Data Version Control (DVC):
Tutorial 2: Iris Demo Project

Mikhail Rozhkov

2019

DVC core concepts

- Experiment
- State (experiment state)
- Reproducibility
- Pipeline
- Workflow
- Data files
- Data cache
- Cloud storage

Source:
https://dvc.org/doc/understanding-dvc/what-is-dvc

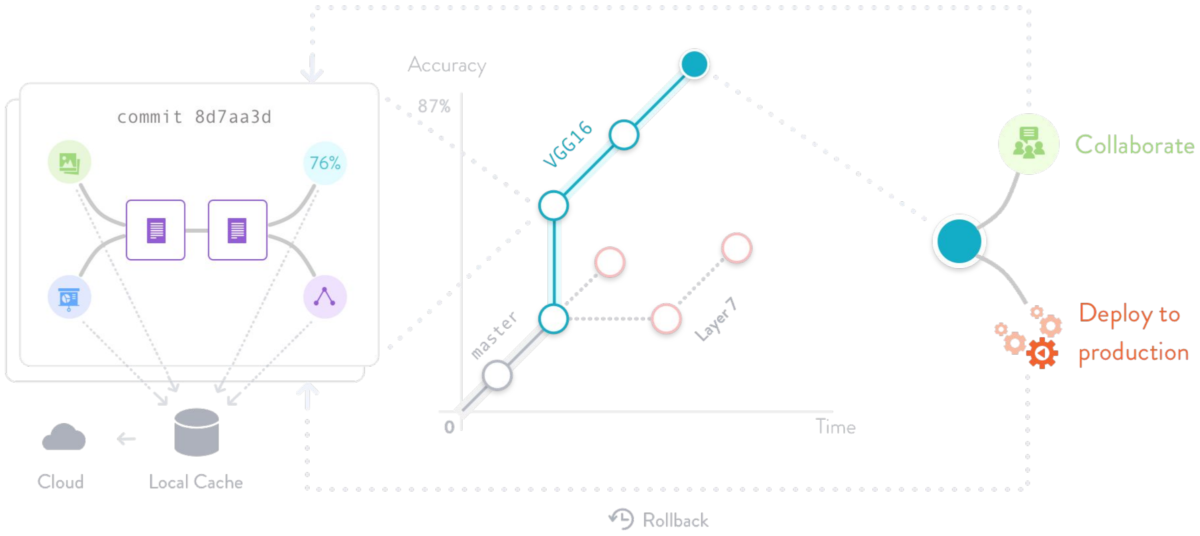


Image source: https://dvc.org/doc/tutorial

Experimentation methodology

DVC core concepts

- Experiment
- State (experiment state)
- Reproducibility
- Pipeline
- Workflow
- Data files
- Data cache
- Cloud storage

