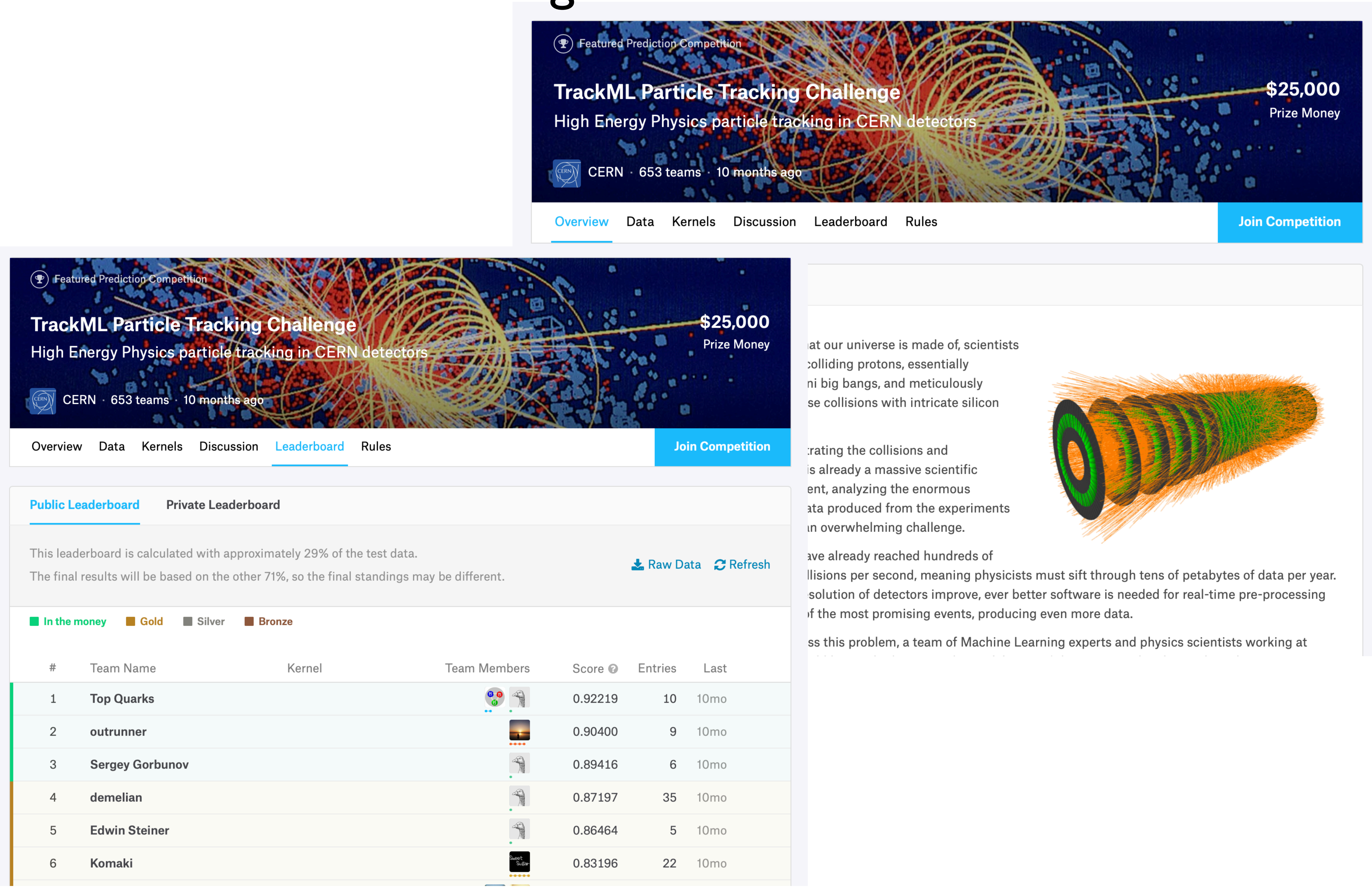
Reproducible Benchmarks for Data Analysis

Heiko Mueller, Irina Espejo,
Kyle Cranmer & **Sebastian Macaluso**
New York University

Shih-Chieh Hsu, Aaron Maritz, Ajay Rawat
University of Washington

IRIS-HEP topical meeting
September 25, 2019

Benchmarks & Challenges:



Featured Prediction Competition

TrackML Particle Tracking Challenge

High Energy Physics particle tracking in CERN detectors

CERN · 653 teams · 10 months ago

$25,000 Prize Money

at our universe is made of, scientists colliding protons, essentially ni big bangs, and meticulously se collisions with intricate silicon

rating the collisions and is already a massive scientific ent, analyzing the enormous ata produced from the experiments n overwhelming challenge.

ave already reached hundreds of llisions per second, meaning physicists must sift through tens of petabytes of data per year. solution of detectors improve, ever better software is needed for real-time pre-processing f the most promising events, producing even more data.

ss this problem, a team of Machine Learning experts and physics scientists working at

