ML experiments management, pipelines automation and reproducibility:

with DVC and MLFlow

Mikhail Rozhkov

ml-repa.ru

How do we want to experiment?

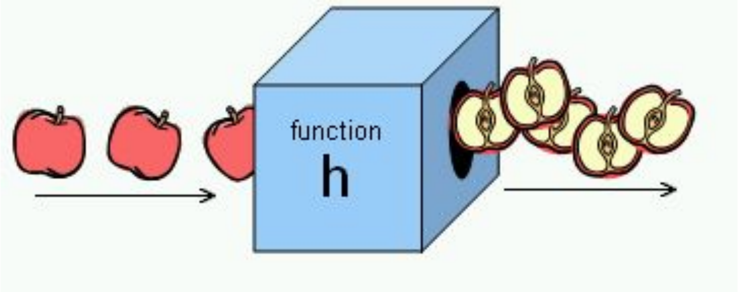
What	Why
Reproducible ML pipelines	Debug / Trust / Consistency / Performance
Save experiment metadata (params, metrics, versions, dependencies…)	Debug / Reproducibility
Store Models and Artifacts	Debug / Inference / Reproducibility
Models and Artifacts version control	Debug / Roll-back / Experimentation
Experiment results tracking	Experiment runs benchmark

Credits to Vlad Grozin: MLFlow for reproducible experiments, ML REPA #2 talk

ML reproducibility is about quality

What is Reproducibility?

using the original methods applied to the original data to produce the original results [Gardner]



Why should you care?

- Trust
- Consistent Results
- Versioned History
- Team Performance

Josh Gardner, Yuming Yang, Ryan S. Baker, Christopher Brooks. Enabling End-To-End Machine Learning Replicability: A Case Study in Educational Data Mining

ML Experiment Management checklist

1. Automated pipelines
2. Control run params
3. Control execution DAG
4. Code version control
5. Artifacts version control (models, datasets, etc.)
6. Use shared/cloud storage for artifacts
7. Environment dependencies control
8. Experiments results tracking

Upgrade ML experiments

use case

- automated pipelines
- reproducibility
- experiments management

Use Case:

Iris Flowers Classification

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

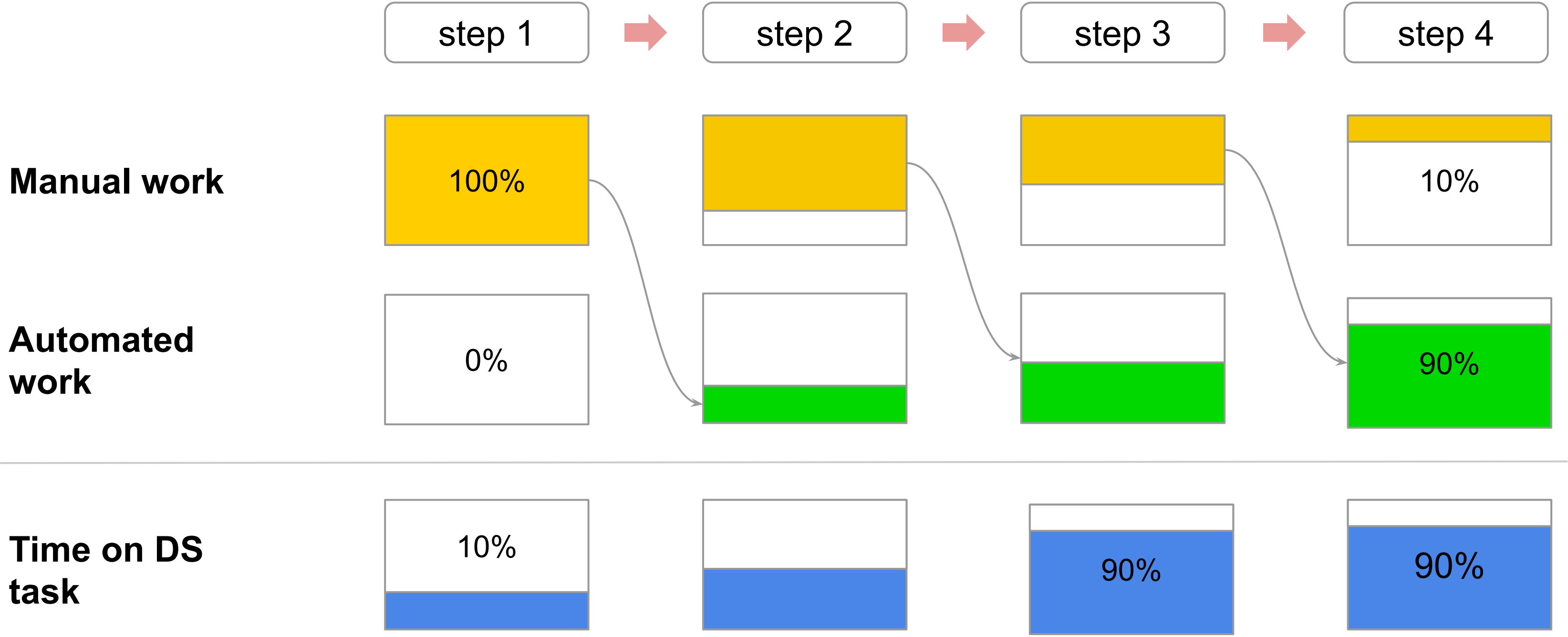
References:

