Tools that help you get your experiments under control

Katharina Rasch

PyData Berlin 2019

Reproducibility in theory and in practice

Katharina Rasch

PyData Berlin 2019

Katharina Rasch
kat@krasch.io
https://github.com/krasch/presentations

PhD Computer Science
Previously: Data science / Computer Vision at zalando
Now: Freelance data science + teaching

https://xkcd.com/1838/

20190921T1114.
json

20190921T1114.
pkl

20190921T1803.
json

20190921T1803.
pkl

20190922T1117.
json

20190924T1815.
json

20190924T1815.
pkl

20190926T1733.
json

20190926T1733.
pkl

20190927T1043.
json

20190927T1043.
pkl

20190927T1344.
json

20190927T1344.
pkl

20190927T1346.
json

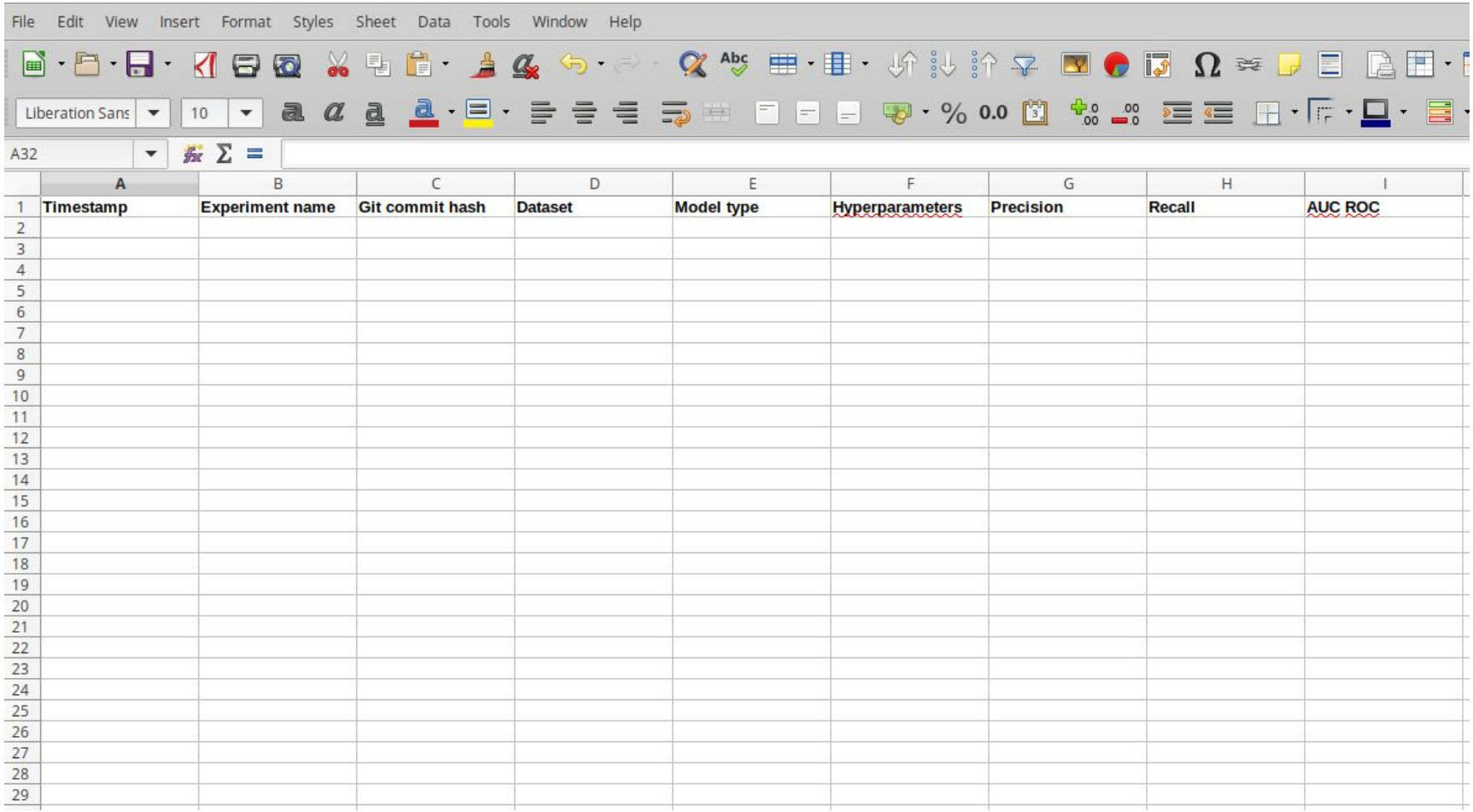
20190927T1346.
pkl

20190927T1411.
json

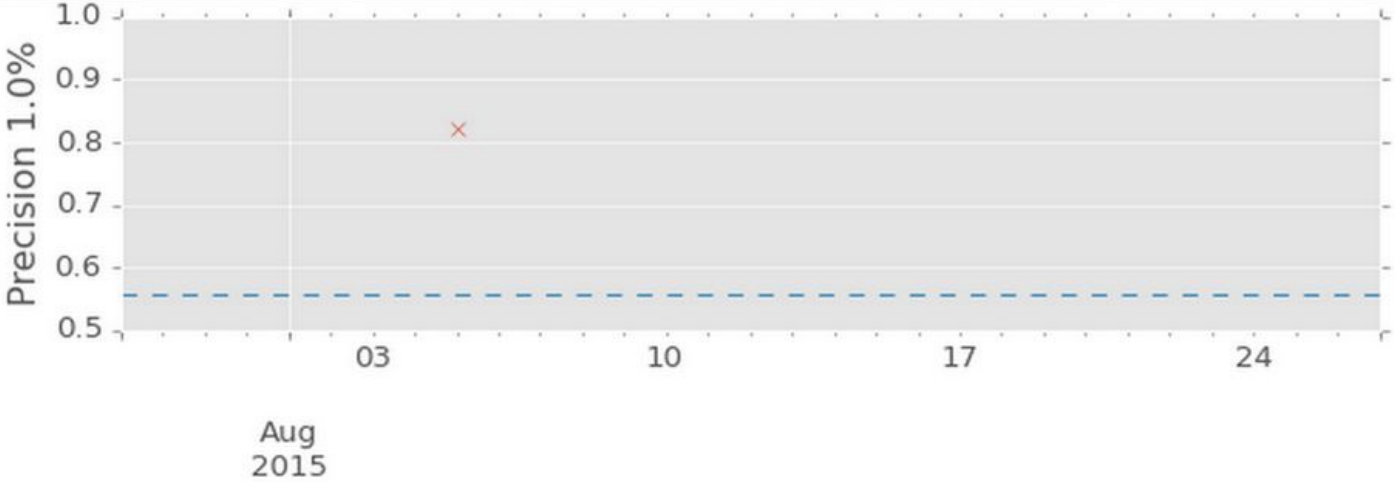
20190927T1411.
pkl

20190927T1607.
json

20190927T1607.
pkl



Timestamp	Experiment name	Git commit hash	Dataset	Model type	Hyperparameters	Precision	Recall	AUC ROC



Time	Features	Model	Precision 1.0%
05 Aug 2015			
12:49	all	RandomForest {'max_features': 'auto', 'n_estimators': 50, 'depth': 3, 'criterion': 'gini'}	0.82
12:52	all	RandomForest {'max_features': 'auto'	0.82

1. Motivation

Pride

Worry

Trust

