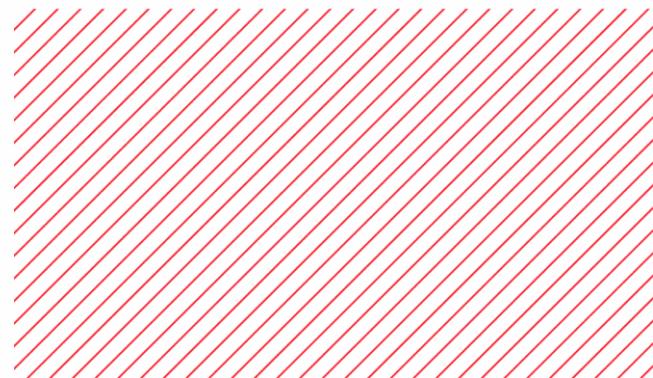


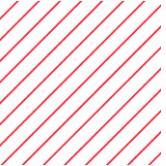
академия
больших
данных



Воспроизводимость в ML

Михаил Марюфич, MLE



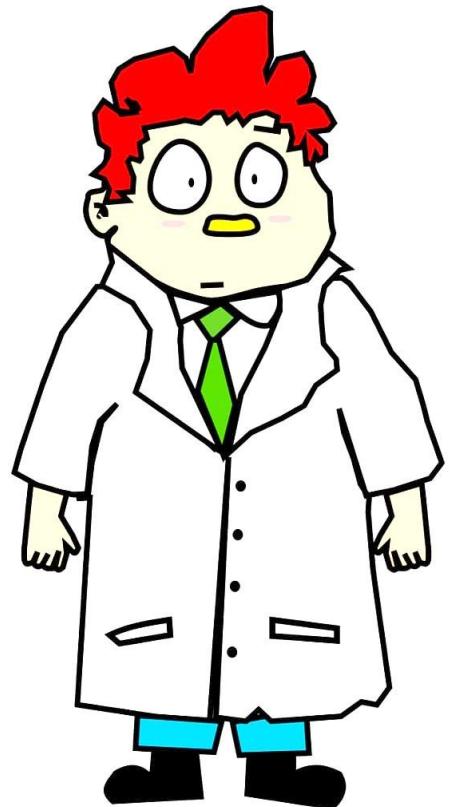


Содержание занятия

- Зачем все это надо?
- Как организовать воспроизводимое обучение
 - Почему не стоит хранить большие файлы в git
 - Где же их хранить? (рассмотрим s3)
- dvc
- mlflow

История одного Пети

Вася



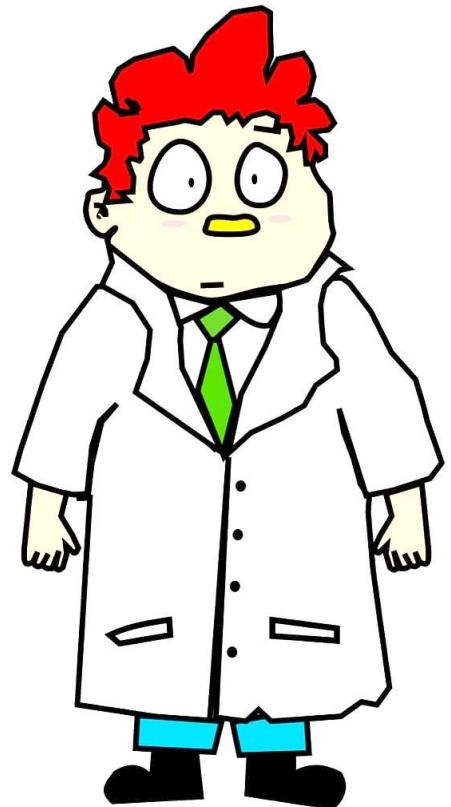
Data Scientist

Федя



Разработчик

Вася



Data Scientist

Федя

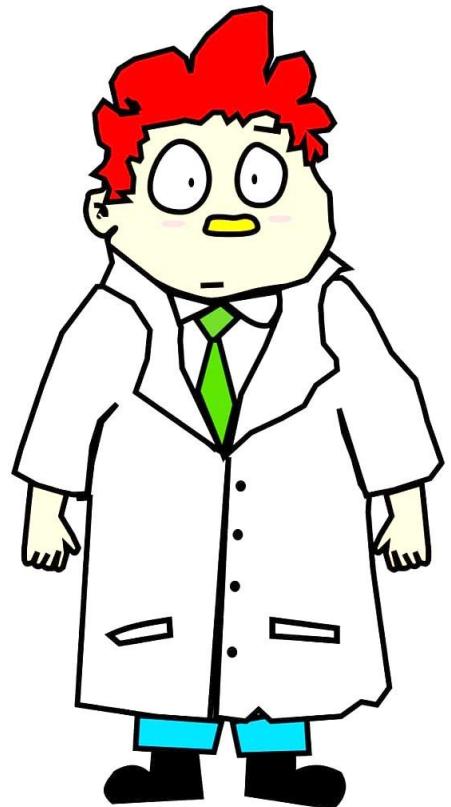


Разработчик



Го в прод?

Вася



Data Scientist

Федя



Разработчик

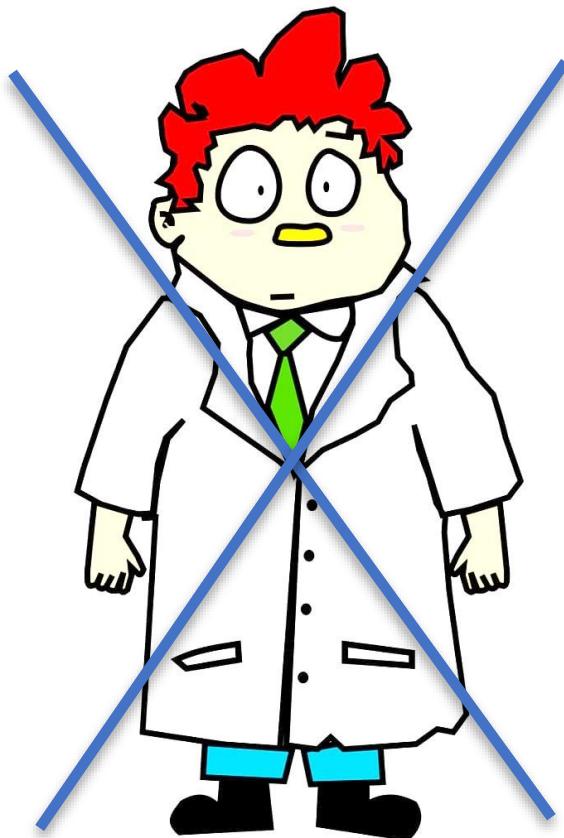


+30%

К ОЧЕНЬ ВАЖНОЙ
МЕТРИКЕ

Спустя год

Вася



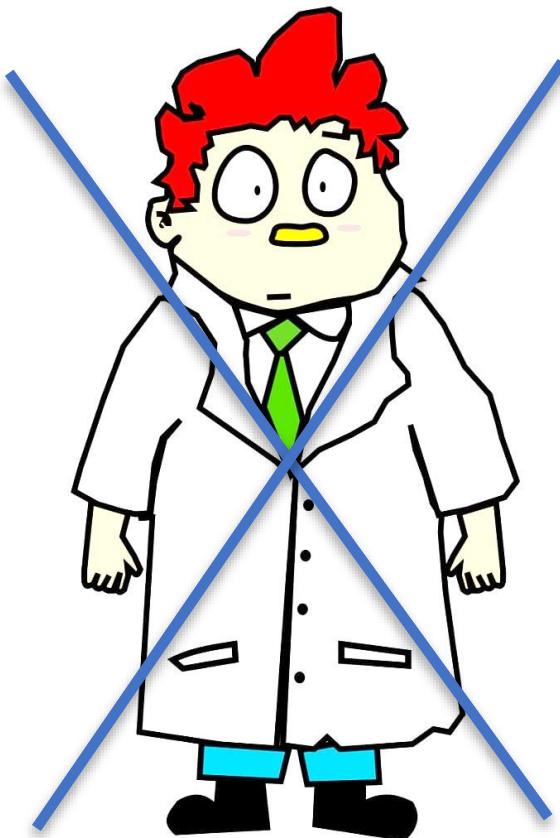
Data Scientist

Федя



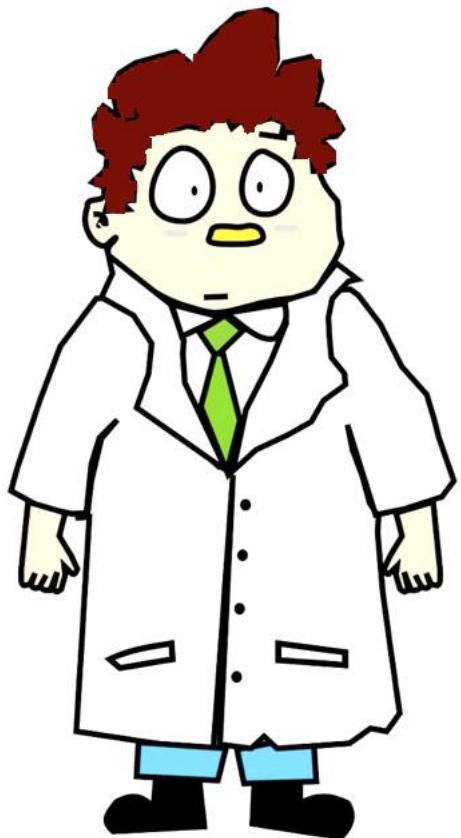
Разработчик

Вася



Data Scientist

Петя



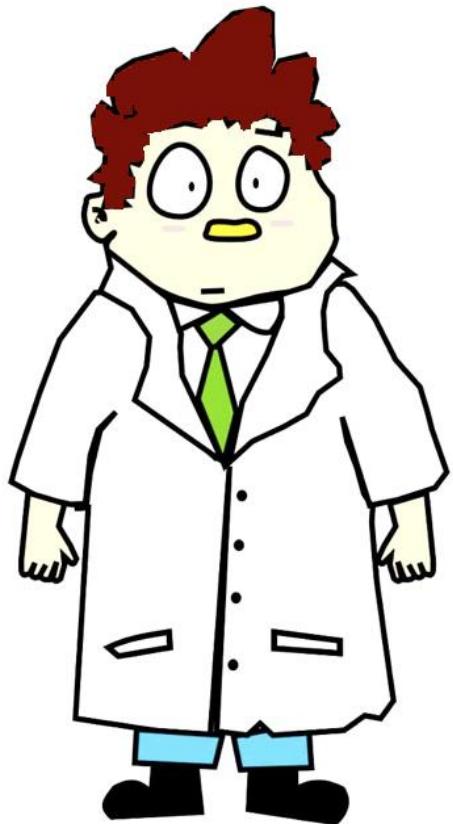
Новый Data Scientist

Федя



Разработчик

Петя



Data Scientist

Федя

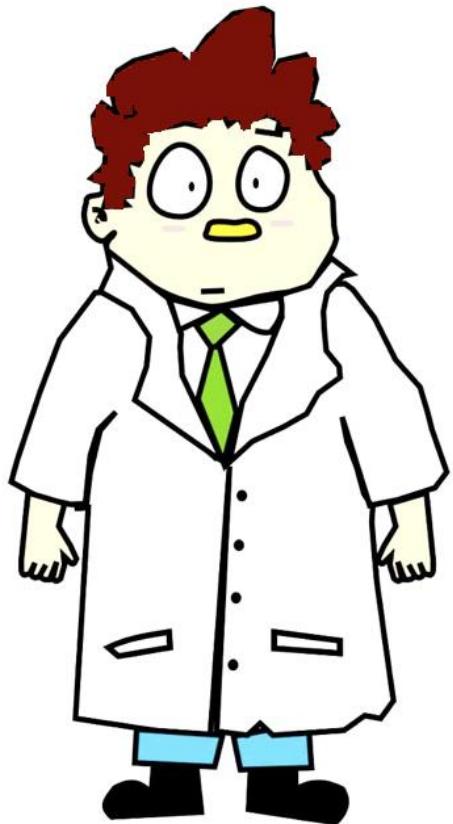


Разработчик



А что там в
продаже?