- 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

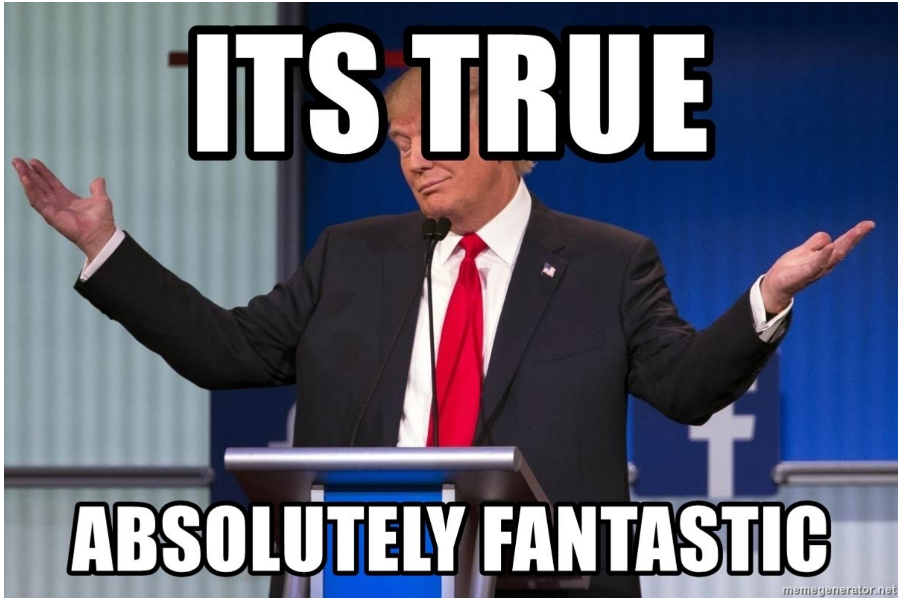
How to start?


HELP!


step 1
step 2
step 3
step 4
Manual work
100%
10%
Automated work
0%
90%
Time on DS task
10%
90%
90%