Source: https://dvc.org/doc/understanding-dvc/what-is-dvc

DVC tracks ML models and data sets

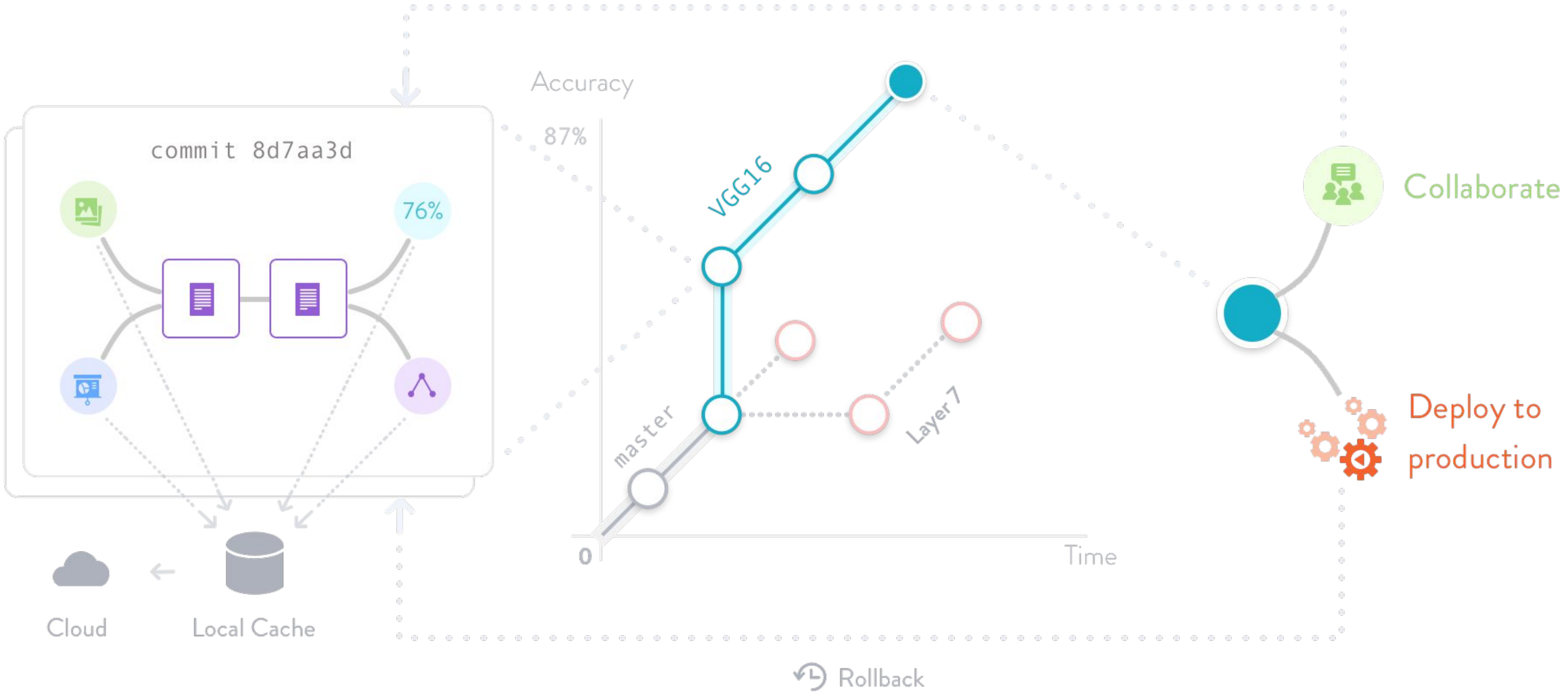
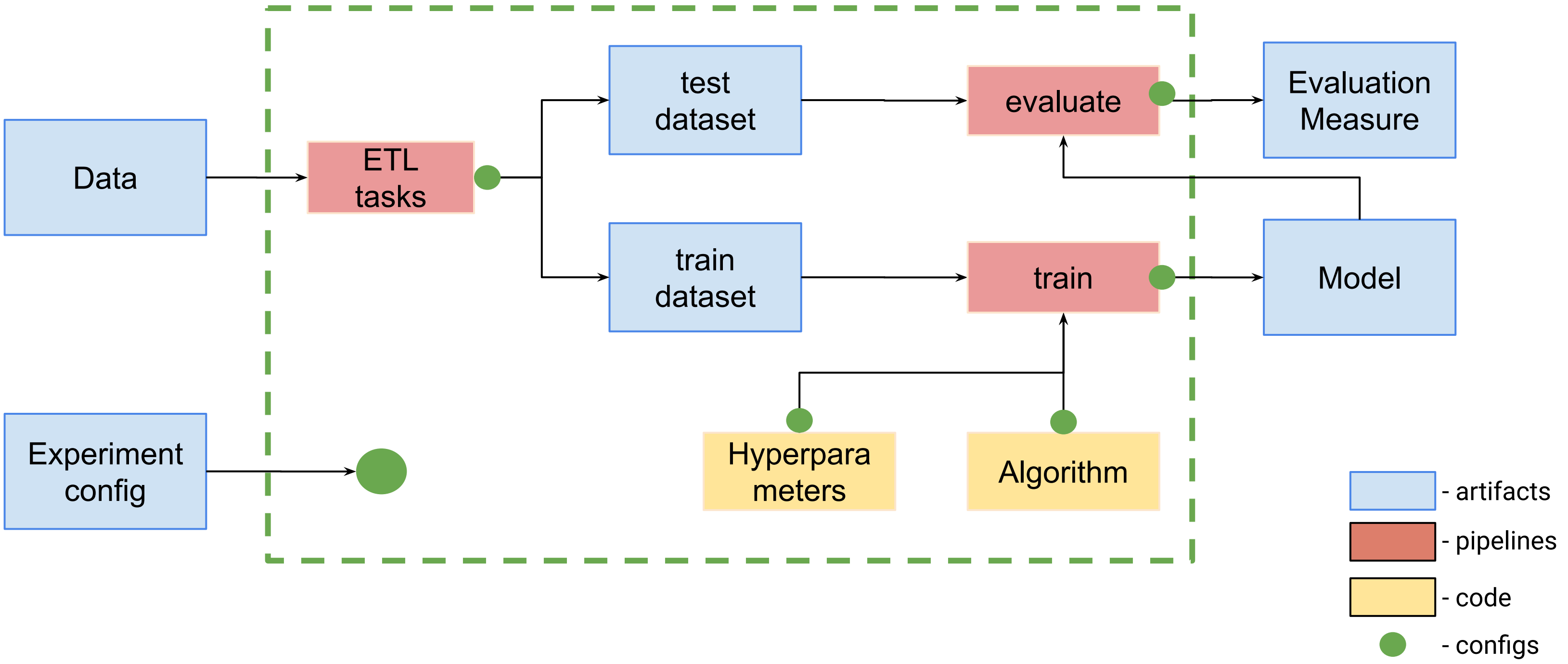