Петя



Data Scientist

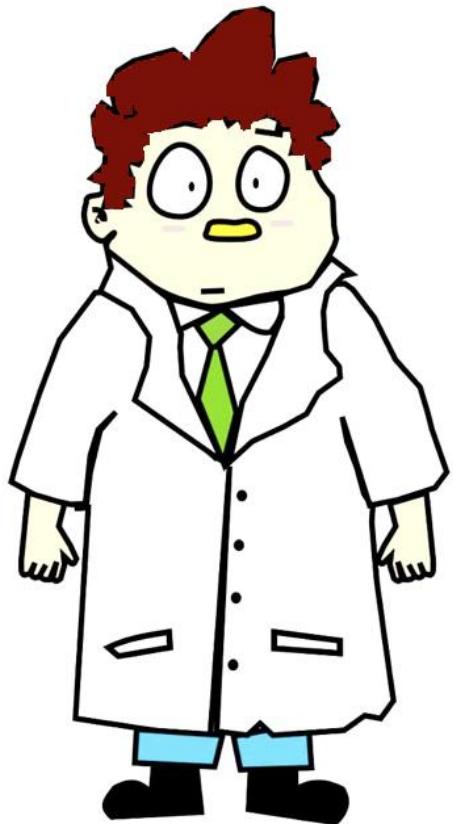
Федя



Разработчик



Петя

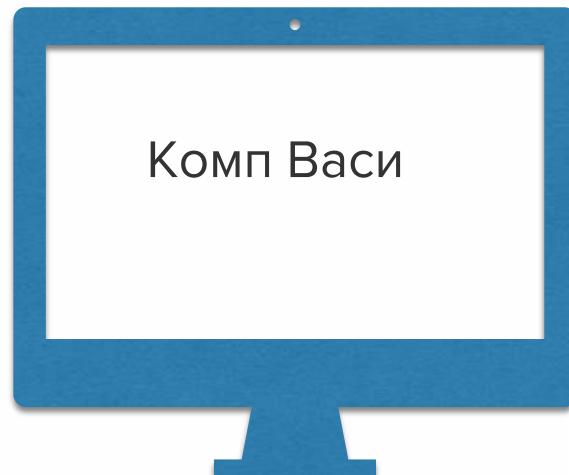


Data Scientist

Федя



Разработчик



Комп Васи

+35%

К ОЧЕНЬ ВАЖНОЙ
МЕТРИКЕ

+35%

К ОЧЕНЬ ВАЖНОЙ
МЕТРИКЕ

Но делать пришлось с нуля

Какие проблемы МОЖНО выделить?

Взгляд Васи на процесс

3 этапа жизни модели:

1) обучение

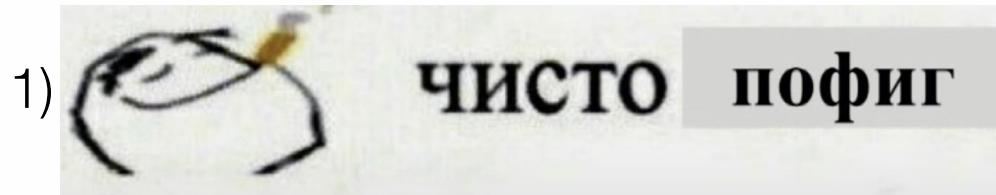
2)