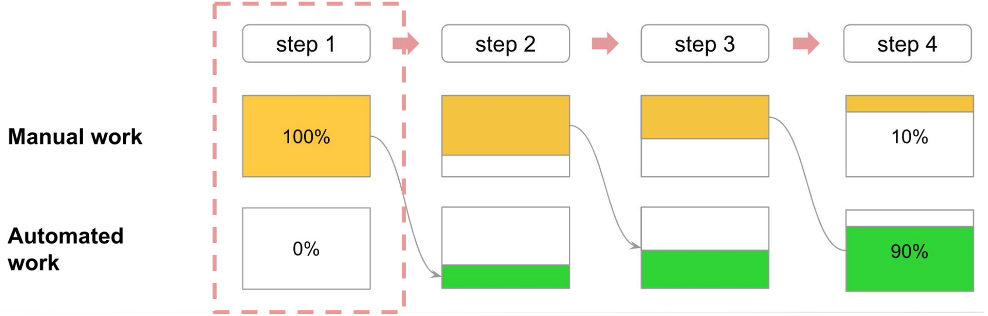
Step 0: AS IS

- common ML practices
- all code in on Jupyter Notebook



Step 1:
Jupyter Notebook

- code in Jupyter Notebook
- everything in Docker

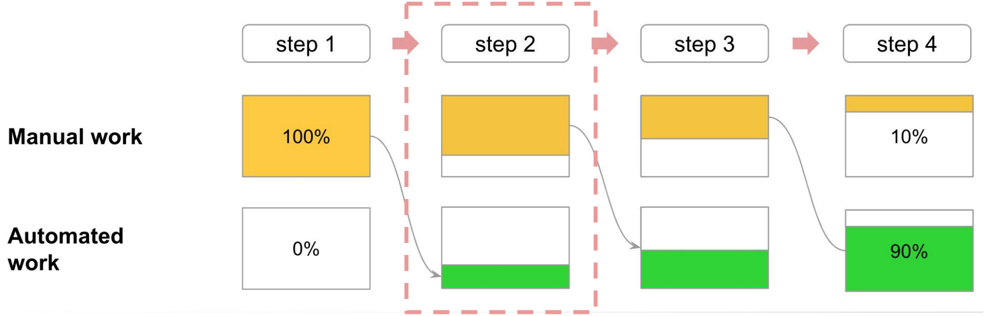


ML Experiment Management checklist

1. Automated pipelines
2. Control run params
3. Control execution DAG
4. Code version control
5. Artifacts version control (models, datasets, etc.)
6. Use shared/cloud storage for artifacts
7. Environment dependencies control
8. Experiments results tracking

Step 2: build pipelines

- move common code into .py modules
- build pipelines
- everything in Docker
- run experiments in terminal or Jupyter Notebook