Image source: https://dvc.org/doc/tutorial

Experiment: pipelines, configs and artifacts



Use Case:

Iris Flowers Classification

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

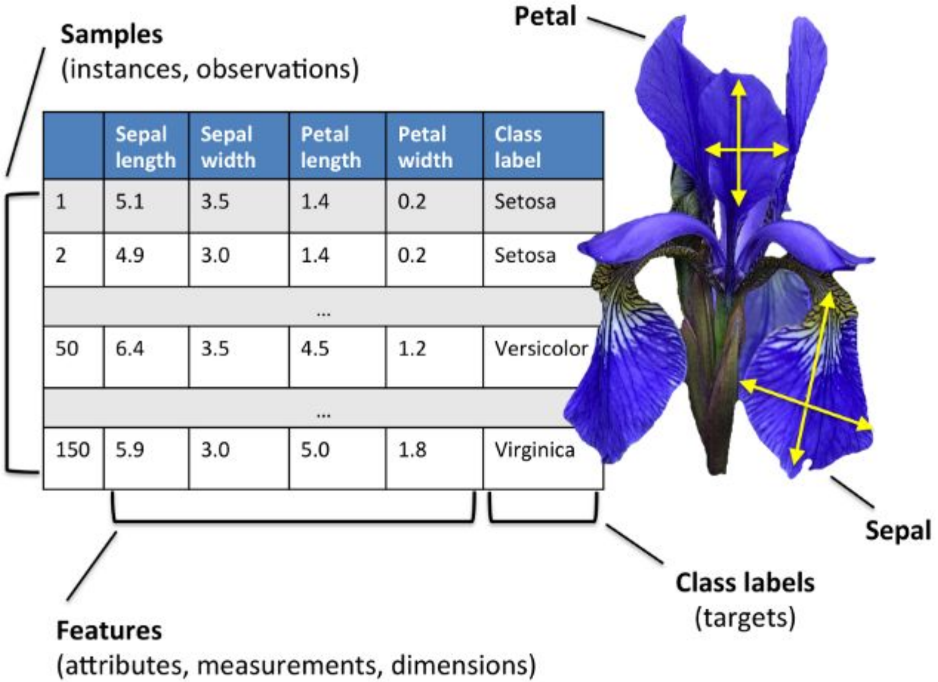
References:

- 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



	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

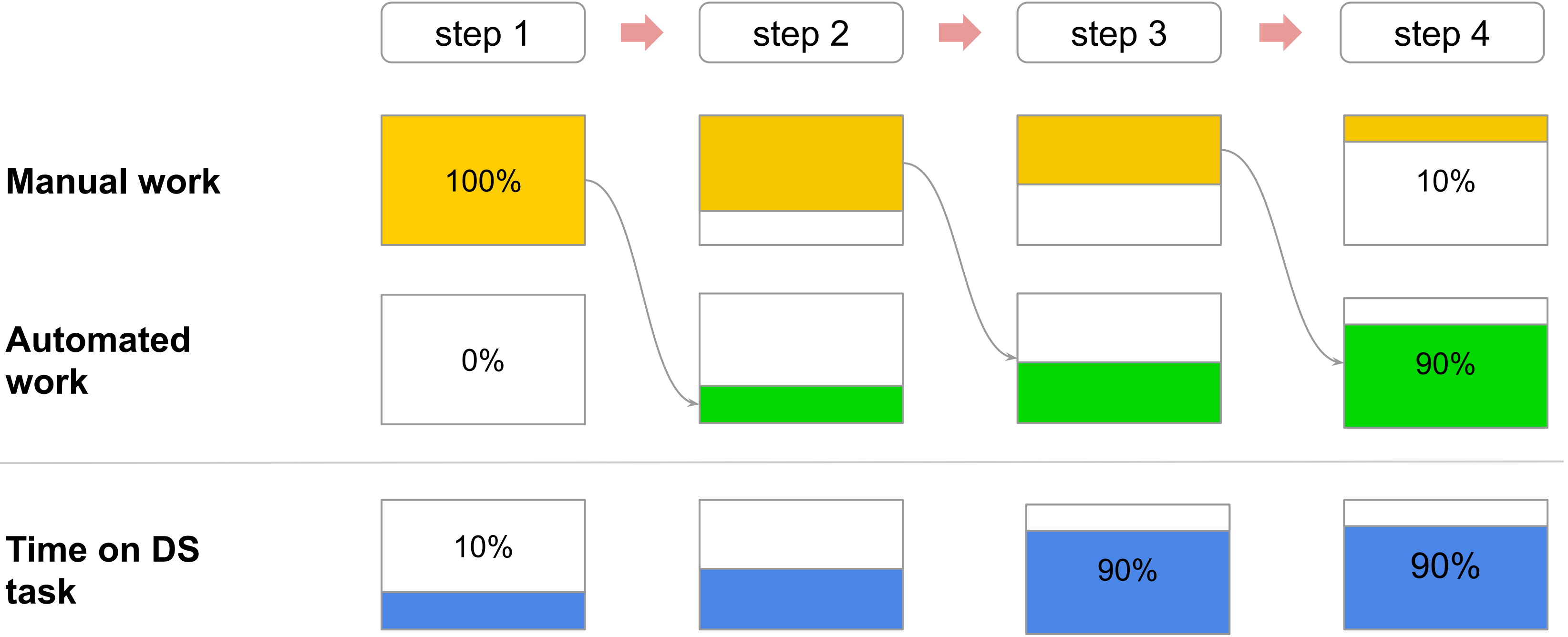
Image source:
https://medium.com/@jebaseelanravi96/machine-learning-iris-classification-33aa18a4a983

ML Reproducibility 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

How to start?

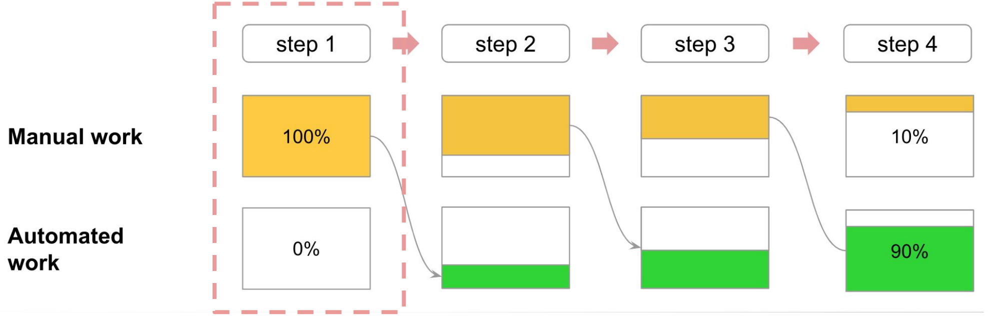

HELP!