чисто пофиг

3) переобучение

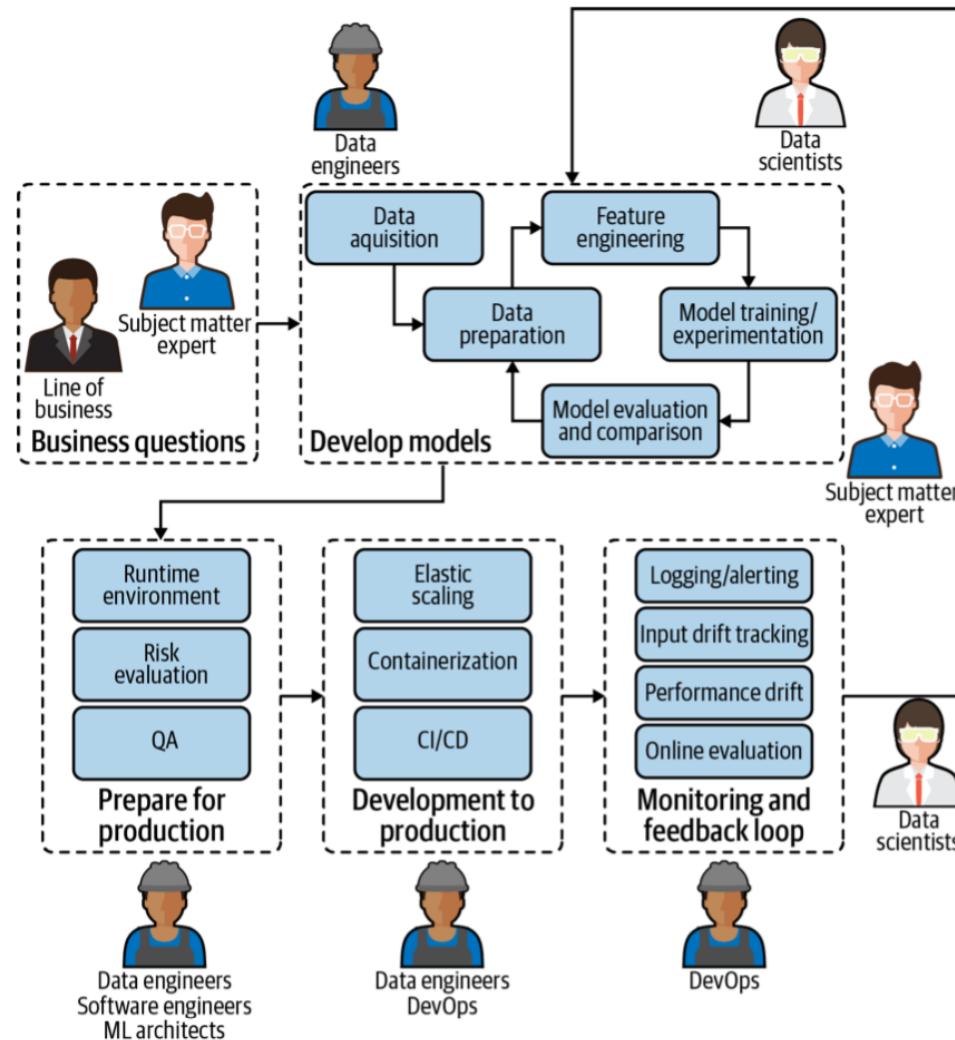
Взгляд Феди на процесс



1) Выкатка и эксплуатация модели



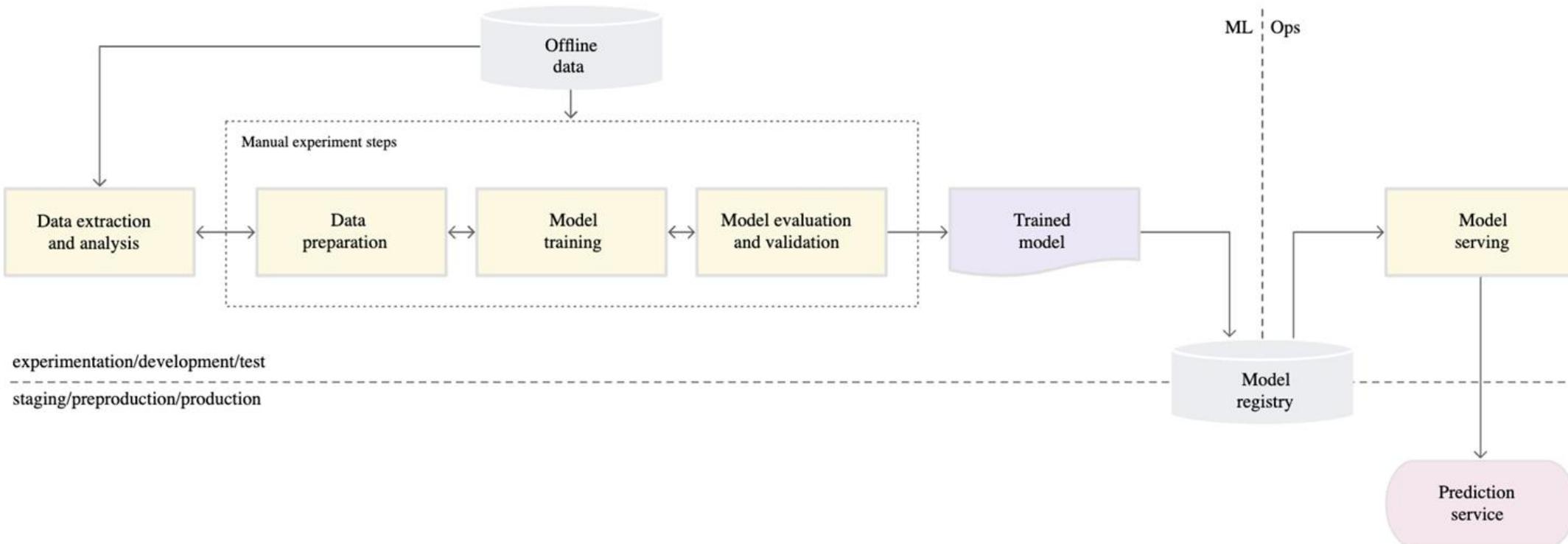
Процесс в ML

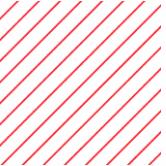


MLOPS Level 0



DS делает модели





Проблемы воспроизводимости

Level 1: Не могу воспроизвести обучение модели от начала до конца

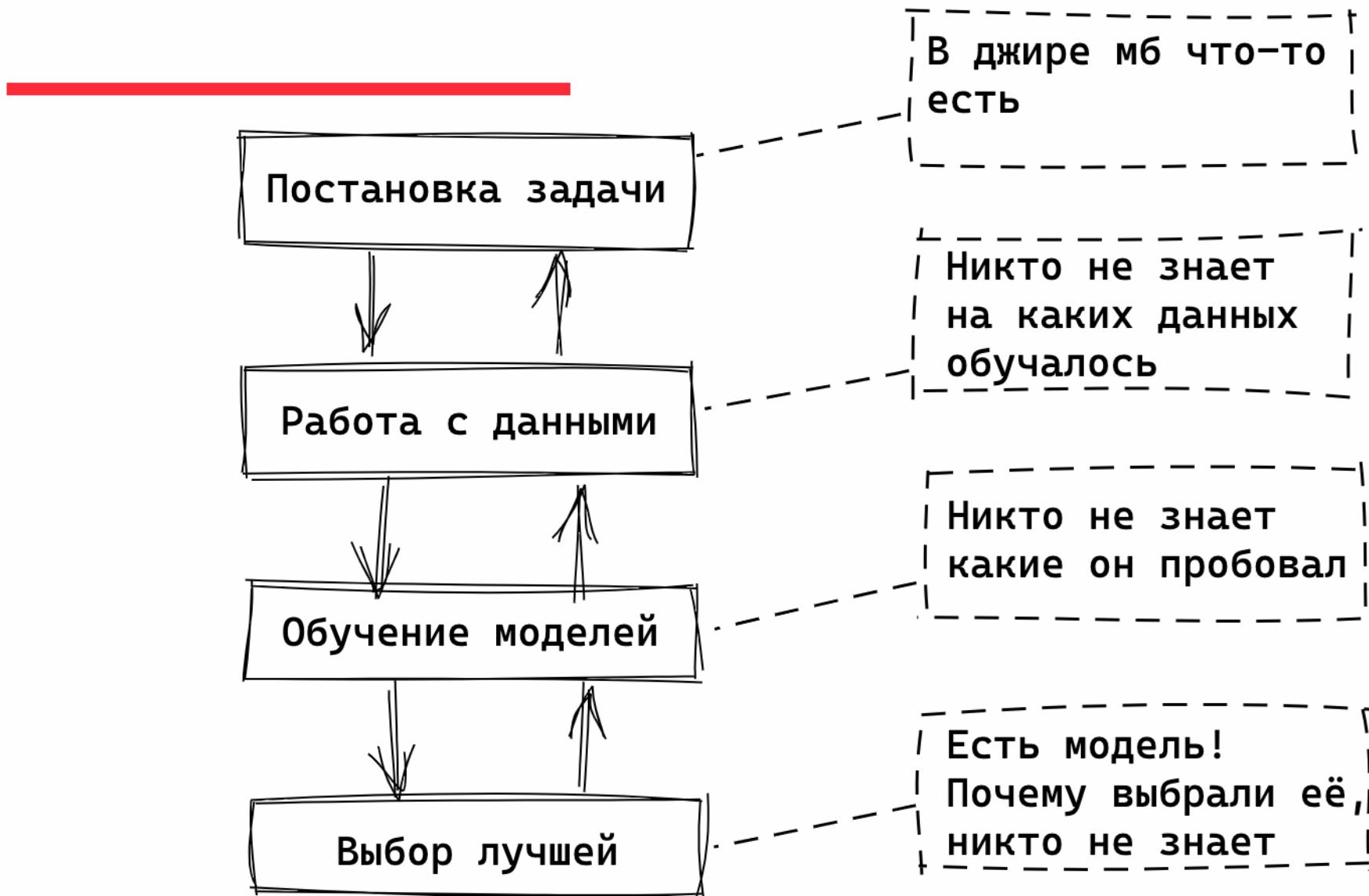


Проблемы воспроизводимости

Level 1: Не могу воспроизвести обучение финальной модели от начала до конца

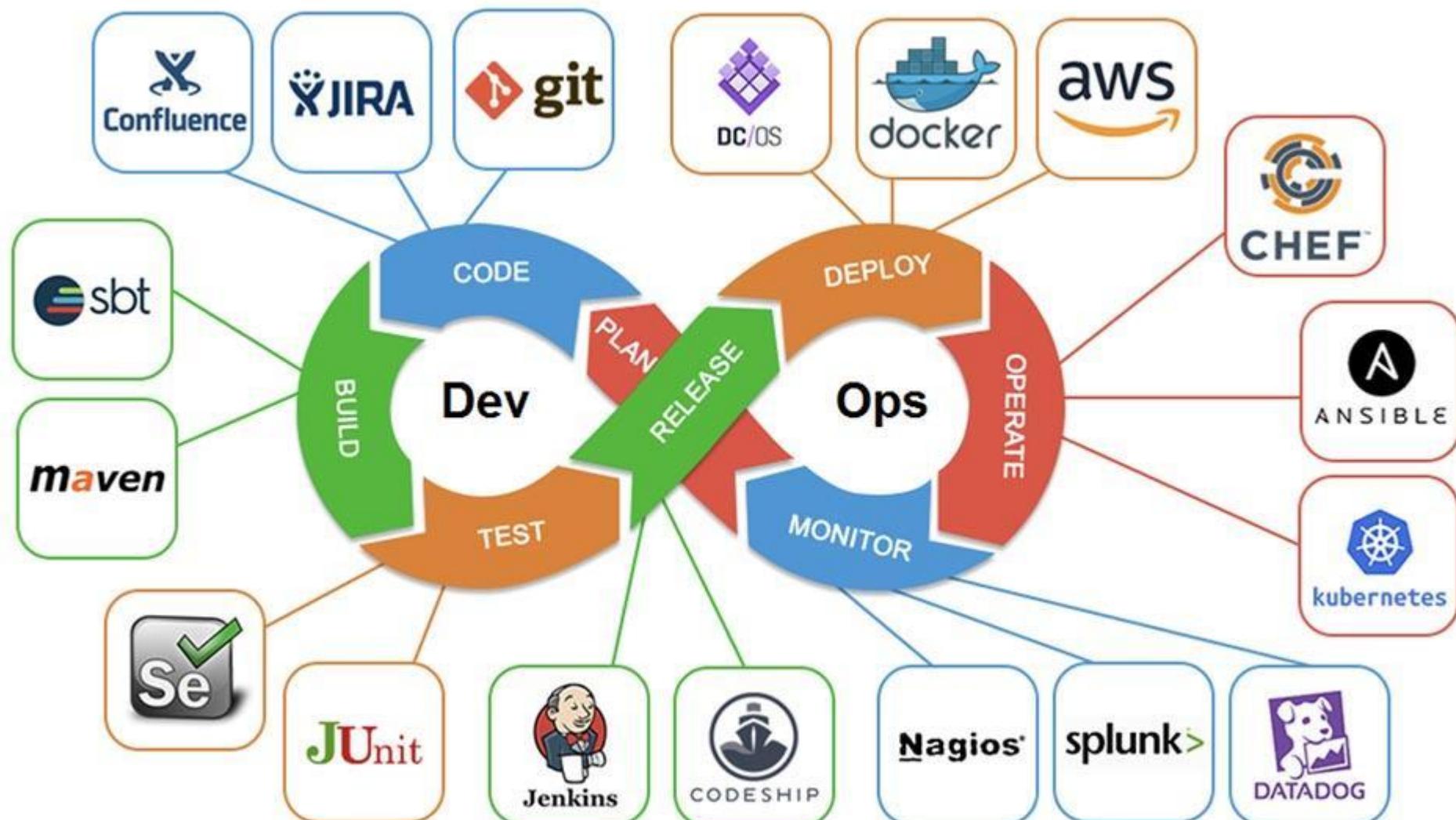
Level 2: Не могу воспроизвести любого сделанного эксперимента

Знания о сделанных экспериментах через год



Где-то я уже подобное
слышал

DEVOPS





Обучение это билд?

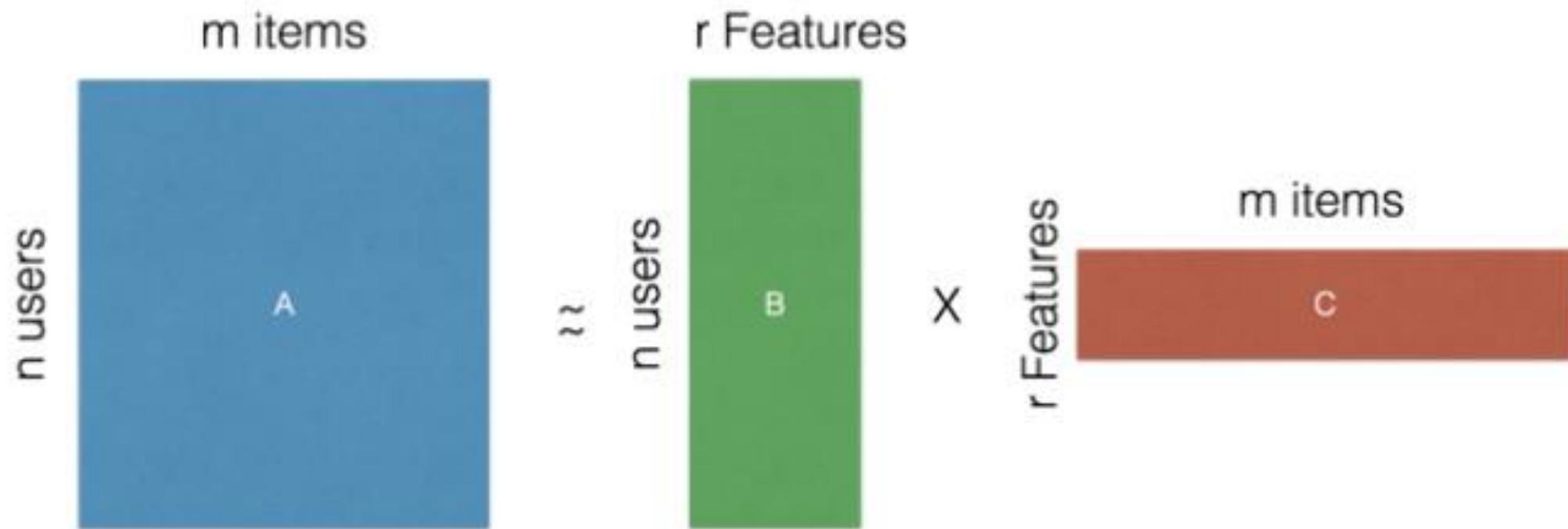
БИЛД	ОБУЧЕНИЕ
Возращает “бинарный” артефакт	Возращает “бинарный” артефакт
Принимает на вход код и зависимости	Принимает на вход код, зависимости, данные и параметры
Занимает минуты	Занимает часы
Делать build через CI -- это стандарт	Обучать руками и класть куда-то артефакт -- это “стандарт”
	DS любят поиграть с параметрами

2 класса задач

Нейронная сеть – выделяет картинки на изображении



Коллаборативная фильтрация





Разница

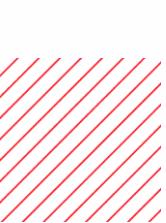
В одном случае, воспроизводимый пайплайн(автообучение и выкатка) — необходимость
В другом случае — “доп плюшка”

Но инструменты уменьшают этот разрыв.

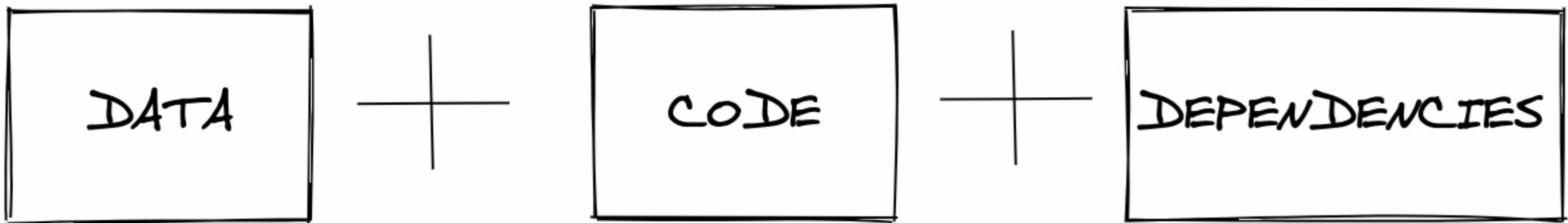
https://github.com/mercari/ml-system-design-pattern/blob/master/Lifecycle-patterns/Train-then-serve-pattern/design_en.md

https://github.com/mercari/ml-system-design-pattern/blob/master/Lifecycle-patterns/Training-to-serving-pattern/design_en.md

Смотрим на проект с
прошлой пары



Рецепт ML модели



Как хранятся история в гите

Вся история проекта находится в .git

В репозитории содержатся:

- объекты коммитов
- ссылки на коммиты и ветки
- конфигурация
- скрипты-хуки



Основные типы объектов

- blob
- tree
- commit



Blob

Blob - базовая единица хранения данных в Git.

Хранит snapshot содержимого файла.

В качестве имени объекта берется SHA1 хеш, содержимого файла и заголовка

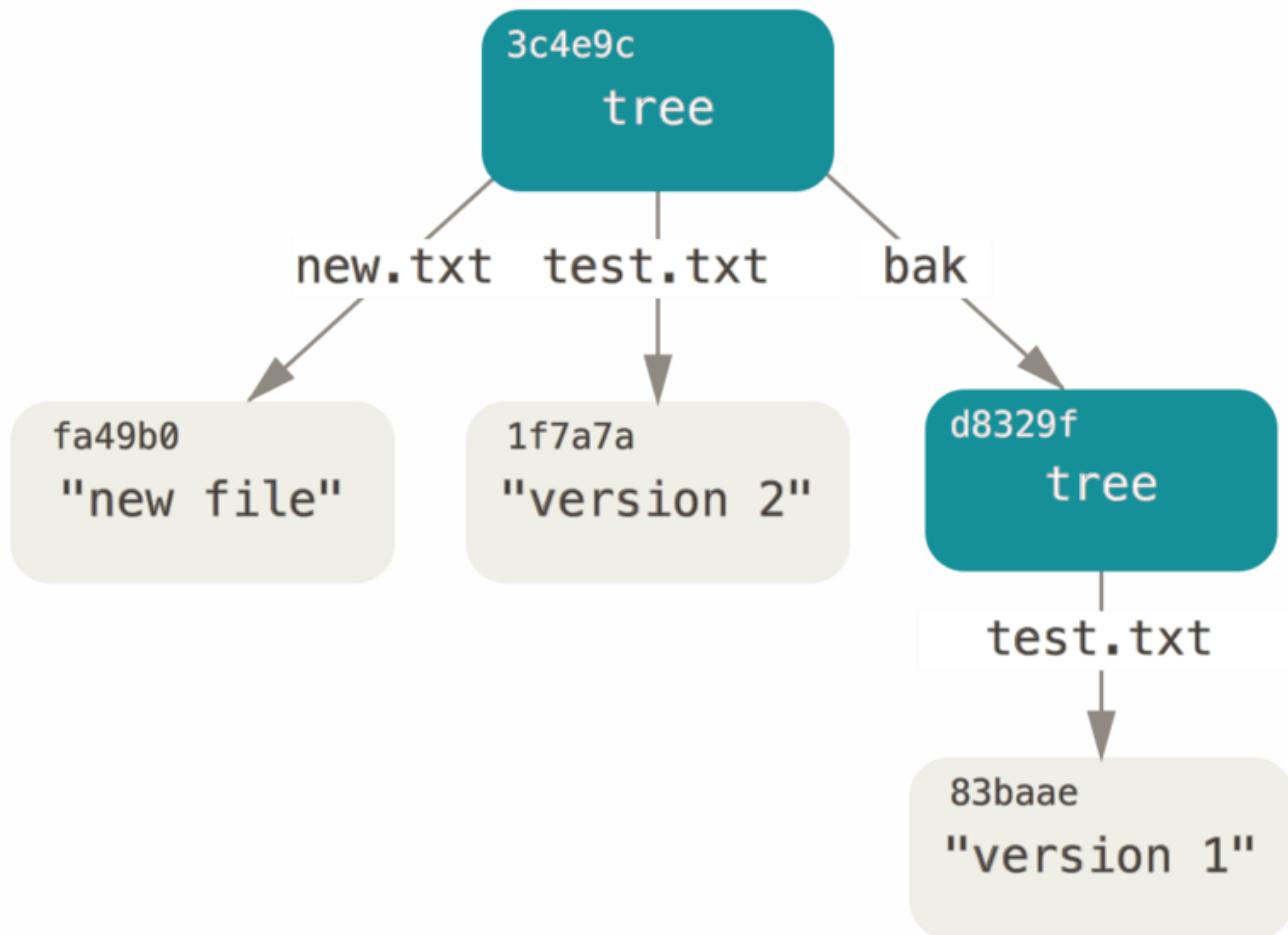
```
(ml_project_example) wh0106259:e4 mikhail.maryufich$ git cat-file -p e45508a6f68227d4c20a6424a97e  
710c13a7c89e  
import os  
import pickle  
from typing import List, Tuple  
  
import pandas as pd  
import pytest  
from py._path.local import LocalPath  
from sklearn.ensemble import RandomForestRegressor  
  
from ml_example.data.make_dataset import read_data  
from ml_example.entities import TrainingParams  
from ml_example.entities.feature_params import FeatureParams
```