Featured Prediction Competition

TrackML Particle Tracking Challenge

High Energy Physics particle tracking in CERN detectors

CERN · 653 teams · 10 months ago

$25,000 Prize Money

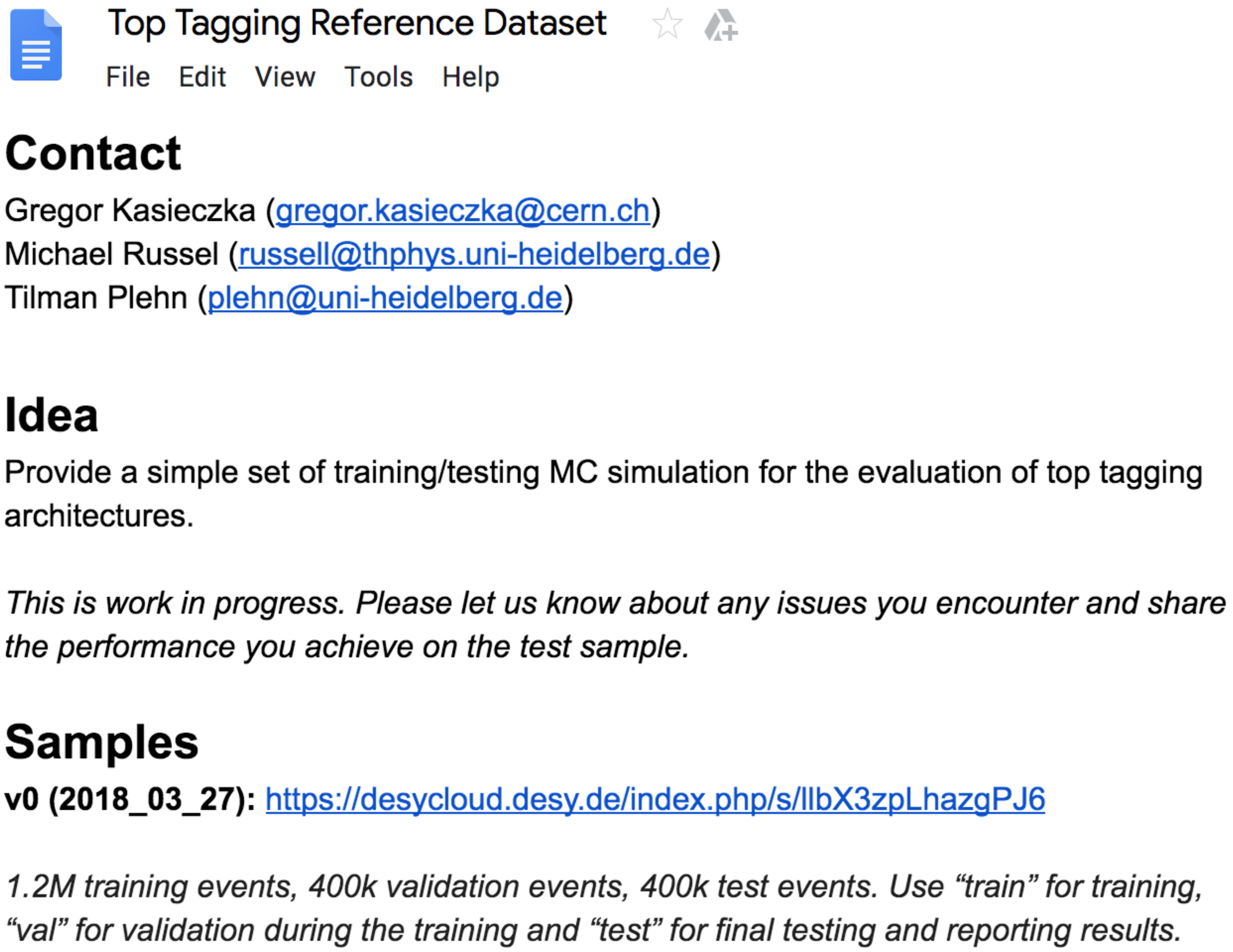
This leaderboard is calculated with approximately 29% of the test data.

The final results will be based on the other 71%, so the final standings may be different.

🟩 In the money 🟧 Gold ⬜ Silver 🟫 Bronze

#	Team Name	Kernel	Team Members	Score	Entries	Last
1	Top Quarks			0.92219	10	10mo
2	outrunner			0.90400	9	10mo
3	Sergey Gorbunov			0.89416	6	10mo
4	demelian			0.87197	35	10mo
5	Edwin Steiner			0.86464	5	10mo
6	Komaki			0.83196	22	10mo

Top-tagging benchmark example



Contact

Gregor Kasieczka (gregor.kasieczka@cern.ch)
Michael Russel (russell@thphys.uni-heidelberg.de)
Tilman Plehn (plehn@uni-heidelberg.de)

Idea

Provide a simple set of training/testing MC simulation for the evaluation of top tagging architectures.

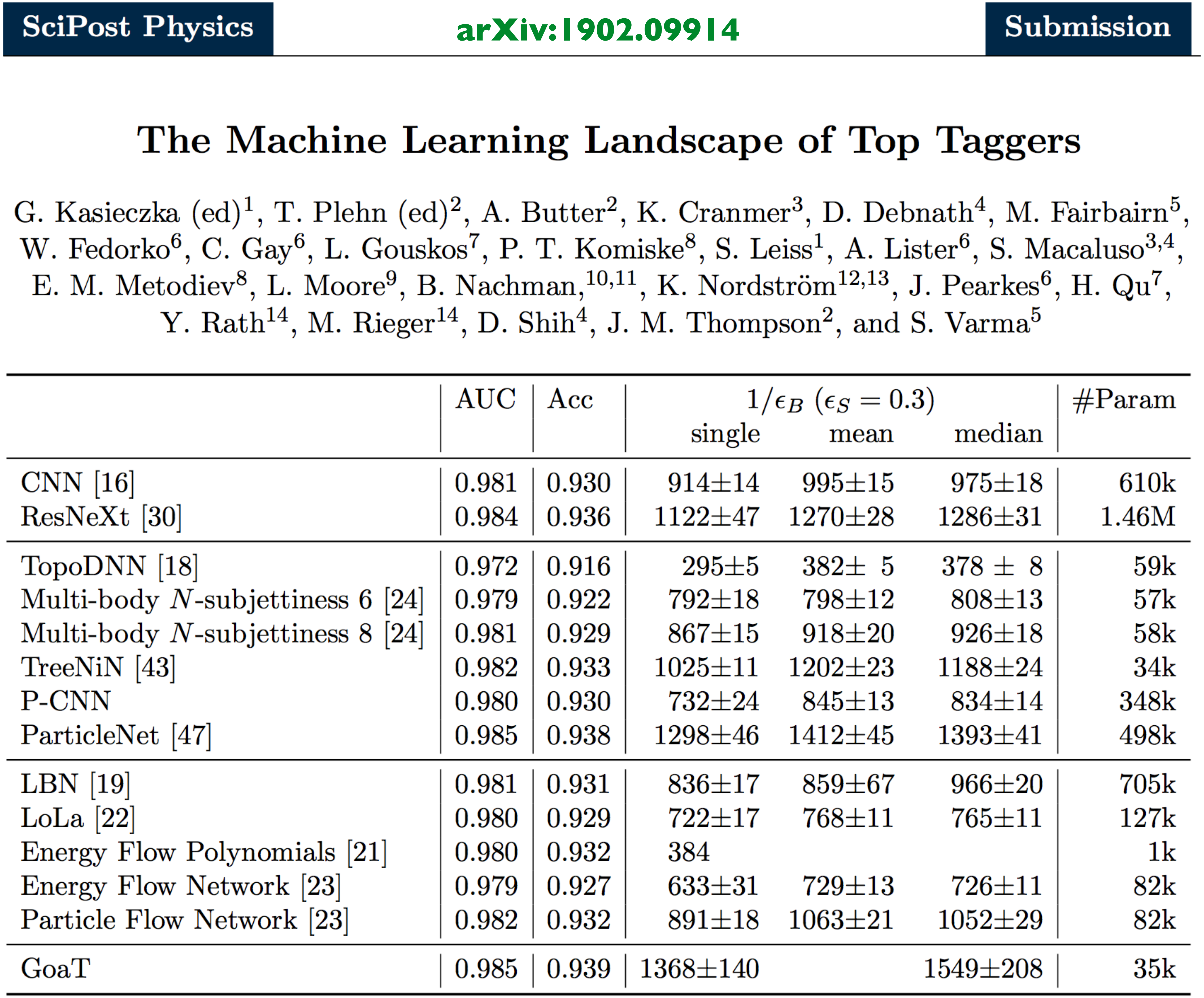
This is work in progress. Please let us know about any issues you encounter and share the performance you achieve on the test sample.