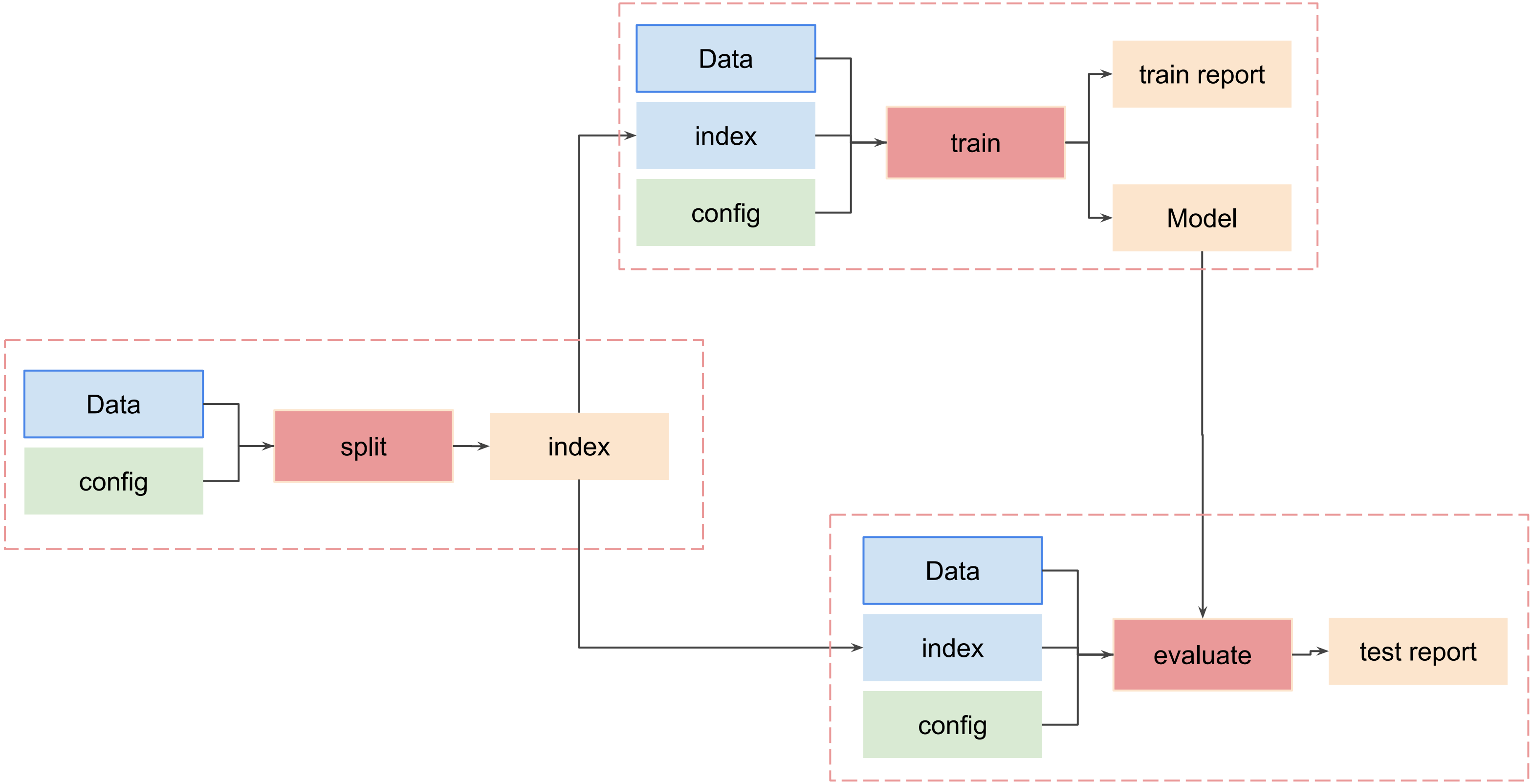


Project structure

```
.
├── README.md
├── config  <- any configuration files
├── data
│     ├── external <- external data
│     ├── interim <- data in intermediate processing stage
│     ├── processed <- data after all preprocessing has been done
│     └── raw <- original unmodified data acting as source of truth and provenance
├── docker <- docker image(s) for running project inside container(s)
├── docs  <- usage documentation or reference papers
├── models  <- compiled model .pkl or HDFS or .pb format
├── notebooks <- jupyter notebooks for exploratory analysis and explanation
└── src
      ├── data <- data prepare and/or preprocess
      ├── evaluate <- evaluating model stage code
      ├── features <- code to compute features
      ├── pipelines <- scripts of pipelines
      ├── report <- visualization (used in notebooks)
      ├── train <- train model stage code
      ├── transforms <- transformations data code (e.g., augmentation)
      └── utils.py <- auxiliary functions and classes
```

not sync with GIT

Setup pipelines

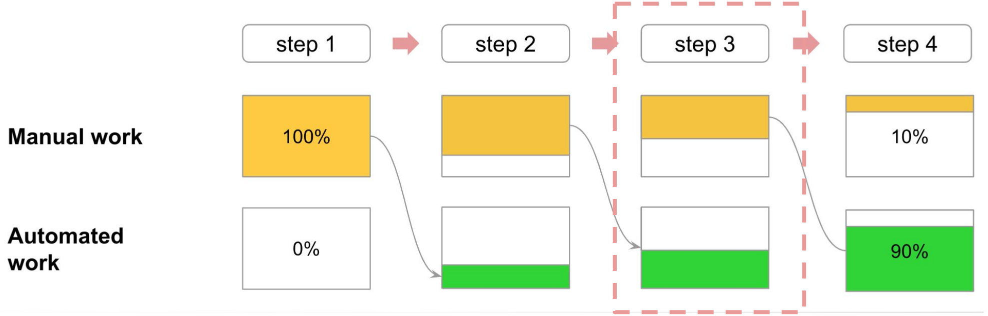


ML Experiment Management checklist

1. Automated pipelines
2. Control run params
3. Control execution DAG
4. Code version control
5. Artifacts version control (models, datasets, etc.)
6. Use shared/cloud storage for artifacts
7. Environment dependencies control
8. Experiments results tracking