Tree

Содержит ссылки на блобы и поддеревья

```
(ml_project_example) wh0106259:objects mikhail.maryufich$ git ls-tree c407c9f73dfc60305b532f67a3d4ee557a347e38
040000 tree 4ce0b0ec8e16229915d8ce820a9fb8b013b68af3    .idea
100644 blob 04cdcc8fe8a0fe5dd96b8312b1833357345c48f2    LICENSE
100644 blob e7e932da0a4fb78de4f1575e8920f6547c3d0b26    README.md
040000 tree b3b6650d0f96fbcac6e92366c7fe035c33ba66c0    configs
040000 tree c668e8a6e61f19ee938a5f25ee85bc464e1470b9    data
040000 tree 85c9484c8ecbf1d8f4fd249e366ff966fa5bf038    docs
040000 tree 1bda0c93f645af0521d3fdb1b8e2339afd11a356    ml_example
040000 tree a1bc1af530817862a71dc990d0a2ea6e18b3672b    models
040000 tree d564d0bc3dd917926892c55e3706cc116d5b165e    notebooks
100644 blob 7933e19a1f10690be0cbea44a3dbe5da7db03fa8    packed-refs
040000 tree d564d0bc3dd917926892c55e3706cc116d5b165e    references
040000 tree 8e1bce21b4452f12884ece241591d4901f08cb22    reports
100644 blob 4cc0538559fea8b062e3e20a75116e00a4740ed5    requirements.txt
100644 blob c13e70556a6bb1057649feb505ef0b773ec202bb    setup.py
040000 tree 682e305a9ae35d291eefd803cbf20bece13fe051    tests
100644 blob 762f425556de5eb8454a1a39628bb4acd12b6b6f    tox.ini
```

Tree



Commit

Коммит имеет автора, название, ссылку на дерево и родителей

```
commit 96cb466dc465cb4b46c258ac332a793eda7b2719 (HEAD -> repro_lecture, origin/repro_lecture)
Author: Mikhail Maryufich <mikhail-maryfich@yandex.ru>
Date:   Mon Apr 12 20:36:15 2021 +0300
```

add boto3 examples

```
commit 6daed4ddfb681da0321fd3d76047acf2cff11b90 (origin/main, origin/HEAD, main)
Author: Михаил Марюфич <Mihail-maryfich@yandex.ru>
Date:   Fri Apr 9 16:56:57 2021 +0300
```

Example project (#1)

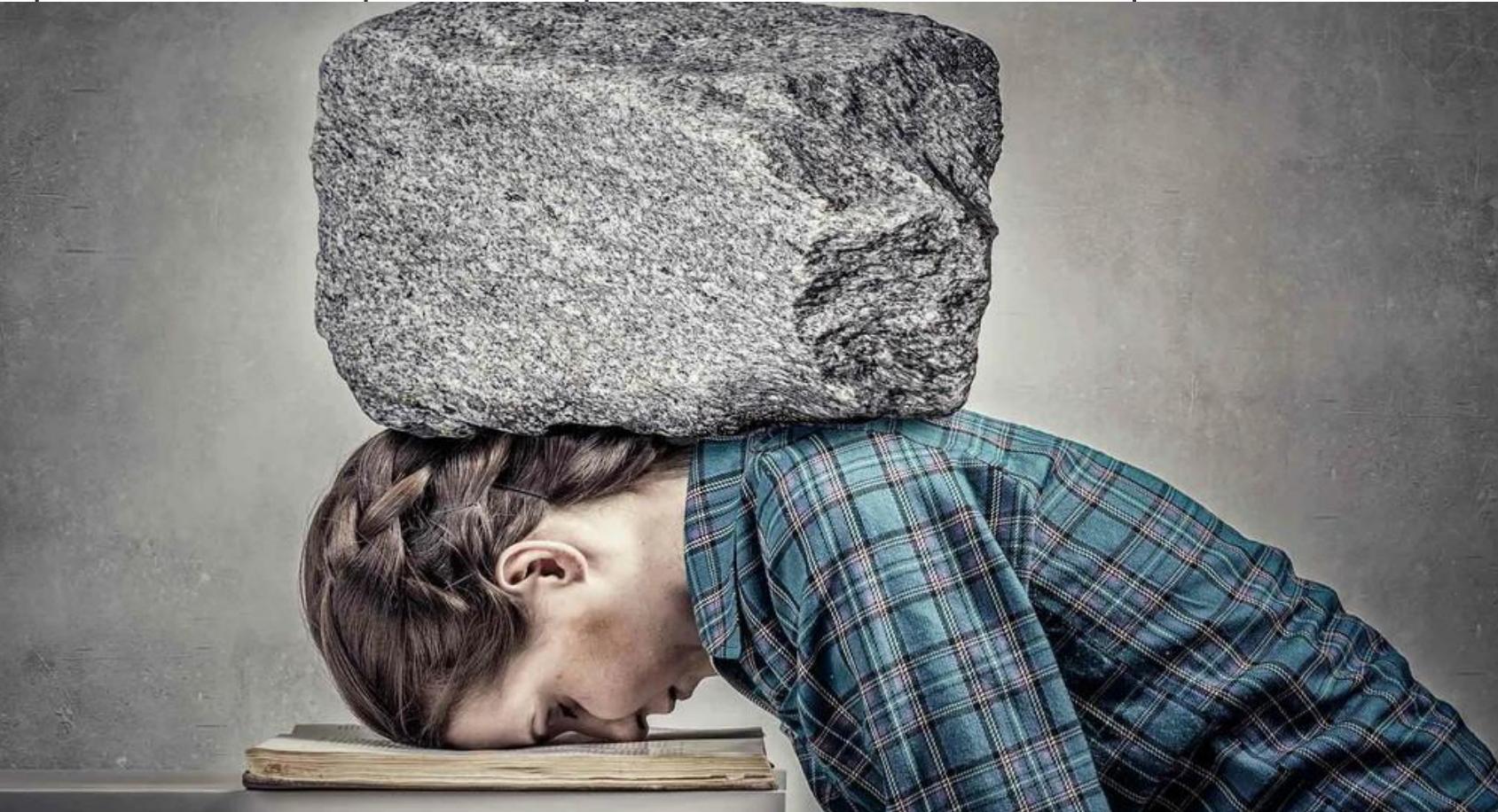
Пример проекта для МЛ

```
commit 2dd4ef9e393c338c80bfcaa4c65863d7d09f7400 (repro)
Author: Михаил Марюфич <Mihail-maryfich@yandex.ru>
Date:   Sun Apr 4 19:07:55 2021 +0300
```

Initial commit

Как хранятся файлы в git?

- При изменении файлы сохраняется его полный snapshot



Pack-файлы

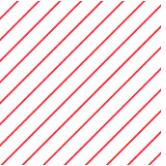
Если бы **Git** действительно хранил все объекты целиком (хоть и сжатые), папка `.git` представляла бы из себя огромный набор файлов, причем их было гораздо больше, чем в рабочей копии. Тем не менее этого не происходит, а появляются загадочные *pack-файлы*, в которые упакованы объекты. Как ни странно, но в интернете мало информации о том, как **Git** хранит данные в этих файлах, поэтому приведем отрывок из электронного письма Линуса Торвальдса, в котором дается некоторое пояснение насчет этих загадочных файлов (источник: gcc.gnu.org/ml/gcc/2007-12/msg00165.html):

▶ Скрытый текст

Если кратко, то в pack-файлах объекты группируются по схожести (например, тип и размер), после чего они сохраняются в виде «цепочек». Первый элемент цепочки представляет из себя самую новую версию объекта, а следующий за ним является диффом к предыдущему. Самые новые версии объекта считаются наиболее запрашиваемыми, поэтому они хранятся выше в цепочке.

Таким образом, **Git** всё же хранит диффы, но только на уровне непосредственного хранения данных. С точки зрения любого API уровнем выше, **Git** оперирует объектами целиком, что позволяет реализовывать различные стратегии слияния и легко разрешать конфликты.

<https://habr.com/ru/company/badoo/blog/163853/>



Почему плохо коммитить большие файлы в GIT

- При выполнении операции clone будут выкачиваться все версии всех больших файлов со всех коммитов всех веток(и это будет работать медленно)
- git не сможет оптимизировать хранение таких файлов (pack-файлы)

Где же все таки их хранить?



Google Cloud Storage



S3

Amazon S3 - объектное хранилище от Amazon

- key-value
- rest api
- можно download, upload, list файлов



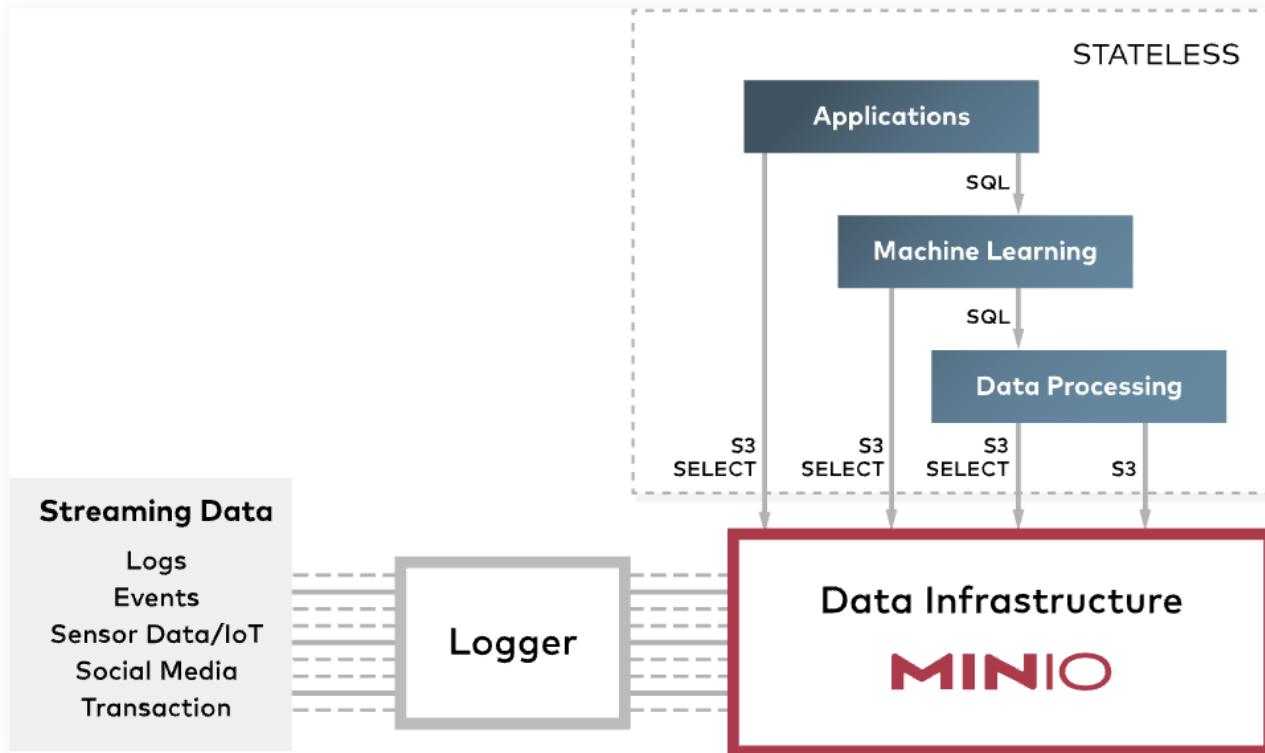
Смотрим на S3

S3 - compatible



Реверс-инжиниринг архитектуры Amazon S3

S3 - compatible



и облачные провайдеры — mail, yandex, etc

Чек-лист воспроизведимости

Чек-лист воспроизводимости

- Знаем ли мы на каких данных была обучена наша модель(можем ли их получить)?
- Знаем ли какой версией кода была обучена модель?
- Какие параметры передавались на вход модели?
- Насколько хорошо была обучена модель?
 - Можем ли повторить обучение?
 - Есть ли связь между моделью на продакшеном и кодом, данными, параметрами, оффлайн оценками?



Workaround

Вместе с сериализованным файлом модели
автоматически сохраняем:

- ссылку на данные
- ссылку на git repo и коммит в нем(либо релизной версии кода)
- параметры
- метрики



Workaround

Вместе с сериализованным файлом модели автоматически сохраняем:

- ссылку на данные
- ссылку на git repo и коммит в нем(либо релизной версии кода)
- параметры
- метрики