Samples

v0 (2018_03_27): https://desycloud.desy.de/index.php/s/llbX3zpLhazgPJ6

1.2M training events, 400k validation events, 400k test events. Use "train" for training, "val" for validation during the training and "test" for final testing and reporting results.

arXiv:1902.09914



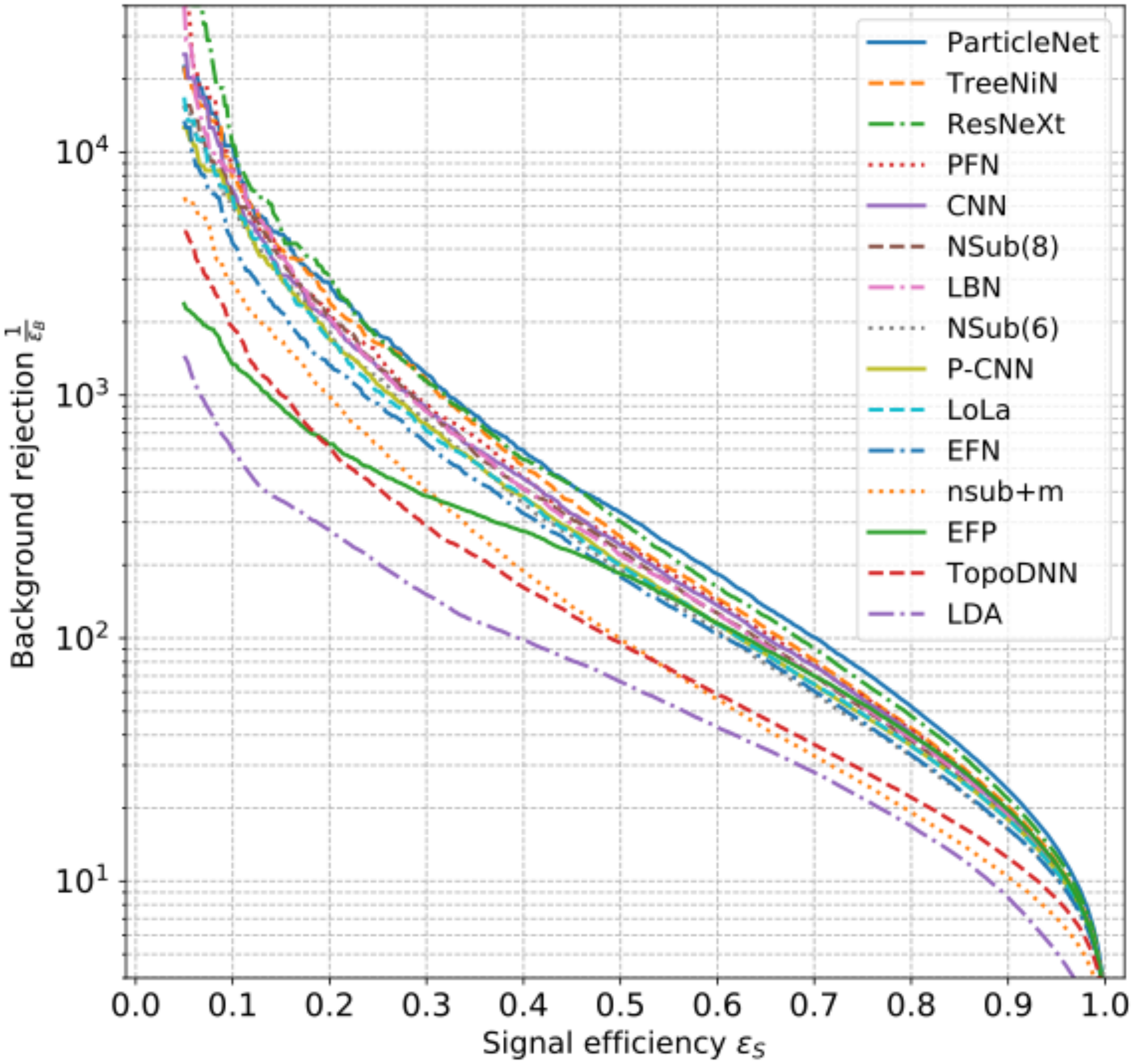
SciPost Physics Submission

The Machine Learning Landscape of Top Taggers

G. Kasieczka (ed)[1], T. Plehn (ed)[2], A. Butter[2], K. Cranmer[3], D. Debnath[4], M. Fairbairn[5], W. Fedorko[6], C. Gay[6], L. Gouskos[7], P. T. Komiske[8], S. Leiss[1], A. Lister[6], S. Macaluso[3,4], E. M. Metodiev[8], L. Moore[9], B. Nachman,[10,11], K. Nordström[12,13], J. Pearkes[6], H. Qu[7], Y. Rath[14], M. Rieger[14], D. Shih[4], J. M. Thompson[2], and S. Varma[5]

