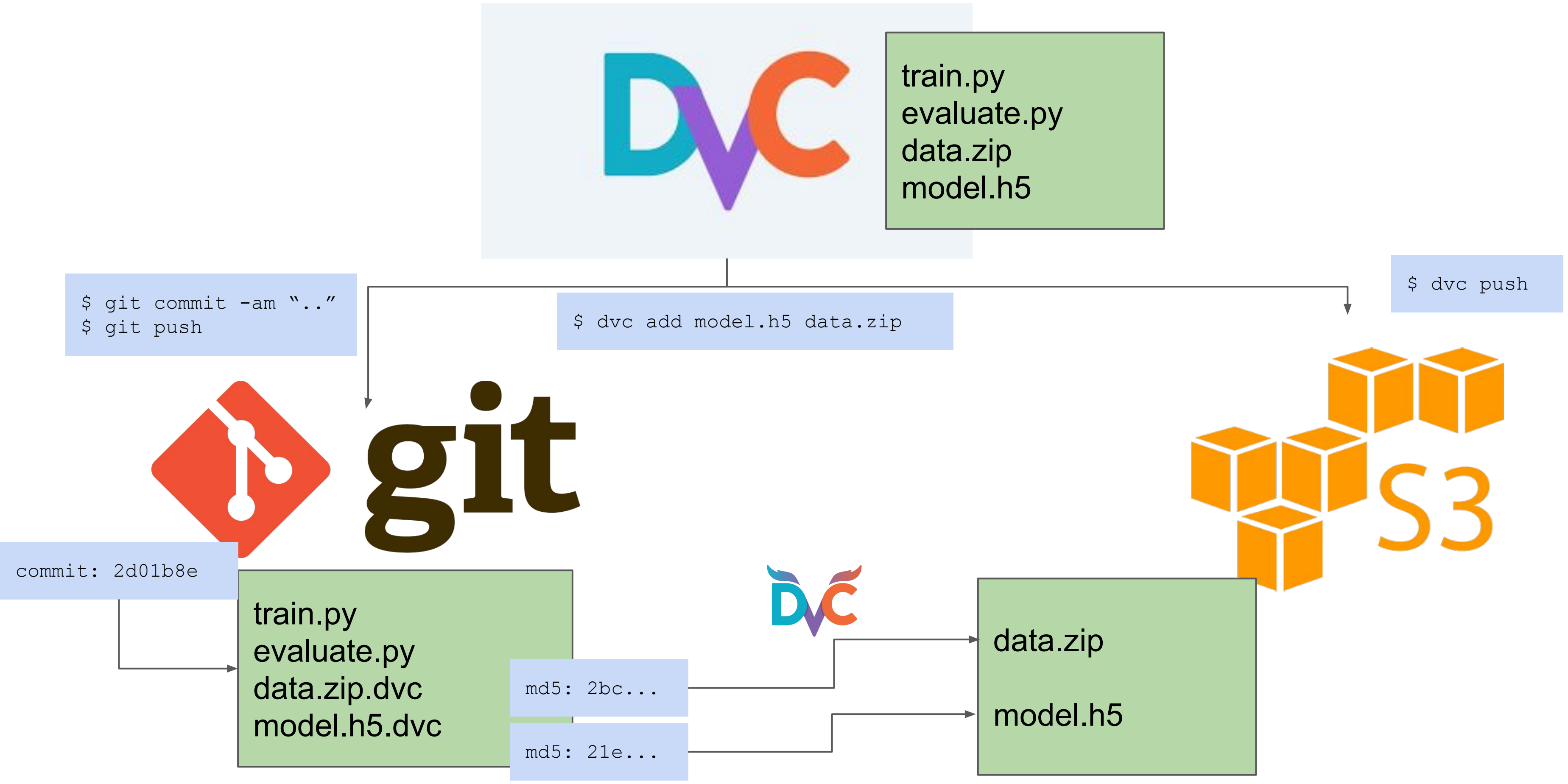
DAG	Owner	Runs	Schedule	Last Run	Recent Tasks	Actions	Links
example_bash_operator (example, example2)	airflow	2	0 0 * * *	2020-10-26, 21:08:11	6		...
example_branch_dop_operator_v3 (example)	airflow		*/1 * * * *				...
example_branch_operator (example, example2)	airflow	1	@daily	2020-10-23, 14:09:17	11		...
example_complex (example, example2, example3)	airflow	1 1	None	2020-10-26, 21:08:04	37 37		...
example_external_task_marker_child	airflow	1	None	2020-10-26, 21:07:33	2		...
example_external_task_marker_parent	airflow	1	None	2020-10-26, 21:08:34	1		...
example_kubernetes_executor (example, example2)	airflow		None				...
example_kubernetes_executor_config (example3)	airflow	1	None	2020-10-26, 21:07:40	5		...
example_nested_branch_dag (example)	airflow	1	@daily	2020-10-26, 21:07:37	9		...
example_passing_params_via_test_command (example)	airflow		*/1 * * * *				...