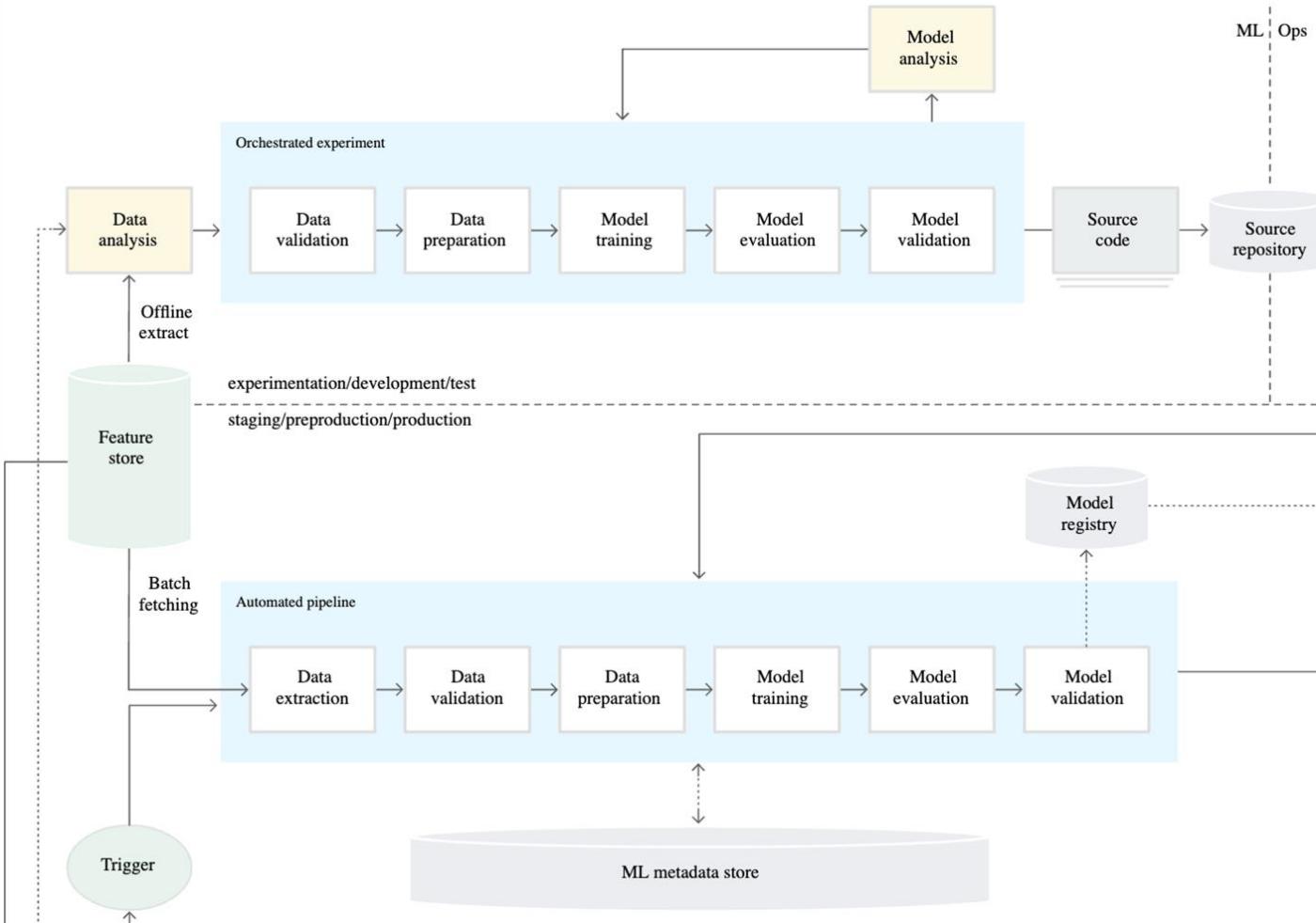
Называем папку `modelname_v1`, где-нибудь в вике
указываем ссылку на все это дело

Workaround

DON'T REWRITE!



Мы с вами изобрели ML metadata store





ML metadata store

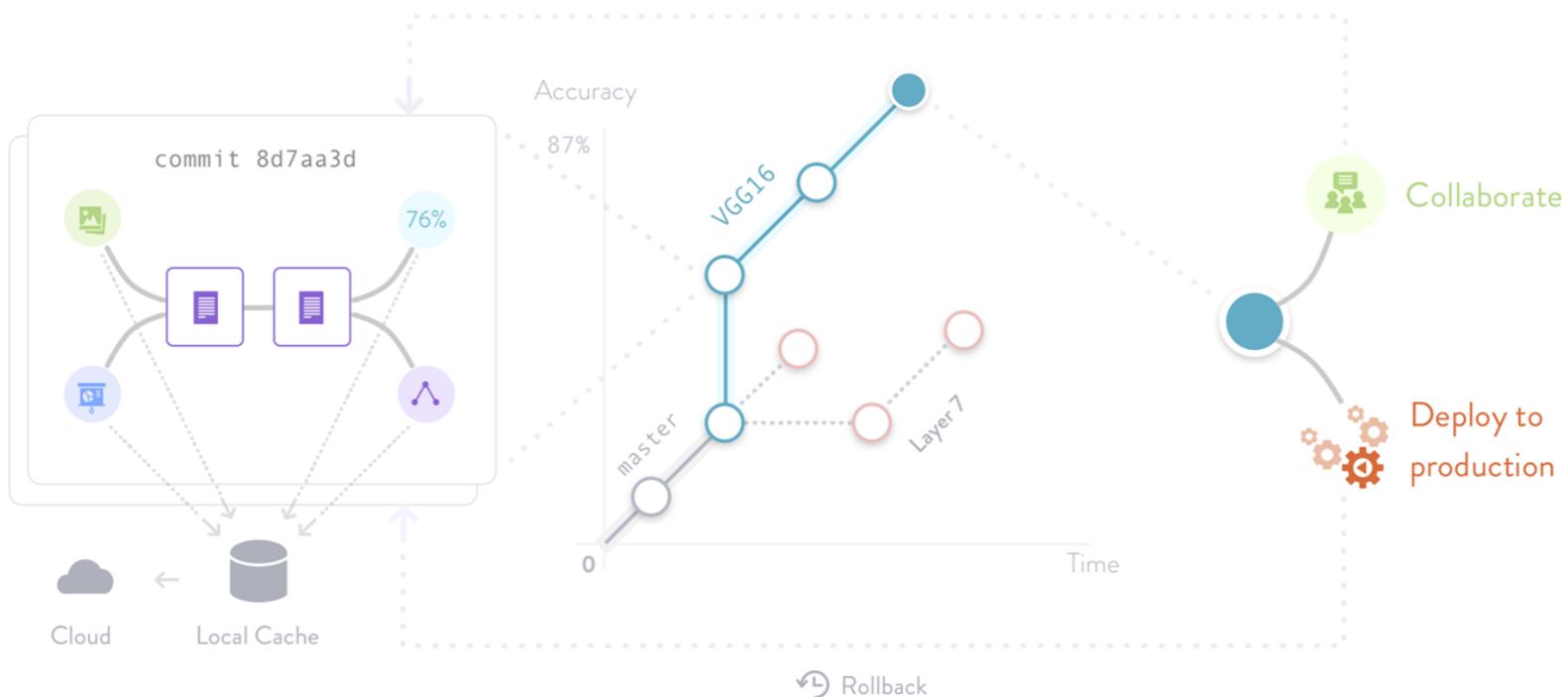
- Какой пайплайн исполнялся
- Кто его исполнял
- Какие параметры были даны на вход
- Какие метрики и артефакты были на выходе
- input, output для каждой из стадий

DVC

Что такое DVC?

DVC tracks ML models and data sets

DVC is built to make ML models shareable and reproducible. It is designed to handle large files, data sets, machine learning models, and metrics as well as code.



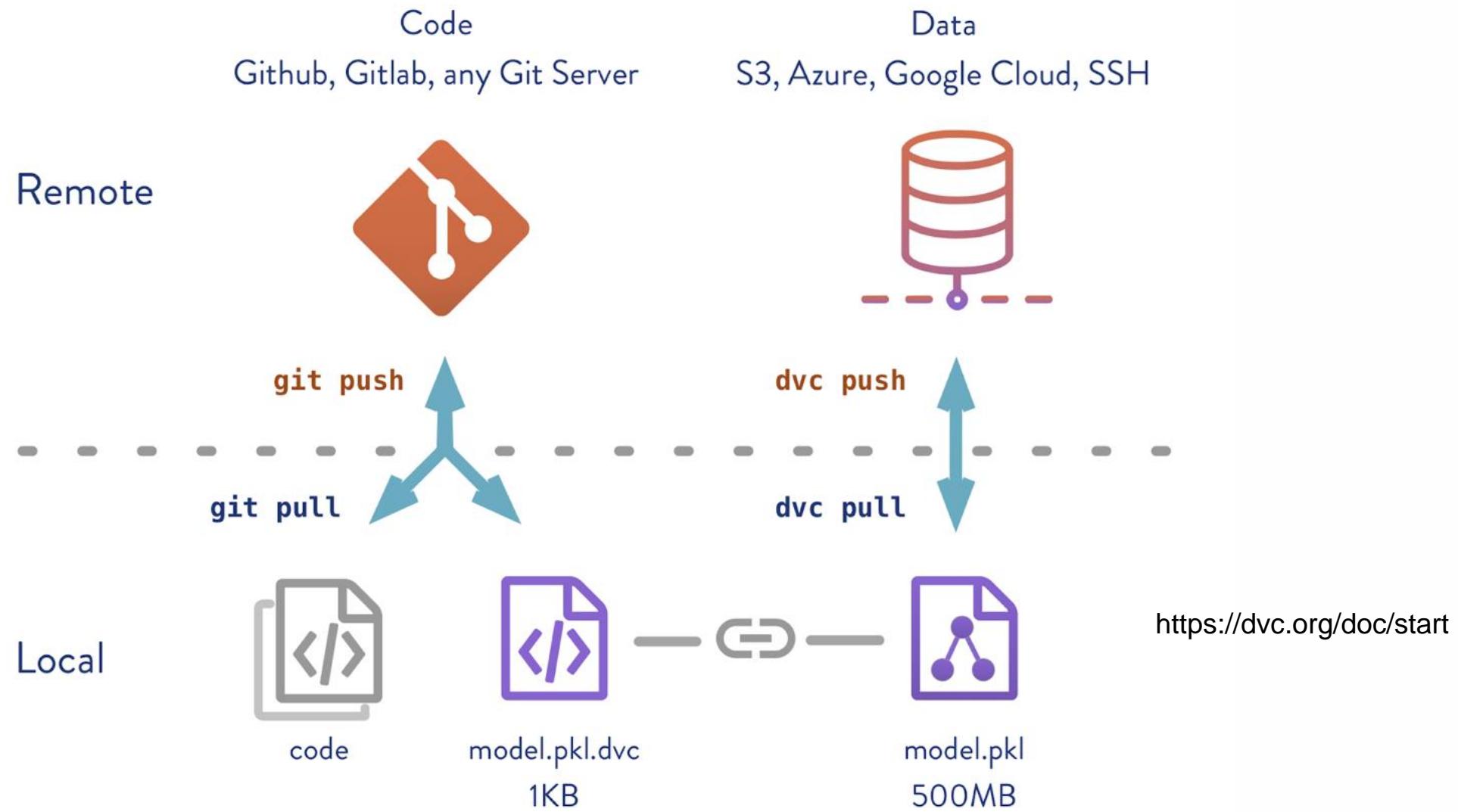
ML project version control

ML experiment management

Deployment & Collaboration

<https://dvc.org/>

Что такое DVC?



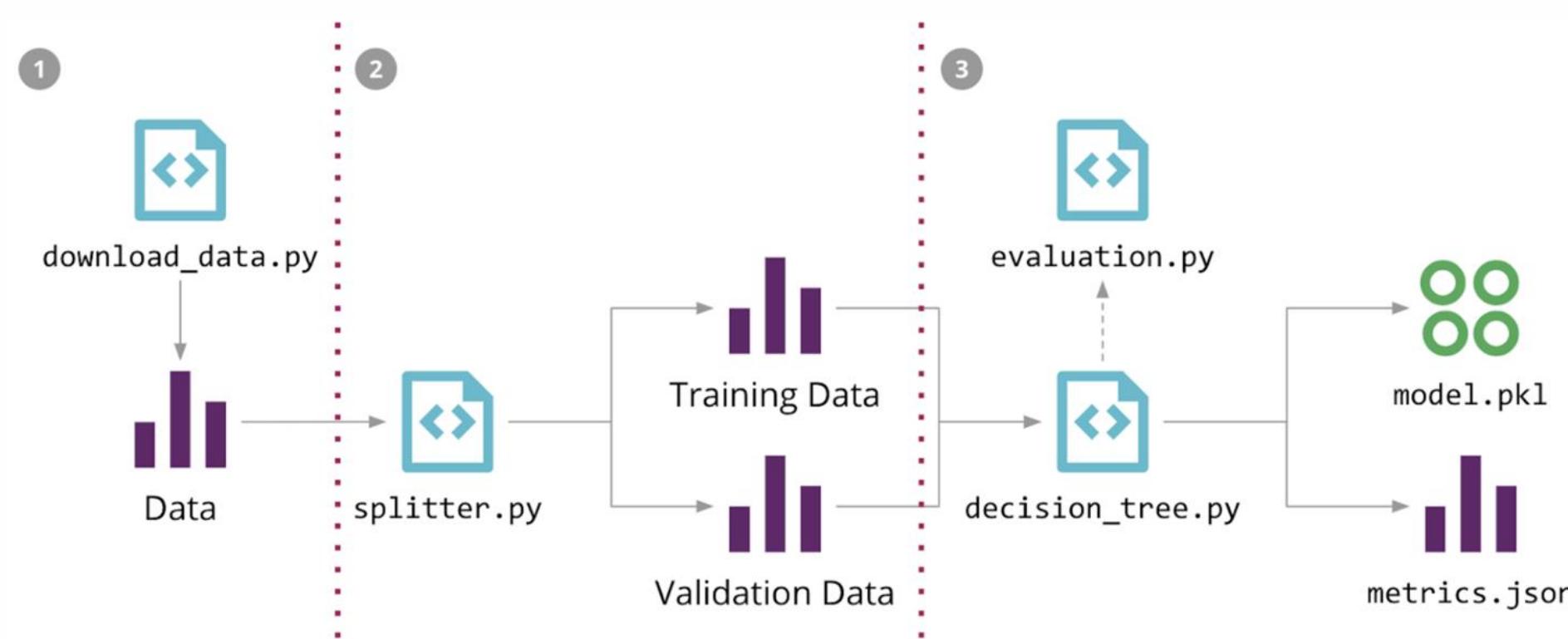


dvc remote

- Версии файлов хранятся локально, есть возможность делать checkout
- Есть возможность синхронизировать cache с удаленным специализированным хранилищем(dvc pull, dvc push)

```
['remote "myremote"']
url = /path/to/remote
[core]
remote = myremote
['remote "newremote"']
url = s3://mybucket/path
endpointurl = https://object-storage.example.com
```