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

Step 1:

Jupyter Notebook

- code in Jupyter Notebook
- everything in Docker

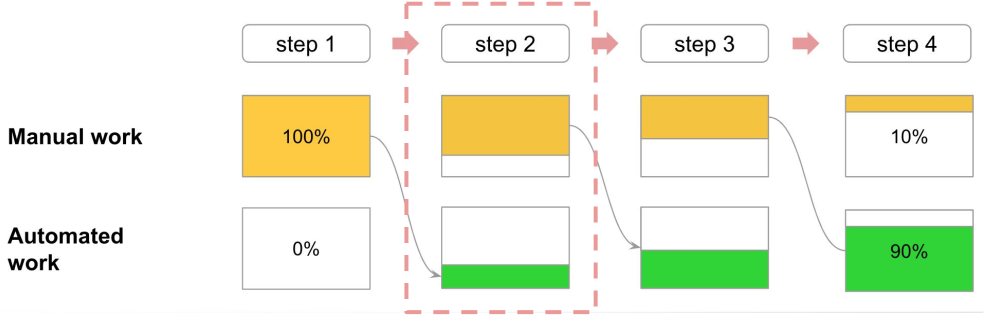


ML Reproducibility checklist

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

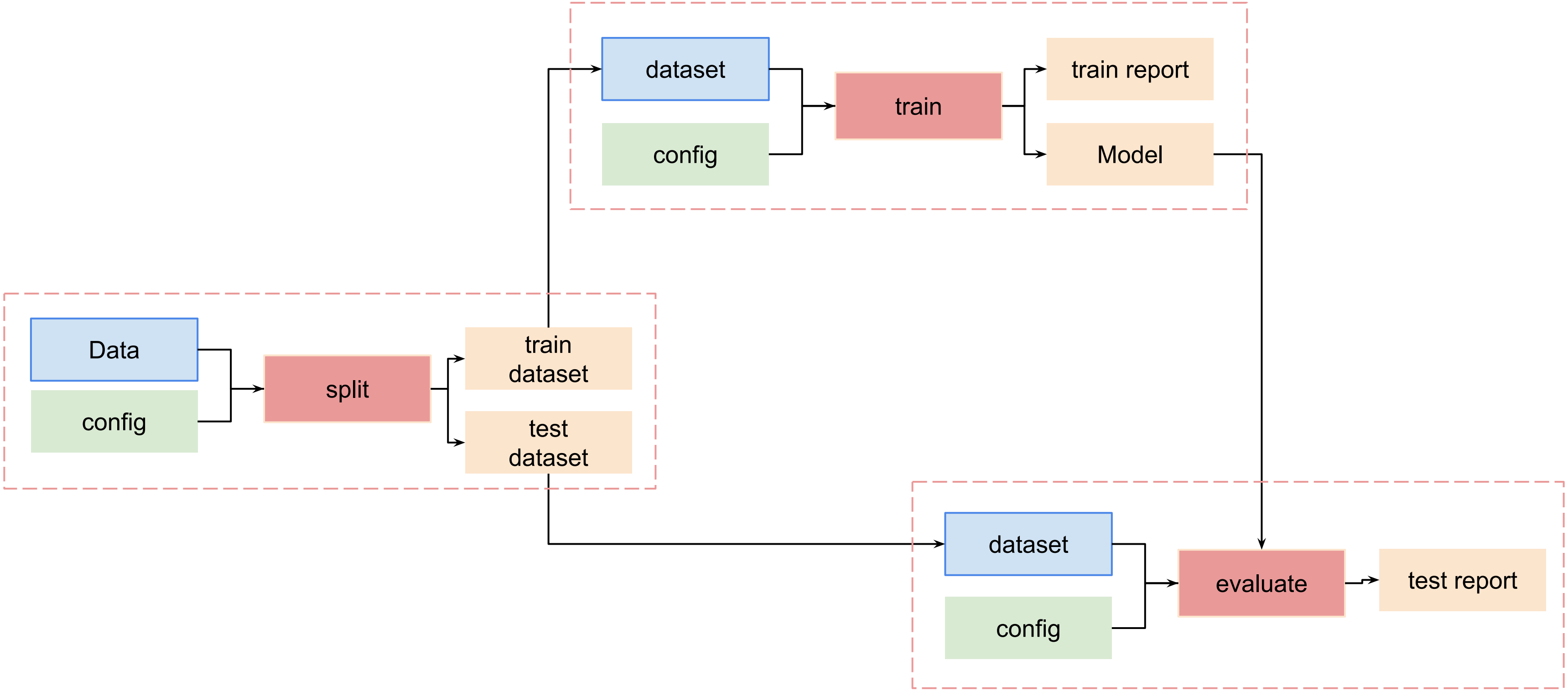


Project structure

```
.
├── 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



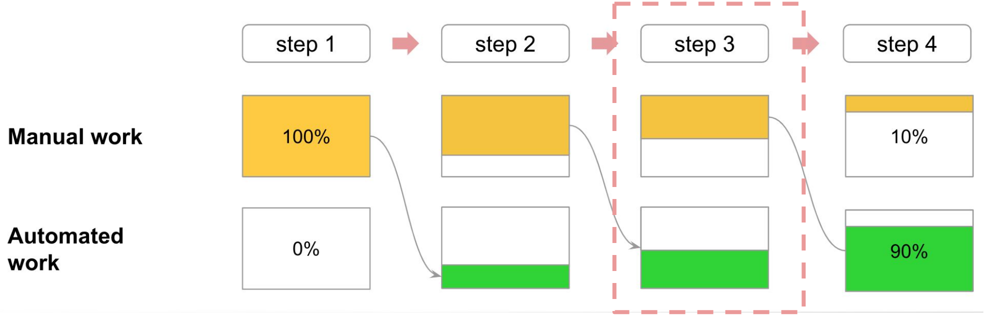
ML Reproducibility checklist

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:

Pipelines automation

- add pipelines dependencies under DVC control