DVC

Why?

1. Easy to setup and use
   `$ pip install dvc`
2. Can be used for many MLOps steps
3. Is it the best one? NO (It depends on your needs)

Experiment & Data versioning



Pipelines: Training & CI/CD

dvc.yaml

```
stages:
 [...]
 featurize:
    cmd: python features.py data/ features
    deps:
    - data/
    - features.py
    outs:
    - features/
  train:
    cmd: python train.py features model.pkl
    deps:
    - features
    - train.py
    outs:
    - model.pkl
[...]
```

Rerun if something change

```
          +---------+
          | prepare |
          +---------+
               *
               *
               *
         +-----------+
         | featurize |
         +-----------+
          **        **
        **           *
       *              **
+-------+               *
| train |              **
+-------+             *
          **        **
            **    **
              *  *
         +----------+
         | evaluate |
         +----------+
```

```
$ dvc repro
```

Git commit: 2d01b8e

```
stages:
 featurize:
    cmd: python featurization.py data/ features/
    deps:
    - path: data/
      md5: 20b78...
    - path: featurization.py
      md5: 28946...
    outs:
    - path: features/
      md5: 52c1f...
  train:
    cmd: python train.py features/ model.pkl
    deps:
    - path: features/
      md5: 52c1f...
    - path: train.py
      md5: 3ffc5...
    outs:
    - path: model.pkl
      md5: b4c48...
```

Experimentation: Pick up the best model

```
$ dvc exp run --queue -S train.min_split=8
Queued experiment 'd3f6d1e' for future execution.
$ dvc exp run --queue -S train.min_split=64
Queued experiment 'f1810e0' for future execution.
$ dvc exp run --queue -S train.min_split=2 -S train.n_est=100
Queued experiment '7323ea2' for future execution.
$ dvc exp run --queue -S train.min_split=8 -S train.n_est=100
Queued experiment 'c605382' for future execution.
$ dvc exp run --queue -S train.min_split=64 -S train.n_est=100
Queued experiment '0cdee86' for future execution.
$ dvc exp run --run-all --jobs 2
```

```
$ dvc exp show --no-timestamp \
          --include-params train.n_est,train.min_split
```

Experiment	avg_prec	roc_auc	train.n_est	train.min_split
workspace	0.56191	0.93345	50	2
master	0.55259	0.91536	50	2
├── exp-bfe64	0.57833	0.95555	50	8
├── exp-b8082	0.59806	0.95287	50	64
├── exp-c7250	0.58876	0.94524	100	2
├── exp-b9cd4	0.57953	0.95732	100	8
├── exp-98a96	0.60405	0.9608	100	64
└── exp-ad5b1	0.56191	0.93345	50	2

```
$ dvc exp apply exp-98a96
```

Deploy: CI/CD pipeline

DVC Pipeline

```
stages:
 test_performances:
   cmd: python test_performances.py model.pkl
   deps:
   - test_performances.py
   - model.pkl
   outs:
   - test_result.md
 deploy:
   cmd: python deploy.py test_result.txt model.pkl
   deps:
   - deploy.py
   - test_result.md
   - model.pkl
```

CML with DVC

```
name: train-my-model
on: [push]
jobs:
  run:
     runs-on: [ubuntu-latest]
     container: docker://dvcorg/cml-py3:latest
     steps:
        - uses: actions/checkout@v2
        - name: cml_run
          env:
             repo_token: ${{ secrets.GITHUB_TOKEN }}
          run: |
             dvc pull model.pkl
             dvc repro
             git config [...]
             git add dvc.lock test_results.txt
             git commit "CI/CD pipeline" --allow-empty
             git push -u origin HEAD"
```

```
Git commit:
2d01b8e
```

→

```
deploy:
    deps:
    - path: model.pkl
      md5: 20b...
    - test_results.txt
      md5: 21e...
```

How to begin:

1. **Tools must be useful**: reduce troubles and takes less time from the team, not more

2. **Start manually, then automate**: difficult to choose what to automate without knowing what issues are there

3. **Consider lock-ins:** easier to adopt a new tool than to leave it

4. **Give some extra points to “mature” tools**

Simone Merello
Head of Deep AI,
Perceptolab

Really happy to discuss about these topics further!
simone.merello@smartlab.ws