DVC pipeline



Позволяет одновременно с изменениями трекать входы и выходы для составных частей пайплайна

Файлы dvc.yaml, dvc.lock

```
1 stages:
2   prepare:
3     cmd: python src/prepare.py data/data.xml
4     deps:
5       - data/data.xml
6       - src/prepare.py
7     params:
8       - prepare.seed
9       - prepare.split
10    outs:
11      - data/prepared
12 featurize:
13   cmd: python src/featurization.py data/prepared data/features
14   deps:
15     - data/prepared
16     - src/featurization.py
17   params:
18     - featurize.max_features
19     - featurize.ngrams
20   outs:
21     - data/features
22 train:
23   cmd: python src/train.py data/features model.pkl
24   deps:
25     - data/features
26     - src/train.py
27   params:
28     - train.n_estimators
29     - train.seed
30   outs:
31     - model.pkl
```

dvc.yaml

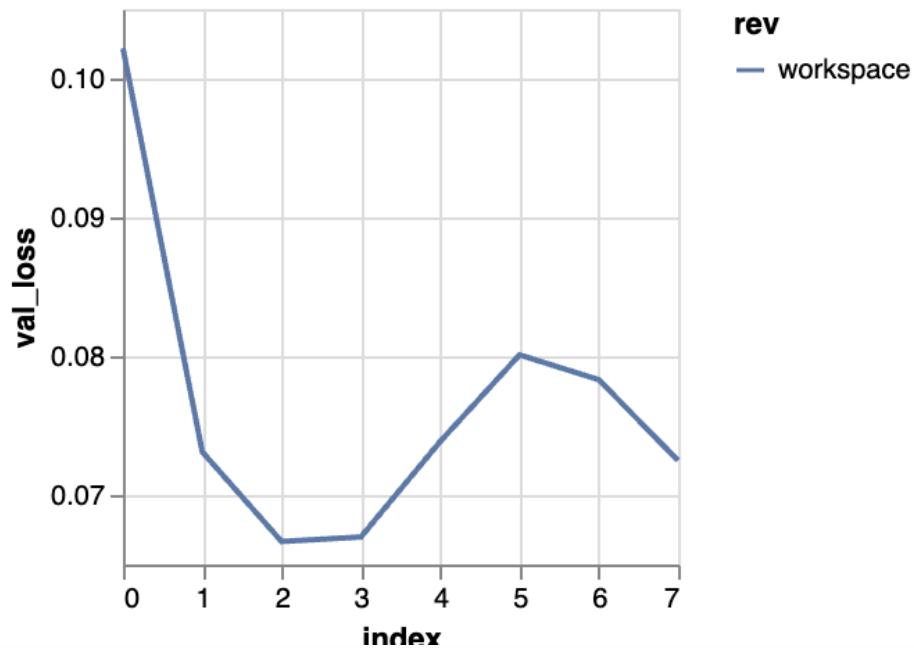
```
1 prepare:
2   cmd: python src/prepare.py data/data.xml
3   deps:
4     - path: data/data.xml
5       md5: a304afb96060aad90176268345e10355
6     - path: src/prepare.py
7       md5: 285af85d794bb57e5d09ace7209f3519
8   params:
9     params.yaml:
10    prepare.seed: 20170428
11    prepare.split: 0.2
12   outs:
13     - path: data/prepared
14       md5: 20b786b6e6f80e2b3fcf17827ad18597.dir
15 featurize:
16   cmd: python src/featurization.py data/prepared data/features
17   deps:
18     - path: data/prepared
19       md5: 20b786b6e6f80e2b3fcf17827ad18597.dir
20     - path: src/featurization.py
21       md5: 02180903aa0559ceaa6ff6ee2471d00
22   params:
23     params.yaml:
24       featurize.max_features: 1500
25       featurize.ngrams: 2
26   outs:
27     - path: data/features
28       md5: a1414b22382ffb76a153ab1f0d69241.dir
29 train:
30   cmd: python src/train.py data/features model.pkl
31   deps:
32     - path: data/features
33       md5: a1414b22382ffb76a153ab1f0d69241.dir
34     - path: src/train.py
35       md5: ad8e71b2cca4334a7d3bb6495645068c
36   params:
37     params.yaml:
38       train.n_estimators: 50
39       train.seed: 20170428
40   outs:
41     - path: model.pkl
42       md5: c8d307aa005d6974a8525550956d5fb3
```

dvc.lock

DVC DEMO

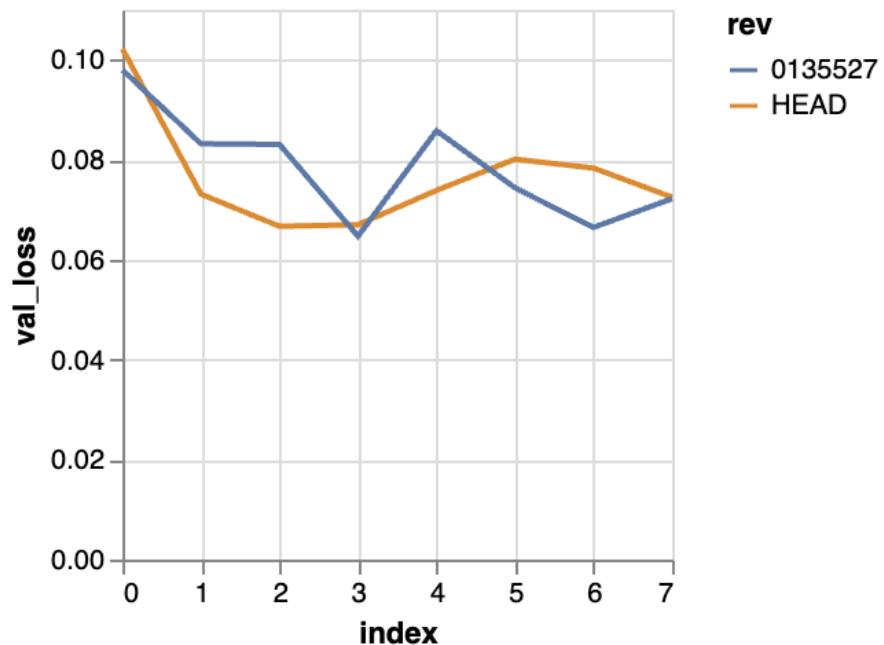
DVC metrics

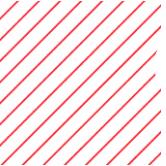
```
$ dvc plots show logs.csv  
file:///Users/usr/src/plots/logs.csv.html
```



DVC metrics

```
$ dvc plots diff -d logs.csv HEAD^  
file:///Users/usr/src/plots/logs.csv.html
```



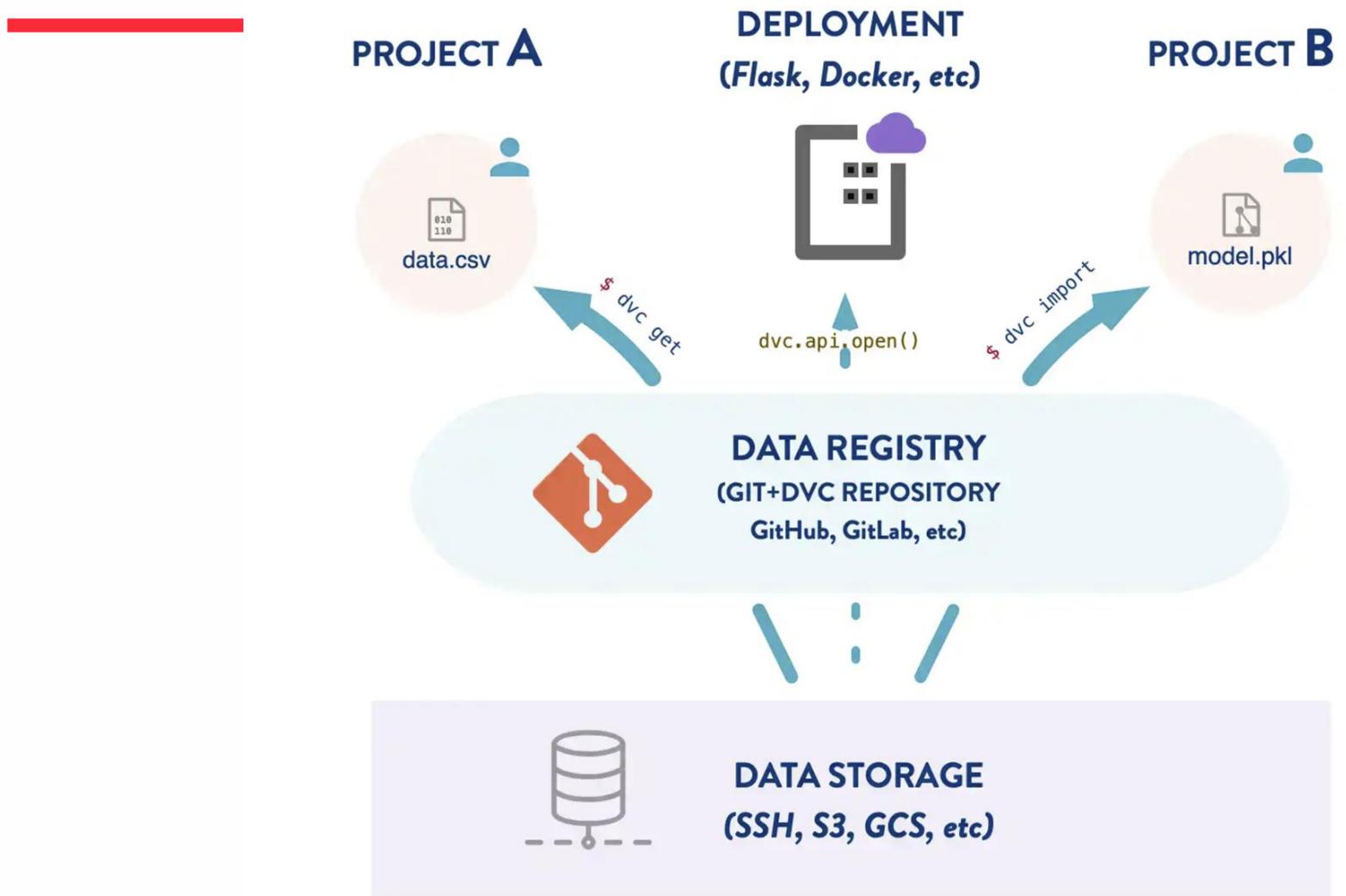


Отличие от GIT LFS

GIT LFS	DVC
Хранит данные на специальных серверах(рядом с GIT)(есть ограничения на размер файла)	Хранит данные в HDFS, S3, GOOGLE CLOUD, etc
Нет связи между кодом, параметрами и данными	Версионируемо хранит связку между кодом, параметрами и данными
Не всякий git server поддерживает LFS	Можно использовать с любым git server

DVC UI

DVC data registry



DVC data registry

Mikhail Maryufich / data_registry

Source

master ... data_registry /

- .dvc
- comment_ranking
- music_classification
- nsfw_classification
- spam_classification

README.md init

README.md

Example Data Registry for DS+PROD=ONE LOVE

This screenshot shows the DVC UI interface. On the left, there's a sidebar with various icons for user profile, download, commit, upload, and settings. The main area displays a list of files under the 'data_registry' branch. It includes a folder named '.dvc' and four subfolders: 'comment_ranking', 'music_classification', 'nsfw_classification', and 'spam_classification'. Below these are two README files: 'README.md' and 'init'. At the bottom, a large 'README.md' block contains the text 'Example Data Registry for DS+PROD=ONE LOVE'. The 'master' dropdown button is highlighted with a blue border.

Mikhail Maryufich / data_registry

Source

master ... data_registry /

Filter tags

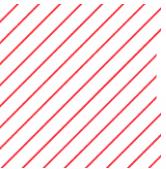
Tags

- spam_classification_v1
- music_classification_v3
- music_classification_v2
- music_classification_v1
- nsfw_classification_v1
- nsfw_classification_v2
- comment_ranking_v2
- comment_ranking_v1

No more tags

This screenshot shows the DVC UI interface. It's similar to the first one but focuses on tags. The 'Tags' tab is selected in the navigation bar. A 'Filter tags' input field is present. Below it is a list of tags, each preceded by a small tag icon. The tags listed are: 'spam_classification_v1', 'music_classification_v3', 'music_classification_v2', 'music_classification_v1', 'nsfw_classification_v1', 'nsfw_classification_v2', 'comment_ranking_v2', and 'comment_ranking_v1'. A message 'No more tags' is displayed at the bottom right of the tag list. The sidebar and other UI elements are identical to the first screenshot.