Teamwork

Reproducibility

1. Motivation
2. Scope
3. Theory
4. Practice

IMHO

2. Scope

Model exploration / development

many models, features, parameters
(mostly) fixed dataset

errors expected to happen

flexibility needed

Model usage / deployment
(e.g. nightly training on new data)

fixed model type, features, parameters
frequent data updates

should be reliable, monitored

rigidity helpful

Model exploration / development

many models, features, parameters
(mostly) fixed dataset

errors expected to happen

flexibility needed

Model usage / deployment
(e.g. nightly training on new data)

fixed model type, features, parameters
frequent data updates

should be reliable, monitored

rigidity helpful

Icon from https://www.stockio.com

Model exploration / development

many models, features, parameters
(mostly) fixed dataset

errors expected to happen

flexibility needed

--> today

Model usage / deployment
(e.g. nightly training on new data)

fixed model type, features, parameters
frequent data updates

should be reliable, monitored

rigidity helpful

--> not today

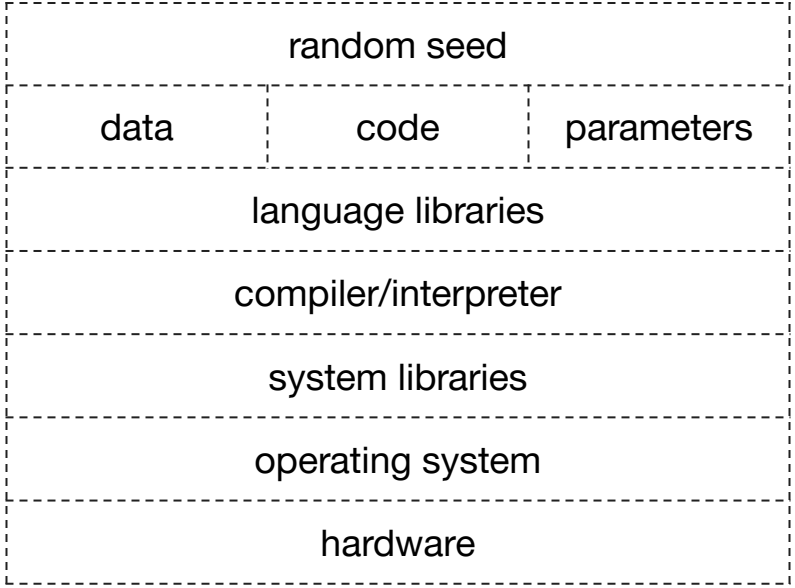
Reproducibility vs. Replicability: A Brief History of a Confused Terminology