Step 3:
add version control for artifacts

- add models/data/congis under DVC control
- same code in .py modules
- same pipelines
- everything in Docker
- run experiments in terminal or Jupyter Notebook

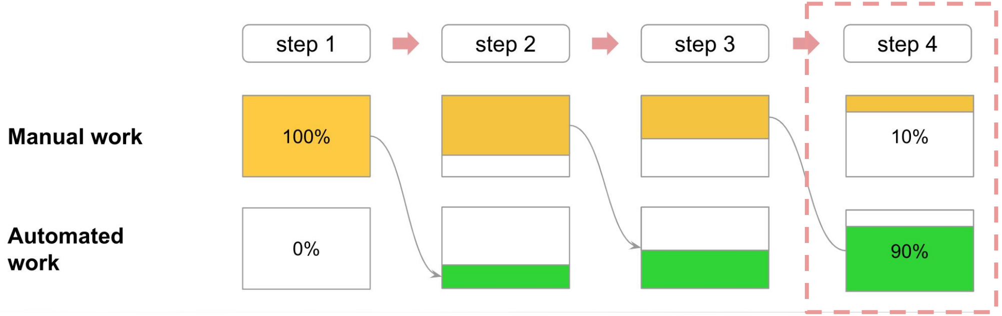


ML Experiment Management checklist

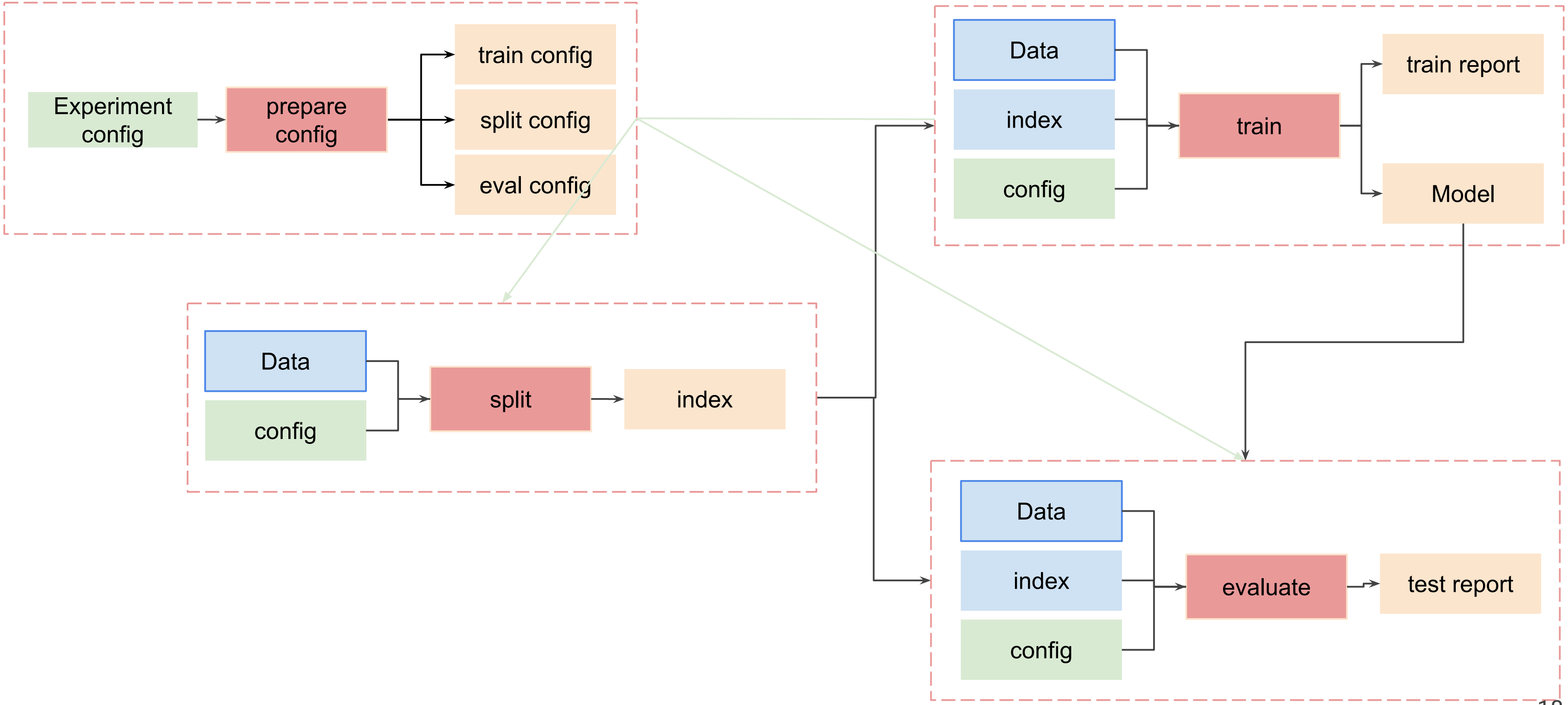
1. Automated pipelines
2. Control run params
3. Control execution DAG
4. Code version control
5. Artifacts version control (models, datasets, etc.)
6. Use shared/cloud storage for artifacts
7. Environment dependencies control
8. Experiments results tracking