MLFLOW



MLFLOW

MLflow is an open source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow currently offers four components:

MLflow Tracking

Record and query experiments: code, data, config, and results

[Read more](#)

MLflow Projects

Package data science code in a format to reproduce runs on any platform

[Read more](#)

MLflow Models

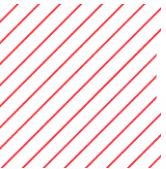
Deploy machine learning models in diverse serving environments

[Read more](#)

Model Registry

Store, annotate, discover, and manage models in a central repository

[Read more](#)



MLFLOW TRACKING

MLFlow Tracking отвечает за

- 1) логирование параметров, метрик и артефактов экспериментов
- 2) За их визуализацию и сравнение между собой

MLFLOW TRACKING

Experiments + <

Default

Search Experiments

Experiment ID : 0

Artifact Location :
file:///Users/mikhail.maryufich/PycharmProjects/mlflow_tutorial/examples/sklearn_elasticnet_wine/mlruns/0

Default

▼ Notes

None

Search Runs: metrics.rmse < 1 and params.model = "tree" and tags.mlflow.source.type = "LOCAL" ? State: Active Search Clear

Showing 4 matching runs Compare Delete Download CSV Columns

						Parameters		Metrics			
						alpha	l1_ratio	mae	r2	rmse	
<input type="checkbox"/>	Start Time	Run Name	User	Source	Version	0.1	0.5	0.562	0.243	0.731	
<input type="checkbox"/>	2020-10-21 20:11:55	-	mikhail.mar...	train.py	-	0.1	0.5	0.562	0.243	0.731	
<input type="checkbox"/>	2020-10-21 20:11:51	-	mikhail.mar...	train.py	-	0.1	0.4	0.56	0.245	0.73	
<input type="checkbox"/>	2020-10-21 20:11:47	-	mikhail.mar...	train.py	-	0.4	0.4	0.597	0.172	0.764	
<input type="checkbox"/>	2020-10-21 20:11:43	-	mikhail.mar...	train.py	-	0.4	0.3	0.586	0.193	0.755	

MLFLOW TRACKING

Default > Comparing 3 Runs

Run ID:	f7995e86280a478591dbb05fe5f90182	c8ffd01f8a0c4559a83c78e4141fcfd3	c63dbfb91ce641728a02941da6104222
Run Name:			
Start Time:	2020-10-21 20:11:55	2020-10-21 20:11:51	2020-10-21 20:11:47
Parameters			
alpha	0.1	0.1	0.4
l1_ratio	0.5	0.4	0.4
Metrics			
mae 	0.562	0.56	0.597
r2 	0.243	0.245	0.172
rmse 	0.731	0.73	0.764

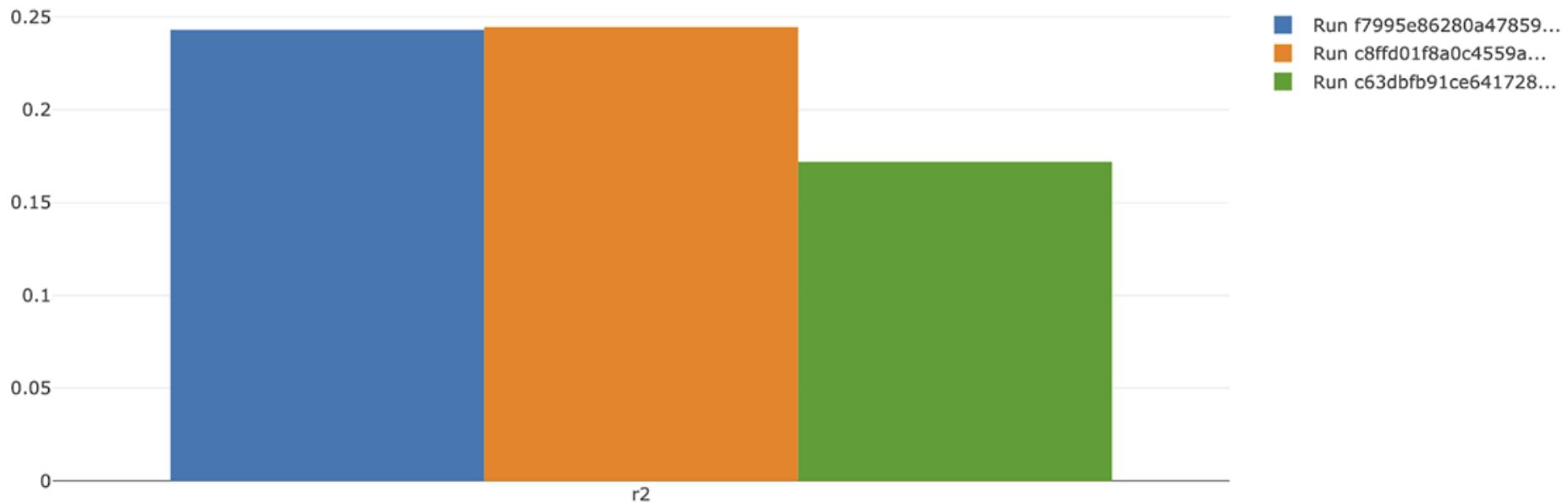
MLFLOW TRACKING – сравнение запусков

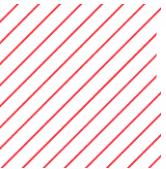
Default > Comparing 3 Runs > r2

Y-axis:

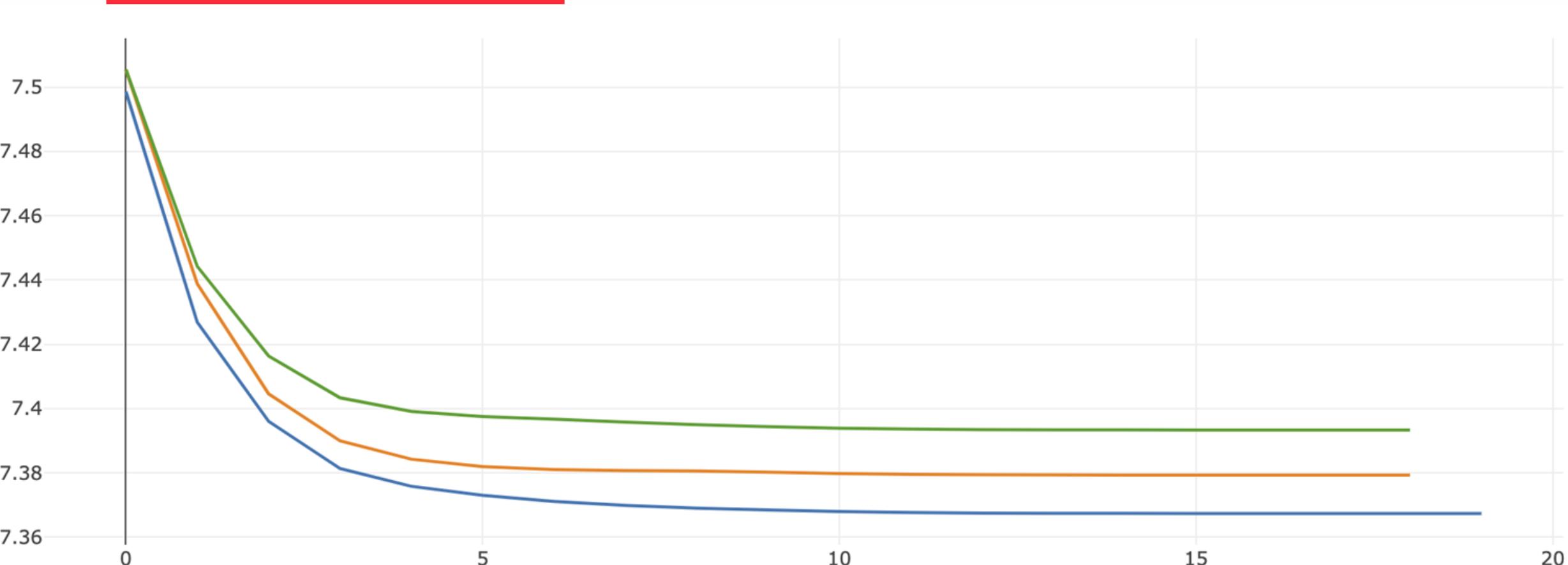
r2 x

Y-axis Log Scale: Off





MLFLOW TRACKING – сравнение запусков



MLFLOW TRACKING — сравнение запусков

```
with mlflow.start_run():
    lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
    lr.fit(train_x, train_y)

    predicted_qualities = lr.predict(test_x)

    (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)

    mlflow.log_param("alpha", alpha)
    mlflow.log_param("l1_ratio", l1_ratio)
    mlflow.log_metric("rmse", rmse)
    mlflow.log_metric("r2", r2)
    mlflow.log_metric("mae", mae)
```



MLFLOW PROJECTS – сравнение запусков

MLproject

- МОЖНО ДОКУМЕНТИРОВАТЬ ОКРУЖЕНИЕ И ИСПОЛНЯЕМЫЕ КОМАНДЫ

```
name: docker-example

docker_env:
  image: mlflow-docker-example

entry_points:
  main:
    parameters:
      alpha: float
      l1_ratio: {type: float, default: 0.1}
    command: "python train.py --alpha {alpha} --l1-ratio {l1_ratio}"
```



MLFLOW PROJECTS – сравнение запусков

Унифицированный запуск проекта

`mlflow run`

Есть возможность передавать параметры

`mlflow run -P alpha=0.2`

Можно запустить конкретную стадию

`mlflow run -e main`

Можно запускать на kubernetes

`mlflow run --backend kubernetes --backend-config config.json`

<https://youtu.be/SGawkljBoGE>

MLFLOW PROJECTS — минусы

- Неудобная работа со стадиями
- Неудобная работа с параметрами
- Не оч полезно, если не python

```
entry_points:  
  main:  
    parameters:  
      epochs: {type: int, default: 30}  
    command: "python train.py --epochs {epochs}"
```

```
import mlflow  
import argparse  
  
if __name__ == "__main__":  
    parser = argparse.ArgumentParser()  
    parser.add_argument "epochs", default=10)  
    args = parser.parse_args()  
  
    mlflow.log_param("epochs", args.epochs)
```



MLFLOW models

Модели можно сохранять из run прям в формате, готовом для использования

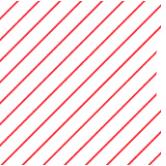
- Python Function (`python_function`)
- R Function (`crate`)
- H₂O (`h2o`)
- Keras (`keras`)
- MLeap (`mleap`)
- PyTorch (`pytorch`)
- Scikit-learn (`sklearn`)
- Spark MLlib (`spark`)
- TensorFlow (`tensorflow`)
- ONNX (`onnx`)
- MXNet Gluon (`gluon`)
- XGBoost (`xgboost`)
- LightGBM (`lightgbm`)
- Spacy(`spaCy`)
- Fastai(`fastai`)



MLFLOW models — serve

Есть встроенный serve

```
$ mlflow models serve -m runs:/my-run-id/model-path &  
  
$ curl http://127.0.0.1:5000/invocations -H 'Content-Type: application/json' -d '{  
  "columns": ["a", "b", "c"],  
  "data": [[1, 2, 3], [4, 5, 6]]  
}'
```



MLFLOW models – deploy

Можно задеплоить

- Microsoft Azure-ML
<https://www.mlflow.org/docs/latest/models.html#deploy-a-python-function-model-on-microsoft-azure-ml>
- Amazon SageMaker <https://www.mlflow.org/docs/latest/models.html#deploy-a-python-function-model-on-amazon-sagemaker>
- Можно экспортнуть как Apache Spark Udf и юзать в кластере
<https://www.mlflow.org/docs/latest/models.html#export-a-python-function-model-as-an-apache-spark-udf>

MLFLOW models — минусы

Чисто питоновская история

Serving не оч производительный

Serving принимает на json или pandas dataframe



MLFLOW MODEL REGISTRY

The screenshot displays the MLflow Model Registry interface. At the top, there is a dark header bar with the "mlflow" logo on the left and "GitHub Docs" links on the right. Below the header, the main content area shows a "Run" page for a specific run identified by the ID `ed089cf2d5d24b8880446761987b7ac2`. The run was created on `2019-10-16 22:49:25` by user `sid.murching`. The source code is `train_predict.py`, and the Git commit hash is `fcefefcd6f830c43dc8cdc048b83ac92fd8d5d7e`. The duration of the run was `6.2s`.

The page includes sections for "Notes" (which is currently empty), "Parameters", "Metrics", and "Tags". Under "Tags", there is a table with columns "Name", "Value", and "Actions". A message indicates "No tags found." Below this, there is a form to "Add Tag" with fields for "Name" and "Value" and a "Add" button.

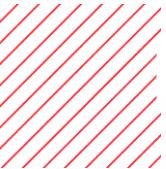
At the bottom, there is a section for "Artifacts" with a list including "model" (selected), "MLmodel", "conda.yaml", and "tfmodel". To the right of the artifact list, there is a "Full Path" field containing `/tmp/swag/0/ed089cf2d5d24b8880446761987b7ac2/artifacts/model`, a "Size" field showing `0B`, and a blue "Register Model" button.

MLFLOW MODEL REGISTRY

The screenshot shows the MLflow Model Registry interface. At the top, there's a navigation bar with the 'mlflow' logo, a GitHub link, and a 'Docs' link. Below the navigation bar, the page title is 'Default > Run ed089cf2d5d24b8880446761987b7ac2'. The run details are listed: Date: 2019-10-16 22:49:25, Source: train_predict.py, Git Commit: fcfefecd6f830c43dc8cdc048b83ac92fd8d5d7e, User: sid.murching, and Duration: 6.2s.

A modal dialog titled 'Register Model' is open in the center. It contains a message: 'Once registered, the model will be available in the model registry and become public.' Below this, there's a section for 'Model' with a dropdown menu labeled '+ Create New Model'. A field for 'Model Name' is filled with 'Model A'. At the bottom of the dialog are 'Cancel' and 'Register' buttons.