HE Plesser, Frontiers in Neuroinformatics, 2018

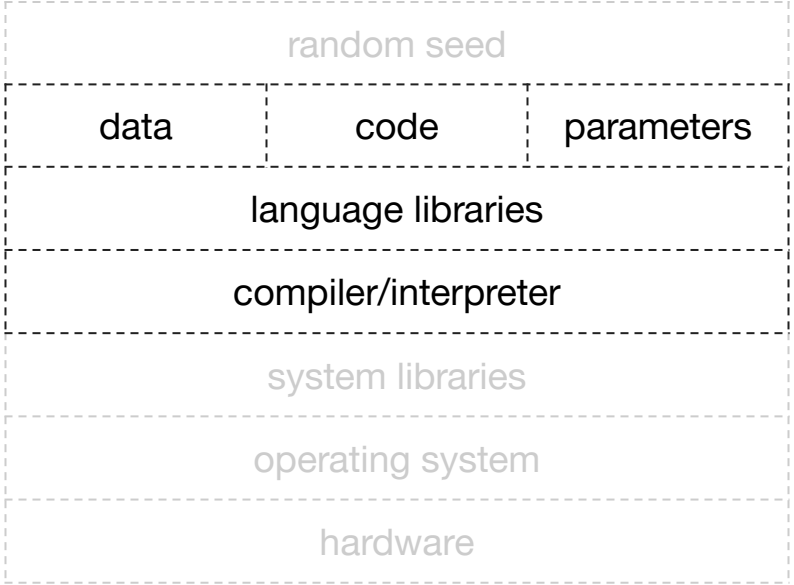
This talk:
reproducibility = ability to obtain the same* model with reasonable* efforts

3. Theory

Exact same model



Same* model


random seed
data
code
parameters
language libraries
compiler/interpreter
system libraries
operating system
hardware

Provenance (?)
(can compare models)

data hash / version	code hash / version	parameter names and values
language libraries name and version		
compiler/interpreter name and version		

Reproducibility
(can reproduce models)

actual data	actual code	parameter names and values
actual language libraries		
actual compiler/interpreter		

Meta	Provenance					Metrics	
Time-stamp	Data version	Code version	Model type	Over-sample?	Hyper-params	Precision	Recall

Meta		Provenance					Metrics	
Time-stamp	Name	Data version	Code version	Model type	Over-sample?	Hyper-params	Precision	Recall
	test							
	test							
	baseline							
	baseline_ fixed							

Meta	Provenance					Metrics	
Time-stamp	Data version	Code version	Model type	Over-sample?	Hyper-params	Precision	Recall

Parameters

Time-stamp	Data version	Code version	Model type	Over-sample?	Hyper-params	Precision	Recall

Project-dependent

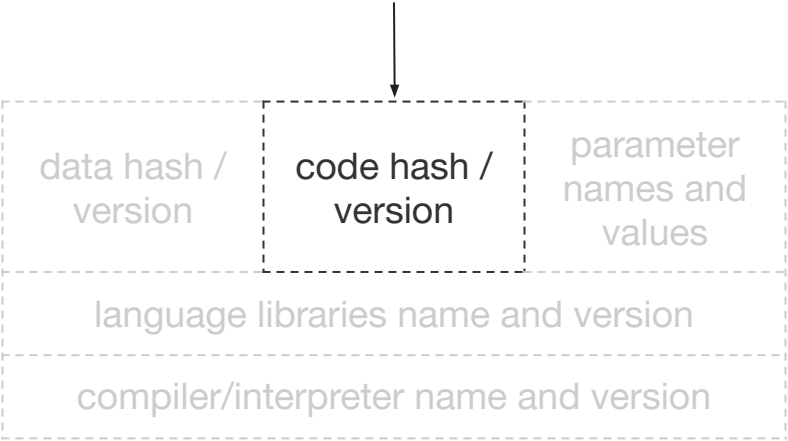
Time-stamp	Data version	Code version	Model type	Over-sample?	Hyper-params	Precision	Recall

Provenance			Reproducibility		
data hash / version	code hash / version	parameter names and values	actual data	actual code	parameter names and values
language libraries name and version			actual language libraries		
compiler/interpreter name and ver			actual compiler/interpreter		