Step 4: add execution DAG control

- add pipelines dependencies under DVC control
- models/data/congis under DVC control
- same code in .py modules
- same pipelines
- everything in Docker
- run experiments in terminal or Jupyter Notebook



Setup pipelines

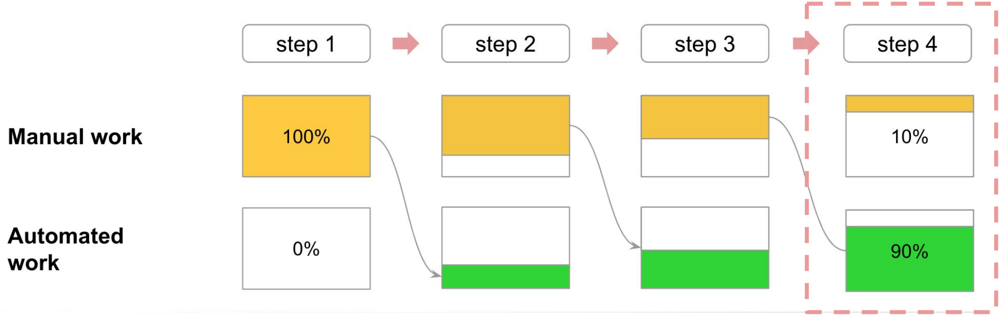


ML Experiment Management checklist

1. Automated pipelines
2. Control run params
3. Control execution DAG
4. Code version control
5. Artifacts version control (models, datasets, etc.)
6. Use shared/cloud storage for artifacts
7. Environment dependencies control
8. Experiments results tracking

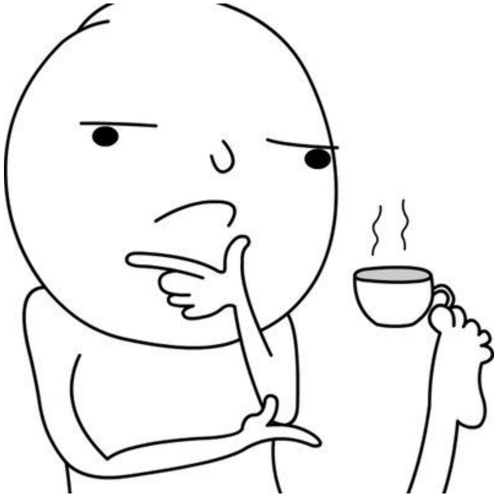
Step 5:
add experiments control

- add experiments benchmark (DVC, mlflow)
- pipelines dependencies under DVC control
- models/data/congis under DVC control
- same code in .py modules
- same pipelines
- everything in Docker
- run experiments in terminal or Jupyter Notebook



Compare experiments

- **dvc metrics show**
- **dvc metrics show** -a
- **dvc metrics show** -t json -x f1_score -a
- **dvc metrics show** -T