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$)			#Param
			single	mean	median	
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [30]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k

Algorithms ROC curves

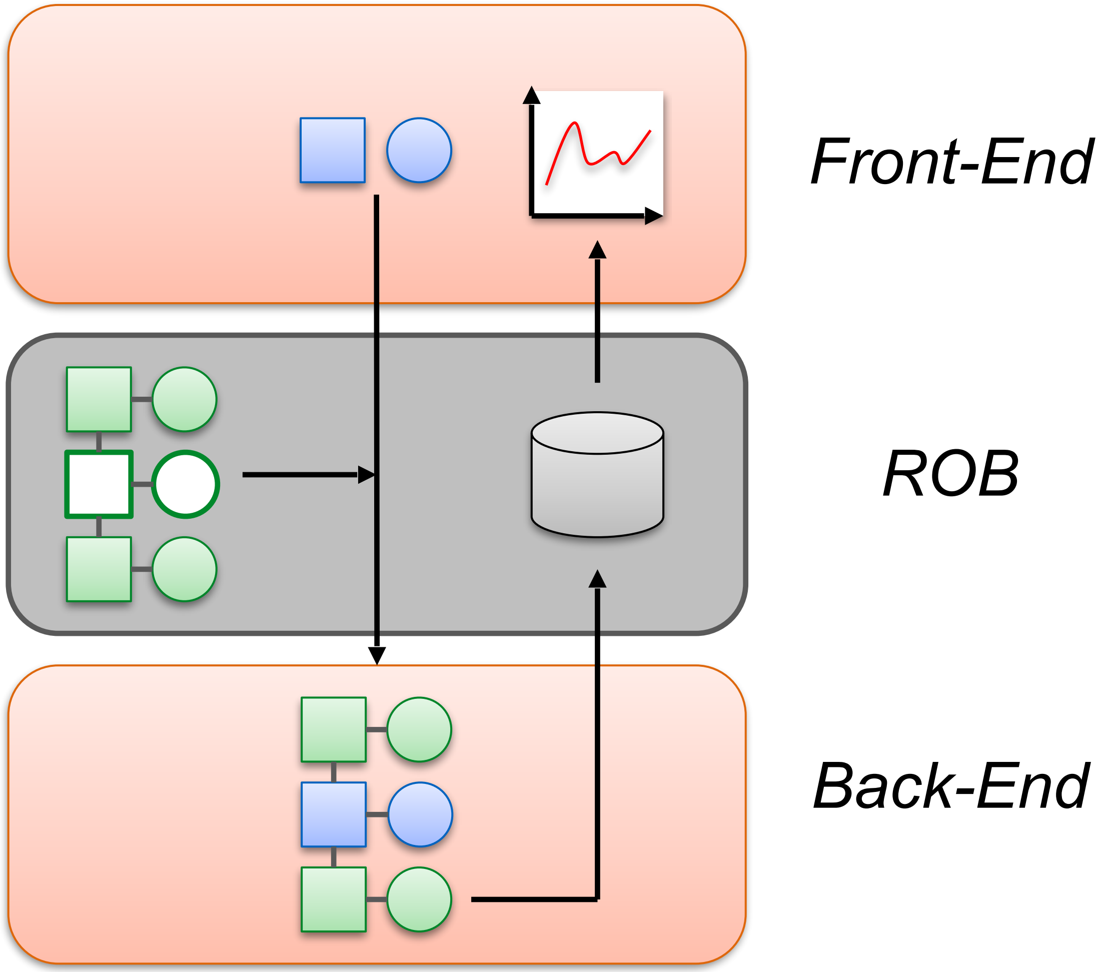


Reproducible Open Benchmarks for Data Analysis Platform (ROB)