Out-of-sync error

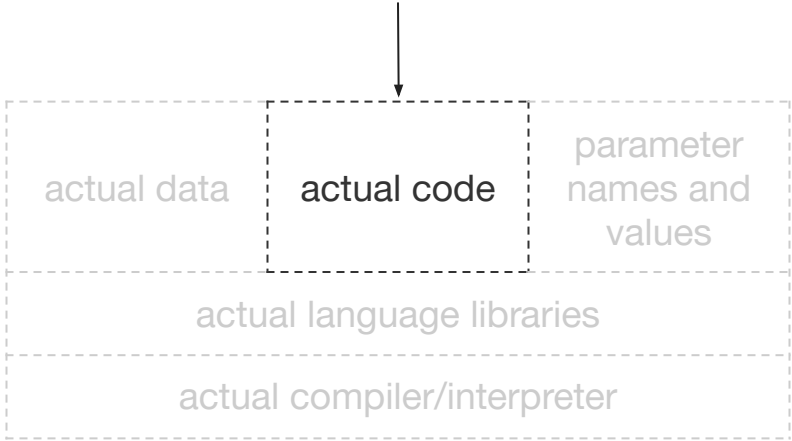
Provenance

Git commit hash



Reproducibility

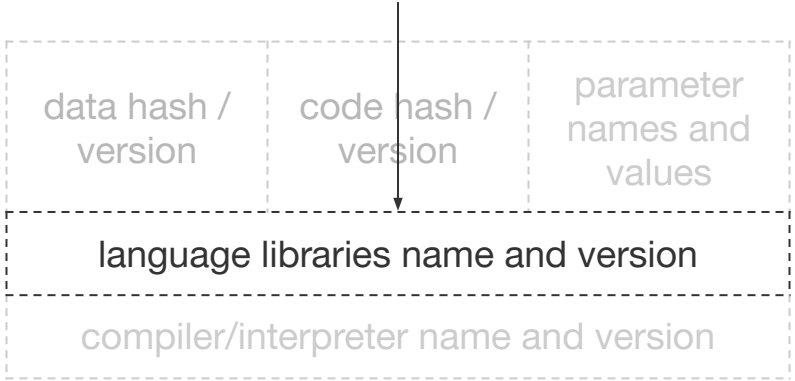
Developer's local source code directory



Forgot to commit before starting experiment

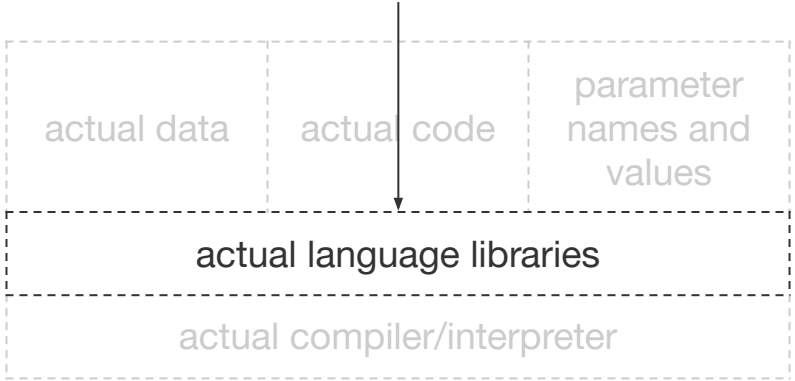
Provenance

requirements.txt
conda.yaml
Dockerfile



Reproducibility

Developer's local environment



Forgot to update requirements file after installing new library

Provenance

Hash calculated over data file(s)

data hash / version	code hash / version	parameter names and values
language libraries name and version		
compiler/interpreter name and version		

Reproducibility

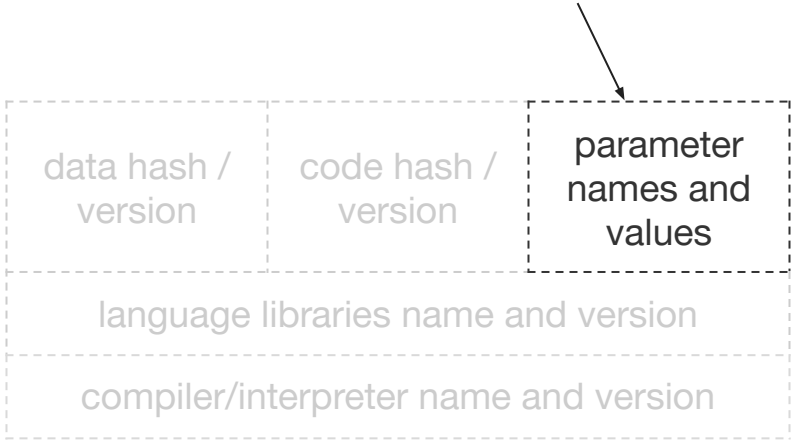
File on developer machine, S3, etc

actual data	actual code	parameter names and values
actual language libraries		
actual compiler/interpreter		

Altered data without calculating new hash

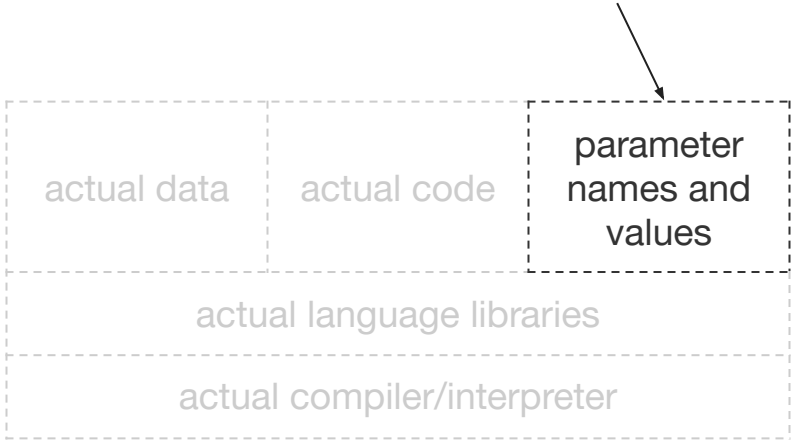
Provenance

command line parameters, config file, config in code



Reproducibility

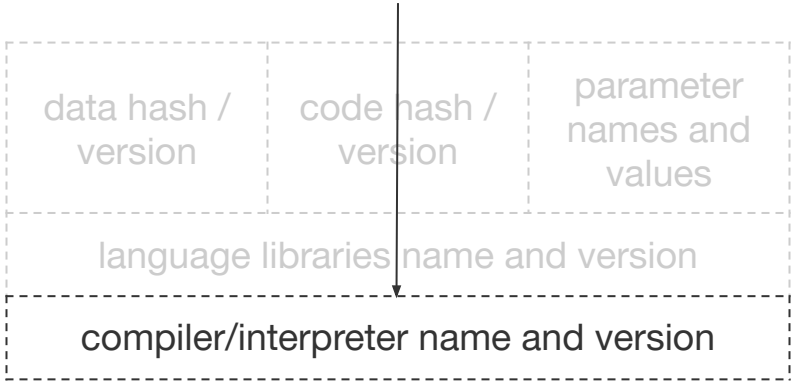
command line parameters, config file, config in code