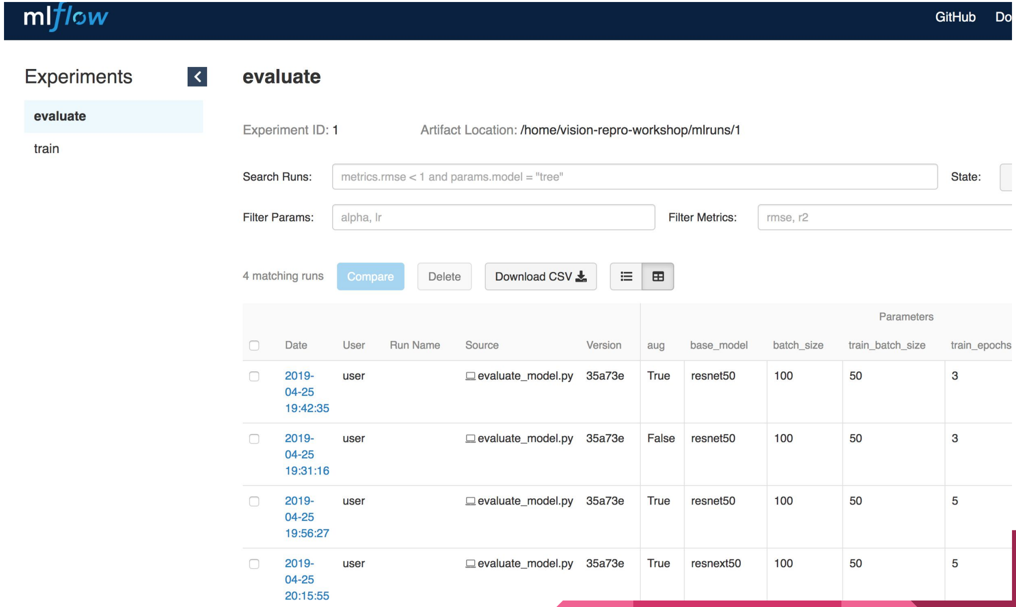
Metrics tracking in mlflow UI

```
from mlflow import log_metric, log_param,
log_artifact


log_artifact(args.config)


log_param('batch_size', config['batch_size'])


log_metric('f1', f1)
log_metric('roc_auc', roc_auc)
```



Experiments benchmarking

params

metrics

runs

				Parameters		Metrics		
Date	User	Source	Version	alpha	l1_ratio	mae	r2	rmse
2018-06-04 23:00:10	mlflow	train.py	05e956	1	1	0.649	0.04	0.862
2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.5	0.648	0.046	0.859
2018-06-04 23:00:10	mlflow	train.py	05e956	1	0.2	0.628	0.125	0.823
2018-06-04 23:00:09	mlflow	train.py	05e956	1	0	0.619	0.176	0.799
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	1	0.648	0.046	0.859
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.5	0.628	0.127	0.822
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0.2	0.621	0.171	0.801
2018-06-04 23:00:09	mlflow	train.py	05e956	0.5	0	0.615	0.199	0.787
2018-06-04 23:00:09	mlflow	train.py	05e956	0	1	0.578	0.288	0.742
2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.5	0.578	0.288	0.742
2018-06-04 23:00:09	mlflow	train.py	05e956	0	0.2	0.578	0.288	0.742
2018-06-04 23:00:08	mlflow	train.py	05e956	0	0	0.578	0.288	0.742

ML Experiment Management checklist

1. Automated pipelines
2. Control run params
3. Control execution DAG
4. Code version control
5. Artifacts version control (models, datasets, etc.)
6. Use shared/cloud storage for artifacts
7. Environment dependencies control
8. Experiments results tracking

Conclusions

1. ML experiments require an engineering approach
2. Reproducibility and automation are important
3. Start where you detect a "copy-paste" pattern
4. Version models and artifacts

Links

- Automate ML experiments with DVC_v3 slides
- Data Version Control (DVC): Tutorial 1: Get Started
- Data Version Control (DVC): Tutorial 2: Iris Demo Project