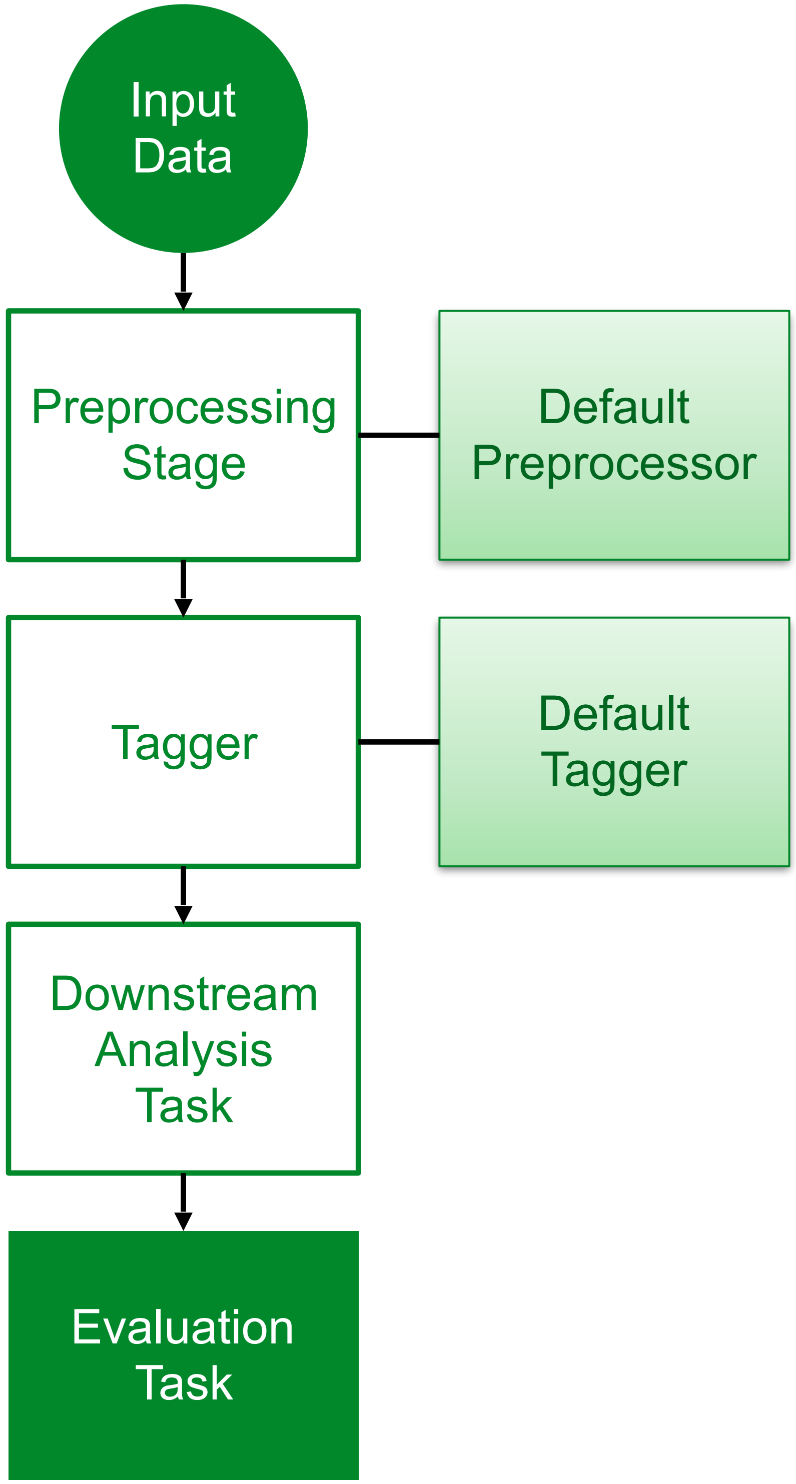
Exploratory work for enabling such community benchmarks.

Components and Actors in ROB

1. Benchmark workflow defined by **coordinator** along with input data.
2. **Users** provide code (e.g. docker containers) that satisfy workflow stages, input parameters, and input data (file upload).
3. **Back-end** processes workflows and evaluates metrics (powered for example by REANA).
4. **Front-end** to collect input and display results.