Not actually passing parameters to model -> code review, tests

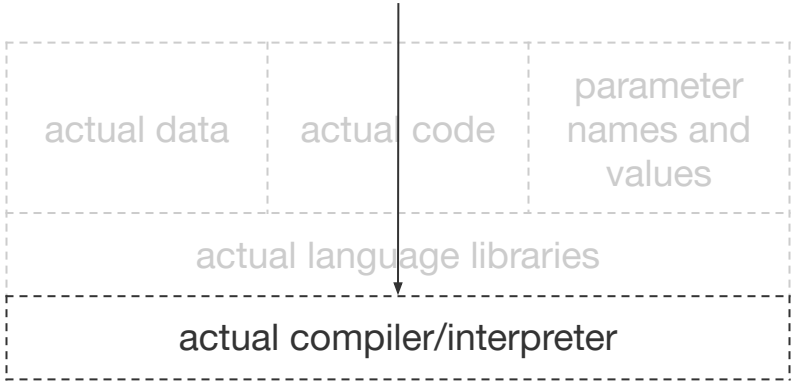
Provenance

Readme.md
Dockerfile



Reproducibility

Developer's local python installation



Using wrong python version locally

For every experiment

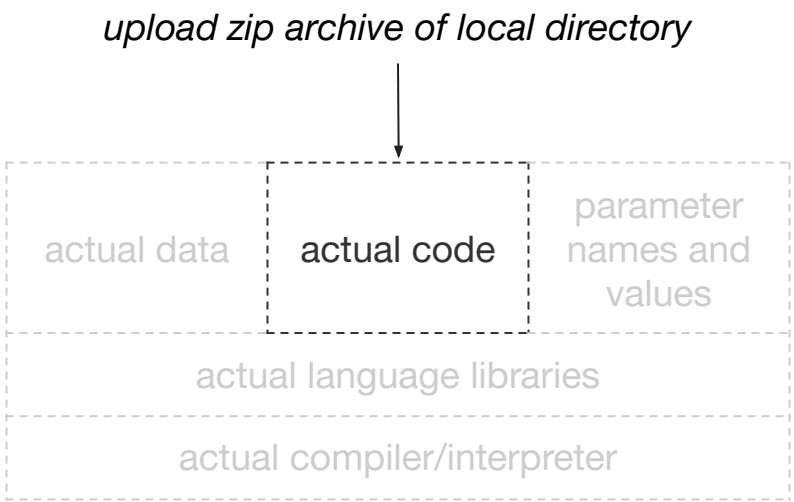
ensure using correct data
ensure local git repo clean
ensure requirements up-to-date
ensure using correct python version

Too much responsibility!

Too much hassle!

Provenance

Reproducibility



Duplicating git functionality
Less easy to compare

Development mode: best-effort reproducibility

Code	Where executed	Who triggers
-/zip	local	user
git	local	user

Reproducibility mode: permanently recording successful experiments

Code	Where executed	Who triggers
-/zip	local	user
git	local	user

☐	#21 Train model	9 minutes ago	discokugel	46.2254	43.8166	Complete	39 seconds	Baseline	lag_one_week
☐	#18 Train model	38 minutes ago	discokugel			Error	52 seconds	Baseline	lag_one_week
☐	#17 Train model	38 minutes ago	discokugel			Error	5 seconds	LinearRegression	all_lag_date
☐	#16 Train model	42 minutes ago	discokugel	37.6169	36.8913	Complete	6 seconds	LinearRegression	lag_features

Screenshot from valohai.com

Code	Where executed	Who triggers
-/zip	local	user
git	local	user
git	remote	user
git	remote	git push / merge request

Open krasch wants to merge 2 commits into master from random_forest

Conversation 0 | Commits 2 | Checks 0 | Files changed 3 | +33 −10

krasch commented 40 seconds ago

No description provided.

krasch added 2 commits 13 minutes ago

Added random forest be87b40

Plot predictiosn 3cd8dc8

Add more commits by pushing to the **random_forest** branch on **krasch/gradient-bikes**.

All checks have failed Hide all checks

1 errored check

gradientci Details

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request or view command line instructions.

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request Comment

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications Customize

Unsubscribe

You're receiving notifications because you're watching this repository.

1 participant

Lock conversation

Screenshot from github.com, using https://gradient.paperspace.com/ CI

How do you want to work?