On the left side of the main content area, there are sections for 'Notes' (None), 'Parameters', 'Metrics', 'Tags' (with an 'Add Tag' button and fields for 'Name' and 'Value'), and 'Artifacts' (listing 'model' with 'MLmodel', 'conda.yaml', and 'tfmodel' files). A 'Register Model' button is located at the bottom right of the artifact list.



MLFLOW MODEL REGISTRY

[GitHub](#) [Docs](#)

Registered Models

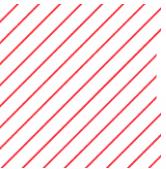
Name	Latest Version	Staging	Production	Last Modified
Model A	Version 1	Version 1	—	2019-10-16 22:51:19
Model B	Version 1	—	—	2019-10-16 22:51:52

MLFLOW MODEL REGISTRY

Ко всему этому есть rest api и возможность делать кодом из клиента
Это позволяет строить релизные циклы на CICD

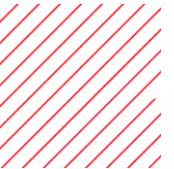
```
client = MlflowClient()
result = client.create_model_version(
    name="sk-learn-random-forest-reg-model",
    source="mlruns/0/d16076a3ec534311817565e6527539c0/artifacts/sklearn-model",
    run_id="d16076a3ec534311817565e6527539c0"
)
```

MLFLOW DEMO



MANAGED MLFLOW

<https://databricks.com/product/managed-mlflow>



Аналоги

- 1) <https://www.wandb.com/>
- 2) <https://neptune.ai/>
- 3) <https://github.com/IDSIA/sacred>
- 4) <https://aws.amazon.com/ru/sagemaker/>
- 5) <https://github.com/allegroai/clearml>

GITFLOW в ML



Основная проблема, что если это
делается руками, то всегда можно
накосячить

GITFLOW в ML



CML

First CML Report

.gitlab-ci.yml

```
stages:
- cml_run

cml:
  stage: cml_run
  image: dvcorg/cml-py3:latest
  script:
    - dvc pull data
    - pip install -r requirements.txt
    - dvc repro

    # Compare metrics to master
    - git fetch --prune
    - dvc metrics diff --show-md master >> report.md

    # Visualize loss function diff
    - dvc plots diff
      --target loss.csv --show-vega master > vega.json
    - vl2png vega.json | cml-publish --md >> report.md
    - cml-send-comment report.md
```

GitLab

Copy

DVC

Tensorboard

Cloud Computing

CML Report

Elle O'Brien @andronovhopf commented on commit e44c21fc 6 minutes ago

Path	Metric	Value	Change
metrics.json	accuracy	0.8694	-0.00296
metrics.json	precision	0.86793	0.00375
metrics.json	recall	0.87352	-0.01216

rev

master

workspace

loss

index

Reply...

Путь модели в OK.ru

JIRA

SPA / SPA-9610 Модель на помещении: внутри или снаружи / SPA-9655

Обучить модель внутри/снаружи

Редактировать Кomentарий Назначить Еще... Return to development Release Экспорт

Детали задачи

Тип:	Подзадача	Статус:	РЕШЕННЫЕ
Приоритет:	Основной	Решение:	Готово
Компоненты:	Рябина		
Метки:	Не заполнено		
Project affected:	Анти-спам		
Phase:	Modeling		

Описание
Обучить модель в облаке и зарелизить

Вложенные файлы
Перенесите файлы, чтобы прикрепить, или обзор.

Активность

Все Комментарии Журнал работ История изменений Активность

Для этого запроса еще нет комментариев.

Даты

Создано:	27.08.2020 12:06
Обновлено:	27.08.2020 15:18
Дата решения:	27.08.2020 15:18

Разработка

1 ветка	Обновленный 27.08.2020 15:12
5 коммитов	Последний 27.08.2020 15:15
1 pull-запрос	Обновленный 27.08.2020 15:15

Создать ветку

https://youtu.be/6ETZn7gc_j8

PULL REQUEST

A screenshot of a GitHub pull request page. At the top, it shows a red horizontal bar, the author's name "Mikhail Maryufich", the target branch "SPA-9655 → master", and a green "MERGED" button. Below this is a large blue rectangular redaction box. Underneath are tabs for "Overview", "Diff", and "Commits", with "Overview" being the active tab. The "Details" section shows the author created the pull request on 27 Aug 2020 with the message "Обучил модельку определять внутри квартиры или снаружи". The "Activity" section lists several events:

- Mikhail Maryufich MERGED SPA-9655 to master in commit 171017f0403 27 Aug 2020
- An emoji user marked the pull request as APPROVED 27 Aug 2020
- An emoji user marked the pull request as APPROVED 27 Aug 2020
- Mikhail Maryufich updated the reviewers 27 Aug 2020, adding three users: +, 🐱, and 🧑
- Neural Network Trainer Service UPDATED the pull request by adding 1 commit 27 Aug 2020, with a commit hash 2a267b147c3 and a large blue redaction box.
- Mikhail Maryufich UPDATED the pull request by adding 1 commit 27 Aug 2020, with a commit hash 0f0f94f88cb and the message "SPA-9655: master data".

https://youtu.be/6ETZn7gc_j8

В гите ссылка на mlflow

The screenshot shows a GitHub pull request interface. At the top left, it says "SPA-9655: [REDACTED]". Below that, there are three tabs: "Overview", "Diff" (which is selected), and "Commits". On the left, there's a sidebar with "Show diff of" and a dropdown menu containing "All changes in this pull request". The main area displays a diff of a file named "mlflow_link.md" which has been "ADDED". The diff shows a single line of code: "1 + https://mlflow.tools.kc.odkl.ru/#/experiments/367/runs/b75e609247d1429990b1f0e1a7fd3e61". A search bar is located at the bottom left of the main area.

https://youtu.be/6ETZn7gc_j8

MLFLOW METRICS

The screenshot shows a web browser displaying the MLflow UI at mlflow.tools.kc.odkl.ru/#/experiments/367/runs/b75e609247d1429990b1f0e1a7fd3e61. The page header includes a red horizontal bar. The main content area shows a summary of the run, including:

- Date: 2020-08-27 14:10:51
- Duration: 1.0h
- Source: ok_pipelines
- Status: FINISHED

Below the summary, there are sections for Notes, Parameters, and Metrics, each with a table:

Name	Value
base_lr	6.944e-5
epoch	19
precision_alt	1
precision_inside	1
precision_outside	0.99
recall_alt	0.78
recall_inside	0.13
recall_outside	0.9
test_accuracy_score	0.99
test_average_precision_score	0.995
test_cohen_kappa_score	0.944
test_f1_score	0.99
test_roc_auc_score	0.994
threshold	0.99
train_AccuracyScore	0.994
train_loss	0.102
val_AccuracyScore	0.993
val_AveragePrecisionScore	0.994
val_RocAucScore	0.996
val_loss	0.102

https://youtu.be/6ETZn7gc_j8

MLFLOW MODEL REGISTRY

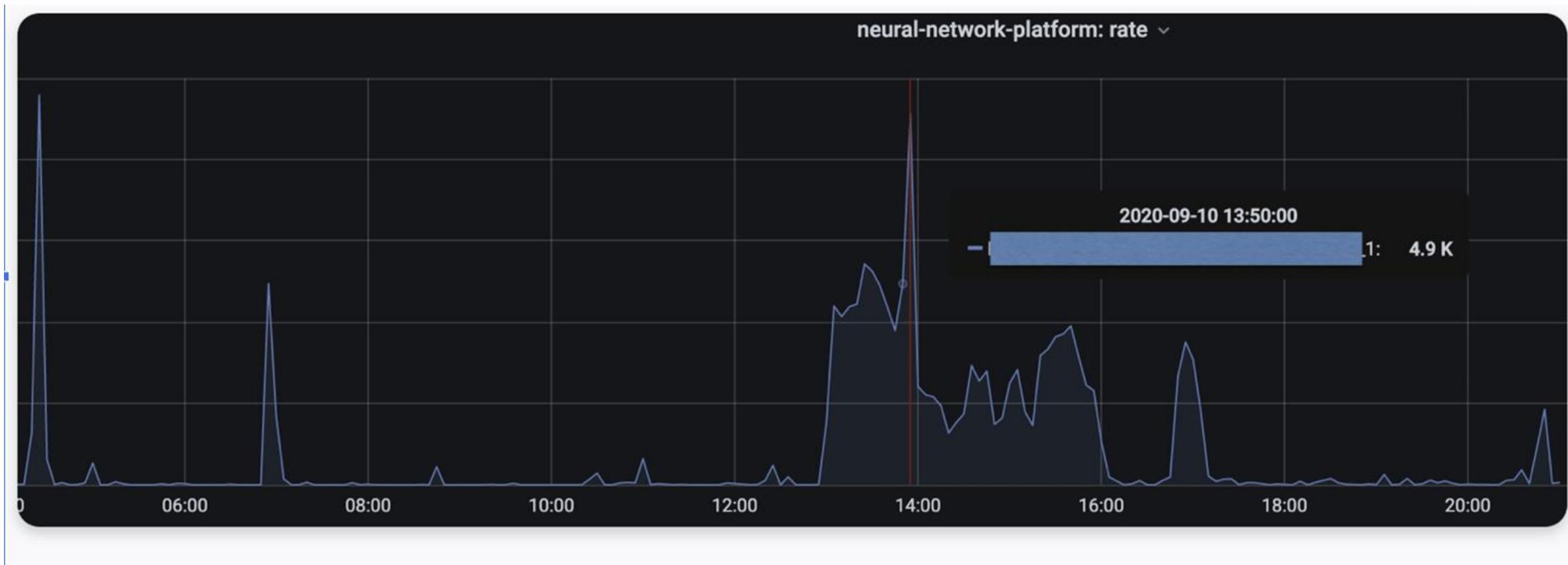
The screenshot shows a web-based interface for managing MLflow models. At the top left is the 'mlflow' logo. Below it, the navigation bar includes 'Registered Models > [REDACTED] > Version 1'. The main content area displays the following information:

- Registered At:** 2020-08-27 15:17:56
- Last Modified:** 2020-08-27 15:38:36
- Creator:** [REDACTED]
- Source Run:** Run b75e609247d1429990b1f0e1a7fd3e61
- Stage:** Production

Below this, there is a collapsed section labeled 'Description' with a link to 'https://stash.odkl.ru/projects>NNP/repos/dvc-experiments/browse/experiments'. The entire screenshot is framed by a light gray border.

https://youtu.be/6ETZn7gc_j8

RPS B GRAFANA



https://youtu.be/6ETZn7gc_j8

MLFLOW MODEL REGISTRY

The screenshot shows a web-based interface for managing MLflow models. At the top left is the 'mlflow' logo. Below it, the navigation bar includes 'Registered Models > [REDACTED] > Version 1'. The main content area displays the following information:

- Registered At:** 2020-08-27 15:17:56
- Last Modified:** 2020-08-27 15:38:36
- Creator:** [REDACTED]
- Source Run:** Run b75e609247d1429990b1f0e1a7fd3e61
- Stage:** Production

Below this, there is a collapsed section labeled 'Description' with a link to 'https://stash.odkl.ru/projects>NNP/repos/dvc-experiments/browse/experiments'. The entire screenshot is framed by a light gray border.

https://youtu.be/6ETZn7gc_j8

Попали в GIT

NN Platform / dvc-experiments

Source

image_classificat... ⚙️ ... | dvc-experiments / experiments / [REDACTED]s_image_classification /

Source	Description
..	
config	
dvc_files	
env	
.bumpversion.cfg	[REDACTED]age_classification: 0.0.0 -> 0.0.1
mlflow_link.md	SPA-9655: [REDACTED]image_classification
pipeline_release_info.yaml	SPA-9655: [REDACTED]image_classification

https://youtu.be/6ETZn7gc_j8

JIRA

SPA / SPA-9610 Модель на помещении: внутри или снаружи / SPA-9655

 Обучить модель внутри/снаружи

Редактировать Кomentарий Назначить Еще... Return to development Release Экспорт

▼ Детали задачи

Тип:	Подзадача	Статус:	РЕШЕННЫЕ
Приоритет:	Основной	Решение:	Готово
Компоненты:	Рябина		
Метки:	Не заполнено		
Project affected:	Анти-спам		
Phase:	Modeling		

▼ Описание

Обучить модель в облаке и зарелизить

▼ Вложенные файлы

Перенесите файлы, чтобы прикрепить, или обзор.

...
▼ Активность

Все Комментарии Журнал работ История изменений Активность

Для этого запроса еще нет комментариев.

▼ Даты

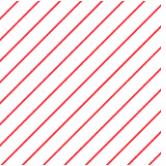
Создано:	27.08.2020 12:06
Обновлено:	27.08.2020 15:18
Дата решения:	27.08.2020 15:18

▼ Разработка

1 ветка	Обновленный 27.08.2020 15:12
5 коммитов	Последний 27.08.2020 15:15
1 pull-запрос	Обновленный 27.08.2020 15:15

Создать ветку

https://youtu.be/6ETZn7gc_j8



Содержание занятия

- Зачем все это надо?
- Как организовать воспроизводимое обучение
 - Почему не стоит хранить большие файлы в git
 - Где же их хранить? (рассмотрим s3)
- dvc
- mlflow

Доклад про dvc и mlflow в OK.ru



SmartData

Михаил Марюфич

Mail.Ru Group

CI/CD для ML-моделей и
датасетов