Benchmark Workflow Example



Input Data

Preprocessing Stage

Default Preprocessor

Tagger

Default Tagger

Downstream Analysis Task

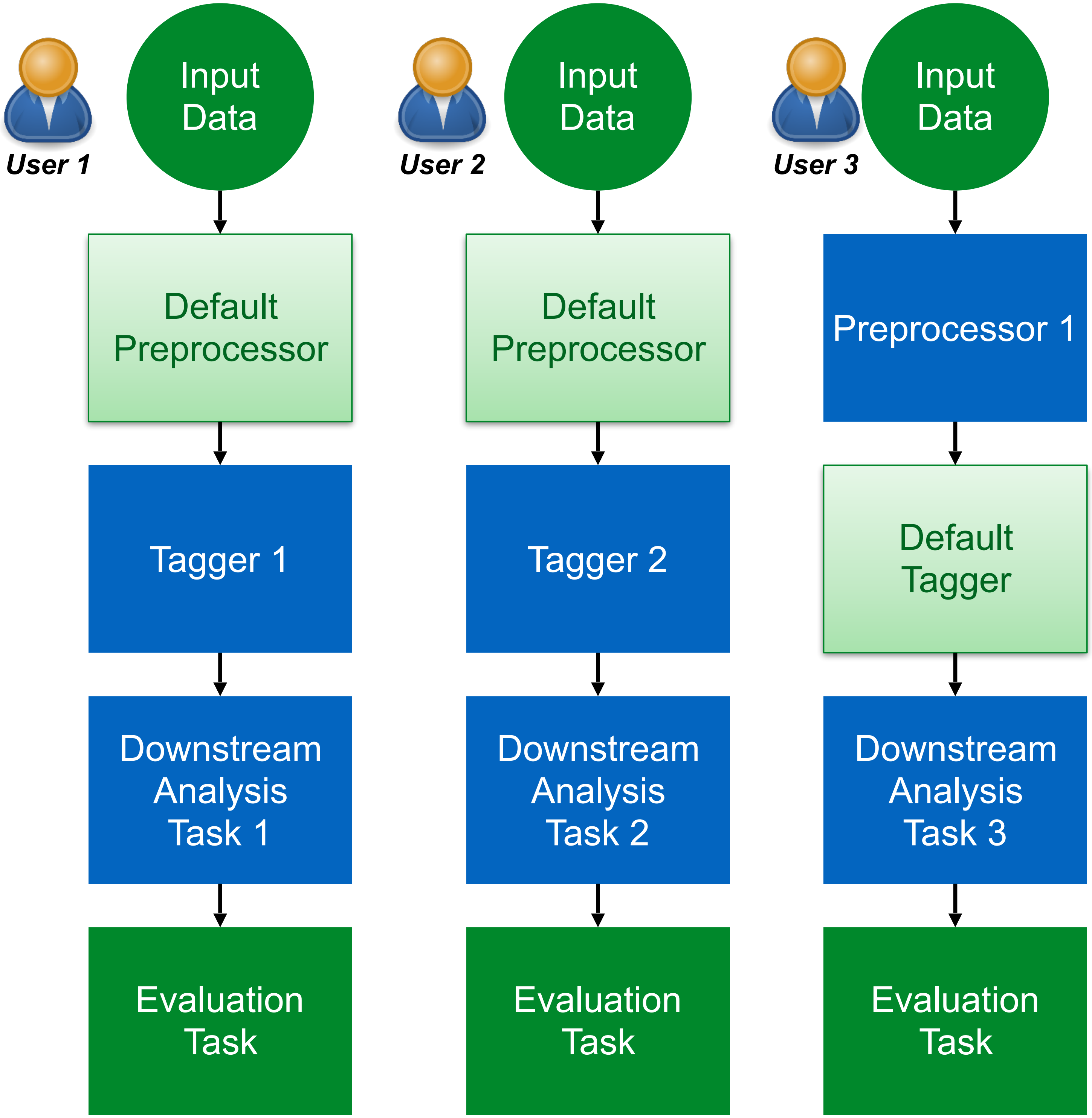
Evaluation Task

Workflow Templates

Coordinator defines structure of the workflow:

- ● Static input data
- ■ Implementation for static workflow stages
- ▨ Default implementation for variable workflow stages
- □ Variable (user-provided) workflow stages
- ○ User-provided input data

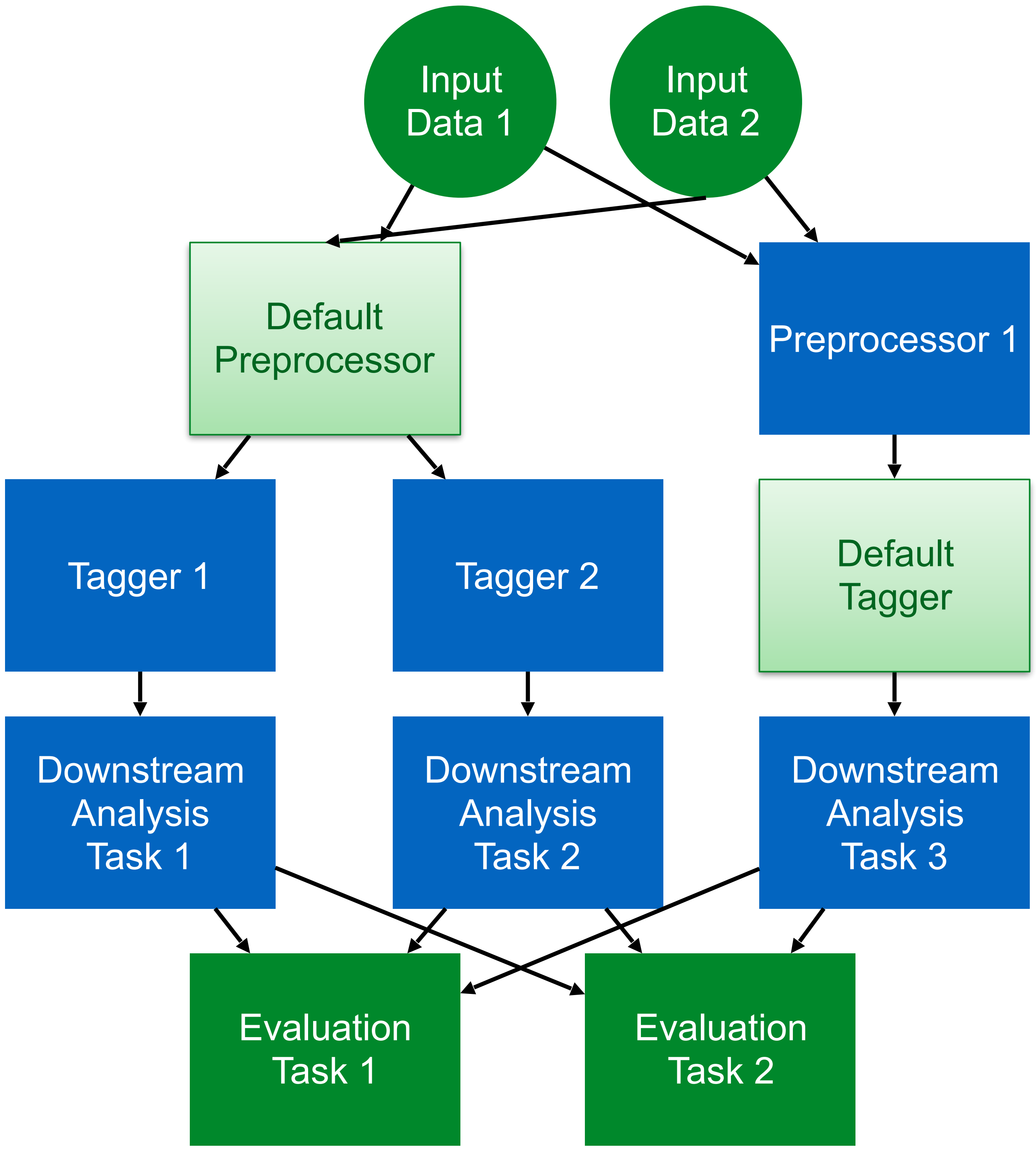
Benchmark Workflow Example (cont.)



Benchmark Participants

Users create different instances of the workflow by providing **implementation for variable workflow stages** (and variable input data).

Benchmark Workflow Example (cont.)



Idea

Create repository of contributed implementations and data.