4. Practice

Software as a Service solutions

DEEPKIT

SPELL

cnvrg.io

gradient

FLOYDHUB

LOSSWISE

Weights & Biases

VALOHAI

hydrosphere.io

comet

neptune.ml

Open source tools

MODELCHIMP

mlflow

DEPLOY-ML

Pachyderm

Sumatra

Sacred

FGLab

Polyaxon

VisTrails

DVC

StreamSets

noWorkflow

Open source tools

MODELCHIMP

mlflow

Pachyderm

Sacred

Polyaxon

DVC

StreamSets

Open source tools

[apache license]

Sacred

[MIT license]

[apache license]

Sacred

```python
from sacred import Experiment
ex = Experiment("bikes")

ex.add_resource("data/hourly_counts.csv")

ex.log_scalar("train.rmse", train_rmse)
ex.add_artifact("results/predictions.png")
```

```python
@ex.named_config
def baseline():
    split = "2018-01-01"
    feature_functions = [features.lag_one_week]

@ex.automain
def run_experiment(split, feature_functions):
    # do data science stuff
```

```
> python train.py with baseline
```

Omniboard v2.3.0 (admin)

Status: 7 selected

Filters: Column Name | == | Enter Value... | Add Filter

Compare | +/- Metric Columns | Show/Hide Columns

24 experiments

Auto Refresh | Last Update: September 26th, 09:24:59 PM | Reload

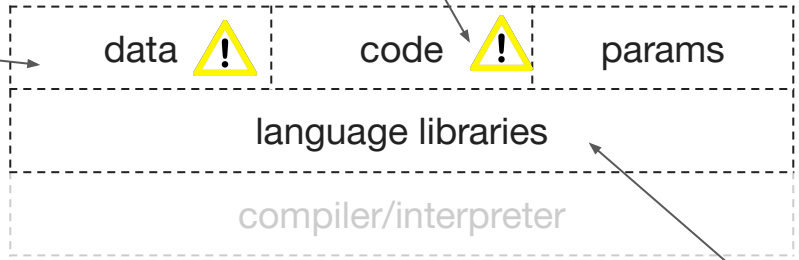
Id ↑	Experiment Name	Start Time	Status	Notes	Resources	Duration	Result	Stop Time	Seed	RMSE Train	RMSE Test
24	bikes	2019-09-26T16:44:44	FAILED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv" ["/home/arbeit/embrace-(	356ms		2019-09-26T16:44:44	965283310		
23	bikes	2019-09-26T16:44:14	FAILED	Enter Notes	[]	18ms		2019-09-26T16:44:14	549376400		
21	bikes	2019-09-22T22:27:44	COMPLETED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv"	53.9s		2019-09-22T22:28:38	152215597	107.66954716456115	107.66954716456115
20	bikes	2019-09-22T22:11:34	COMPLETED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv" ["/home/arbeit/embrace-(	1.7s		2019-09-22T22:11:36	403214227	34.45518373155321	34.45518373155321
19	bikes	2019-09-22T22:11:21	COMPLETED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv" ["/home/arbeit/embrace-(	1.7s		2019-09-22T22:11:23	9631558	34.24920057236309	34.24920057236309
18	bikes	2019-09-22T22:11:07	FAILED	Enter Notes	[]	12ms		2019-09-22T22:11:07	209327342		
17	bikes	2019-09-22T22:10:52	FAILED	Enter Notes	[]	15ms		2019-09-22T22:10:52	66310965		
16	bikes	2019-09-22T21:13:56	COMPLETED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv" ["/home/arbeit/embrace-(	1.6s		2019-09-22T21:13:58	136782595	34.42437733721705	34.42437733721705
15	bikes	2019-09-22T21:13:41	COMPLETED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv" ["/home/arbeit/embrace-(			2019-09-22T21:13:44	178673814		
14	bikes	2019-09-22T21:05:27	COMPLETED	Enter Notes	["/home/arbeit/embrace-/data/hourly_counts.csv" ["/home/arbeit/embrace-(	1.6s		2019-09-22T21:05:28	241019292	34.31922926598029	34.31922926598029

Code	Where executed	Who triggers
zip	local	user
git (with local changes, flagged as “dirty” in UI)	local	user
git (no local changes)	local	user
git	remote	user
git	remote	git push / merge request

always: zip uploaded

optional: reproducibility mode, must remember to use `--enforce_clean`

md5 hash
also stores data??
how to get data out ??



auto-detects
used libraries

Steps to actually reproduce

1. Obtain the data (somehow)
2. Obtain the code
   a. download zip from UI
   b. git checkout (if clean commit)
3. Find list of libraries in UI and install
4. Find config values in UI
5. `> python train.py with param1=value1 ...`

mlflow

```python
with mlflow.start_run():
    mlflow.log_param("model", args.model)
    mlflow.log_param("features", args.features)

    # do data science stuff
    ...

    mlflow.log_metric("RMSE_train", rmse_train)
    mlflow.log_artifact("results/predictions.png")
```

conda.yaml

```yaml
name: bikes
channels:
 - defaults
dependencies:
 - scikit-learn=0.21.1
 - matplotlib
 - keras
 - pip:
   - mlflow
```

MLProject

```
name: bikes

conda_env: conda.yaml
entry_points:

  hyper_train:
    parameters:
      max_depth: int
      n_estimators: int
      bootstrap: boolean
    command: "python -m
hyper.train ..."
```

```
> mlflow run -e hyper_train .
```

```
> mlflow run git@gitlab.com:org/mlflow_bikes.git
```

mlflow

GitHub Docs

Experiments

Default

Default

Experiment ID: 0 Artifact Location: ./mlruns/0

Description:

Search Runs: metrics.rmse < 1 and params.model = "tree" State: Active Search

Filter Params: alpha, lr Filter Metrics: rmse, r2 Clear

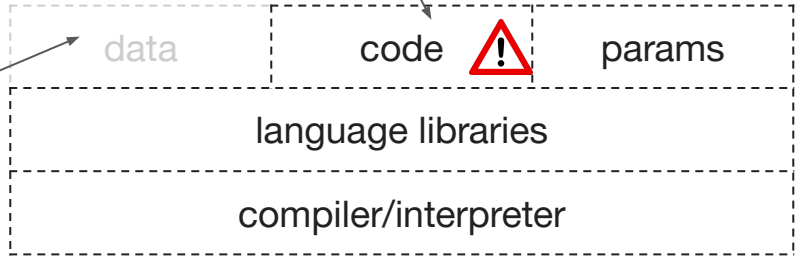
Showing 36 matching runs Compare Delete Download CSV

Date	User	Run Name	Source	Version	Tags	model	Parameters	Metrics
2019-09-20 11:06:03	arbeit		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	RMSE_test: 35.6768504256… RMSE_train: 36.5240951492…
2019-09-20 11:05:43	kat		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	
2019-09-20 11:05:17	kat		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	
2019-09-20 11:02:05	kat		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	
2019-09-20 11:01:43	kat		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	RMSE_test: 35.6768504256… RMSE_train: 36.5240951492…
2019-09-20 10:59:15	kat		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	RMSE_test: 35.6768504256… RMSE_train: 36.5240951492…
2019-09-20 10:59:01	kat		bikes	eaac67		LinearRegression	features: all_lag_date_w… log: true	
2019-09-20 10:56:54	kat		mlflow …	eaac67		LinearRegression	features: all_lag_date_w… log: true	RMSE_test: 35.6768504256… RMSE_train: 36.5240951492…
2019-09-20 10:55:46	arbeit		mlflow …	eaac67		LinearRegression	features: all_lag_date_w… log: true	RMSE_test: 35.6768504256… RMSE_train: 36.5240951492…
2019-09-20 10:53:39	arbeit		mlflow …	eaac67		LinearRegression	features: all_lag_date_w… log: true	RMSE_test: 35.6768504256… RMSE_train: 36.5240951492…

Code	Where executed	Who triggers
-	local	user
git (with local changes, NOT flagged as dirty in UI)	local	user
git (no local changes)	local	user
git	remote	user
git	remote	git push / merge request

```
> mlflow run .
```

does not check if local git repo clean, records stale hash



not mentioned in documentation

Steps to actually reproduce

(if have been careful with `mlflow run .`)

1. Obtain the data (somehow)
2. `> mlflow run -v commit_hash git@repo...`

// add data and push to remote storage (e.g. S3)

```
> dvc add hourly_counts.csv
> dvc push
```

// set up pipeline

```
> dvc run -d hourly_counts.csv \
          -d train.py \
          -M results/rmse.json \
          python train.py
```

... change data or train.py

// re-run affected pipeline steps

```
> dvc repro
```

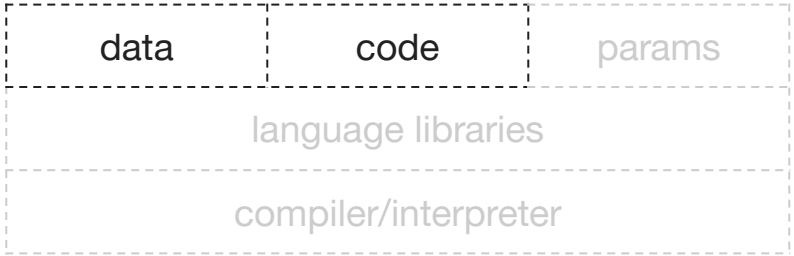
git tags

```
> dvc metrics show -T
```

```
baseline:
        results/rmse.json: {"RMSE (train)": 46.22540681588555, "RMSE (test)": 43.81664782510664}
linear_regression:
        results/rmse.json: {"RMSE (train)": 37.616881728555924, "RMSE (test)": 36.89125150760678}
linear_regression_with_lag_date_weather:
        results/rmse.json: {"RMSE (train)": 36.52409514921099, "RMSE (test)": 35.67685042569101}
```

Code	Where executed	Who triggers
-/zip	local	user
git (with local changes)	local	user
git (no local changes)	local	user
git	remote	user
git	remote	git push / merge request

⚠ Need to ensure pipeline in-sync with reality



⚠ Need to ensure `dvc repro` is run before committing (commit hook?)

Steps to actually reproduce

1. `> git clone ...`
2. `> dvc pull`
3. `> dvc repro -f`

How do you want to work?