- add models/data/configs 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



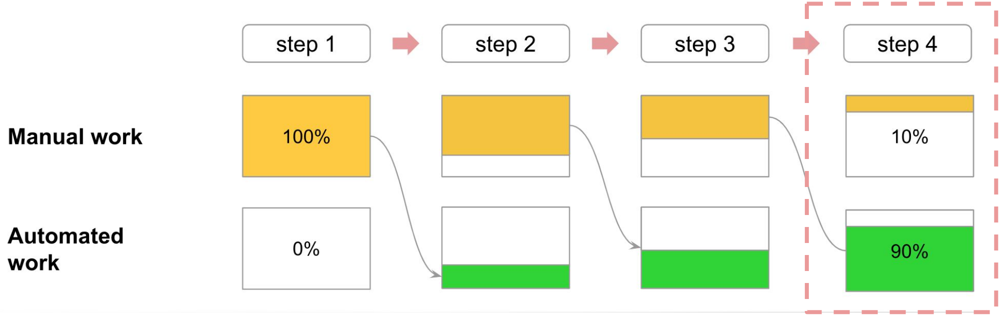
ML Reproducibility checklist

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:

Experiments management

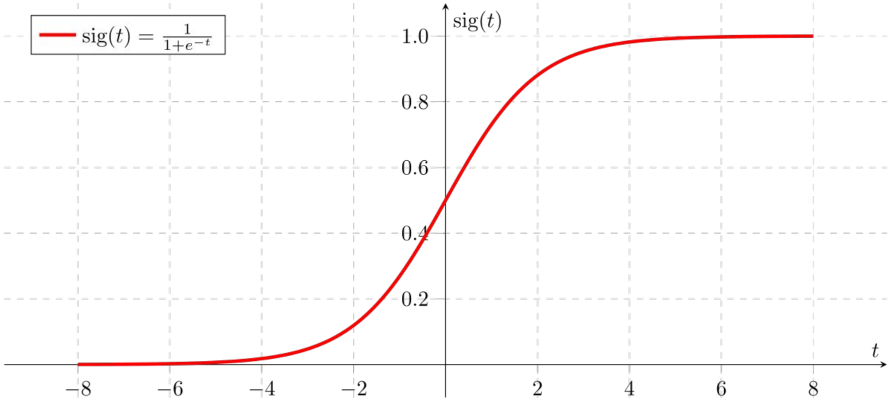
- add metrics tracking
- add experiments benchmark (DVC)
- pipelines dependencies under DVC control
- models/data/congis under DVC control
- same code in .py modules
- same pipelines
- everything in Docker
- run experiments in terminal or Jupyter Notebook



Experiment 1:

Tune Logistic Regression

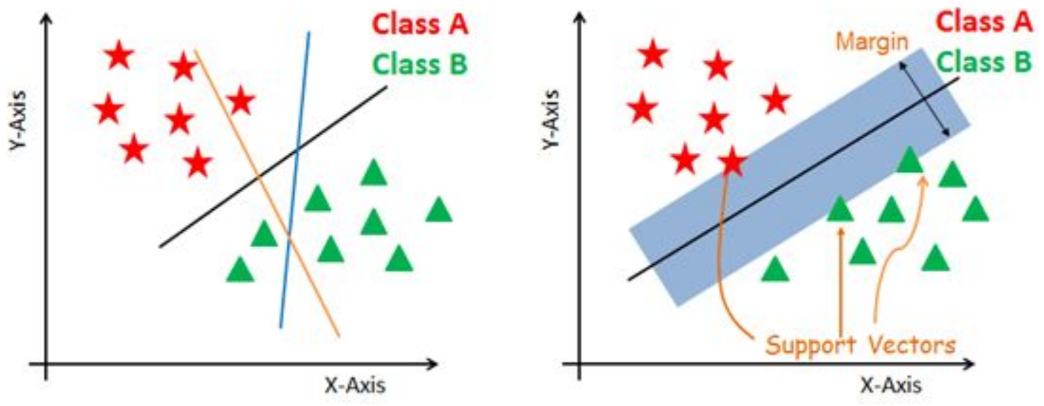
- hyperparameters tuning



Experiment 2:

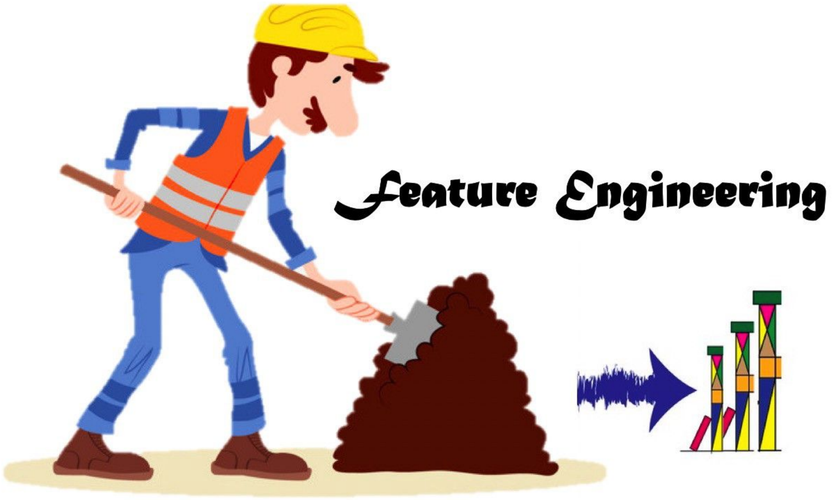
Use SVM

- use SVM estimator
- hyperparameters tuning



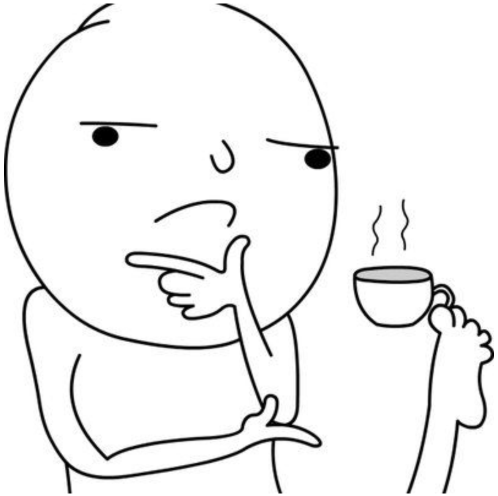
Experiment 3:
Add new features

- add squared features



Compare experiments

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



ML Reproducibility 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. pipelines - not difficult
2. start where you detect a “copy-paste” pattern
3. artifacts version control - MUST
4. discipline in a team is important
5. more benefits for high complexity and large team projects

Contact me

Mikhail Rozhkov

mail: mnrozhkov@gmail.com

ods: @Mikhail Rozhkov