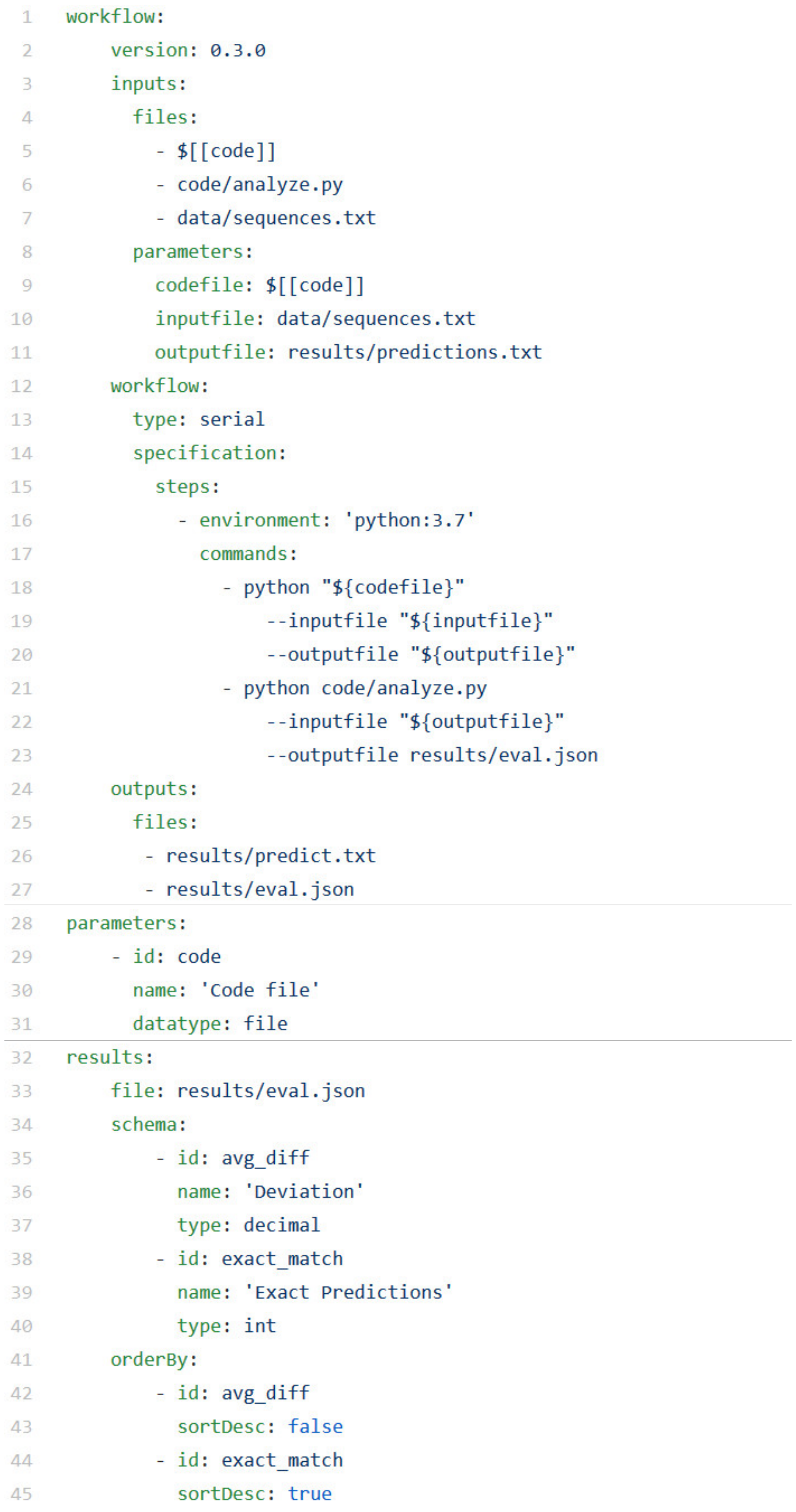
Compossible workflows support contributions used for multiple workflows.

Workflow Templates

Components of Workflow Templates

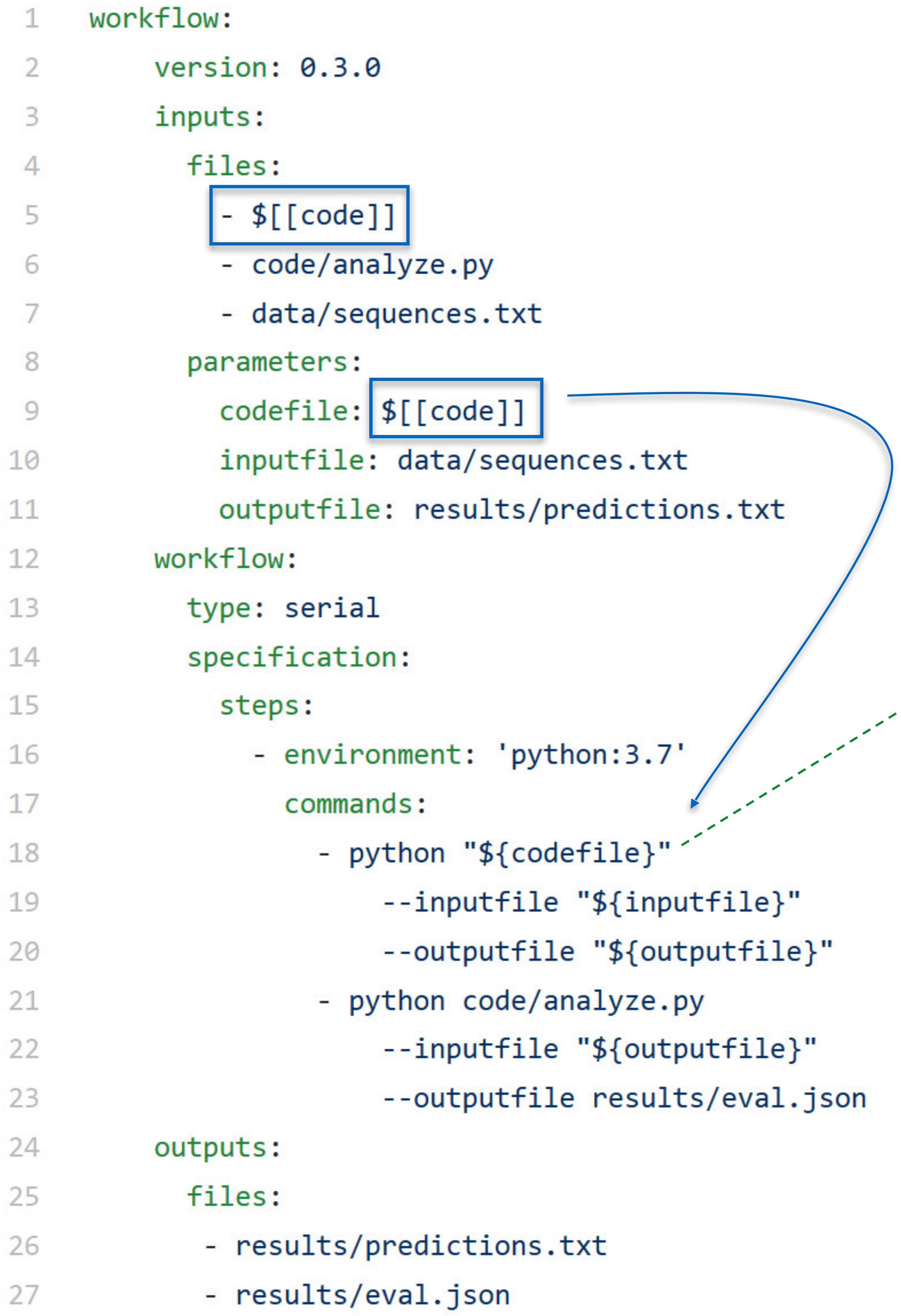
1. Workflow specification (e.g. REANA serial workflow) with optional references to template parameters.
2. Declaration of template parameters (used by front-end for data input)
3. Specification of result schema to generate '*leader board*'.