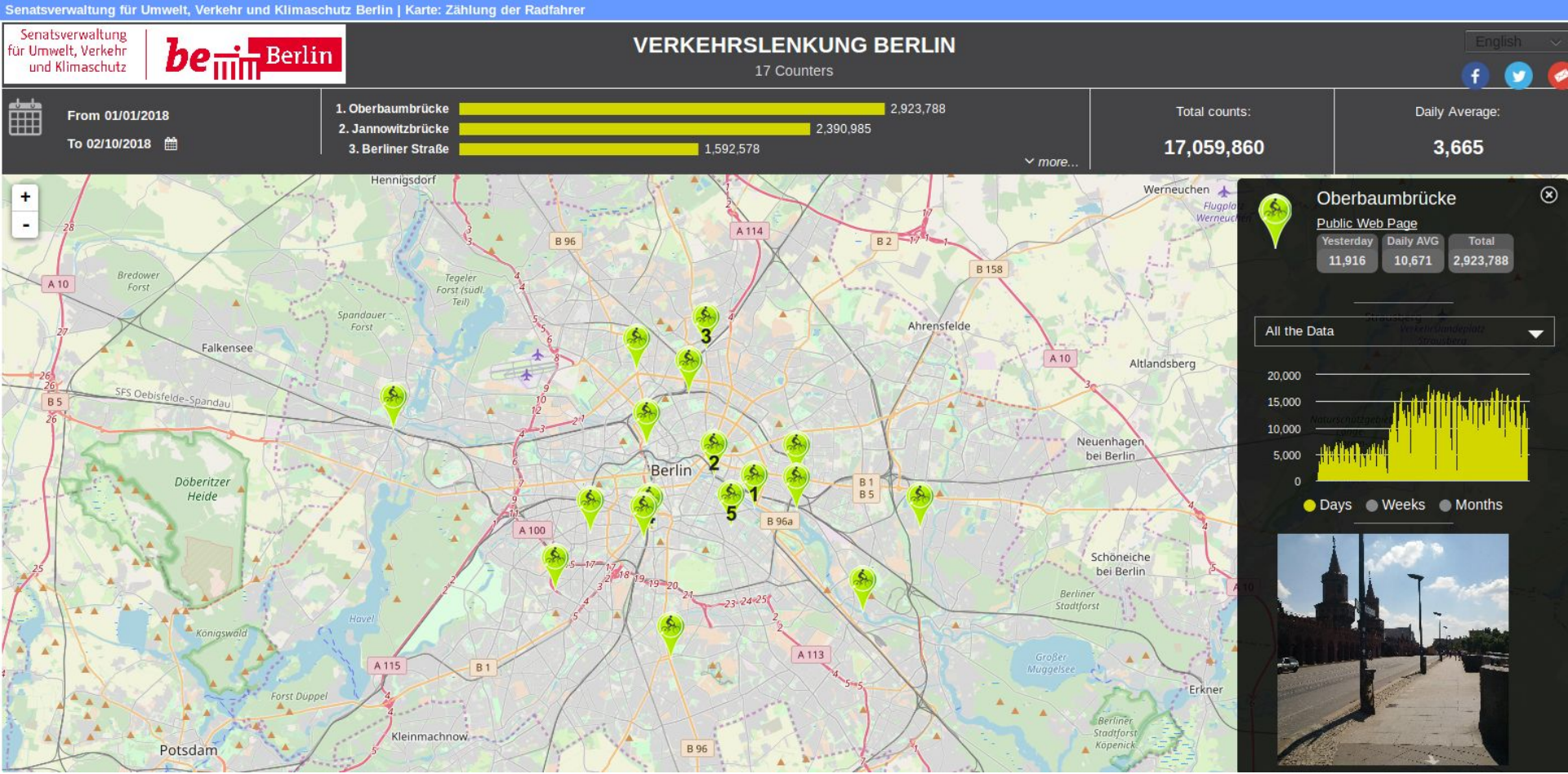
Envy

~~Envy~~ Inspiration

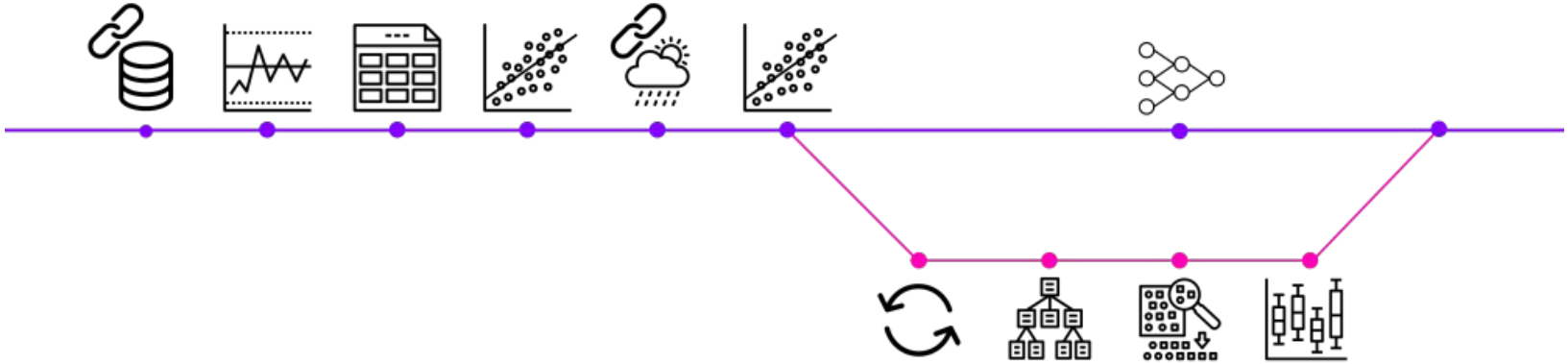
How do we want to work?

How do we want to work?

https://github.com/krasch/presentations
kat@krasch.io



https://www.berlin.de/senuvk/verkehr/lenkung/vlb/de/karte.shtml



Icons by Vectorstall, Vaibhav Radhakrishnan, Kimmi Studio, Becris; https://thenounproject.com