Reproducible research data analysis platform

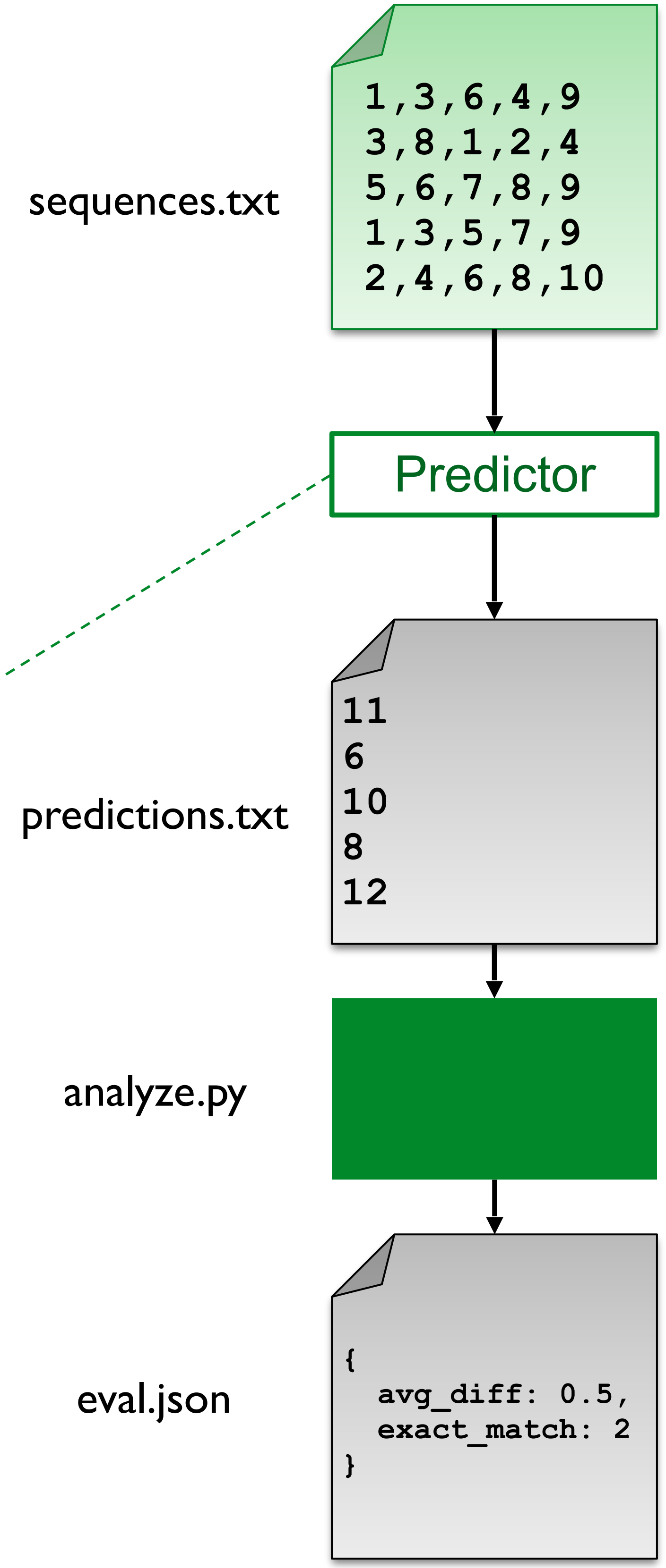


```
workflow:
    version: 0.3.0
    inputs:
      files:
        - $[[code]]
        - code/analyze.py
        - data/sequences.txt
      parameters:
        codefile: $[[code]]
        inputfile: data/sequences.txt
        outputfile: results/predictions.txt
    workflow:
      type: serial
      specification:
        steps:
          - environment: 'python:3.7'
            commands:
              - python "${codefile}"
                  --inputfile "${inputfile}"
                  --outputfile "${outputfile}"
              - python code/analyze.py
                  --inputfile "${outputfile}"
                  --outputfile results/eval.json
    outputs:
      files:
       - results/predict.txt
       - results/eval.json
parameters:
    - id: code
      name: 'Code file'
      datatype: file
results:
    file: results/eval.json
    schema:
        - id: avg_diff
          name: 'Deviation'
          type: decimal
        - id: exact_match
          name: 'Exact Predictions'
          type: int
    orderBy:
        - id: avg_diff
          sortDesc: false
        - id: exact_match
          sortDesc: true
```

Workflow Templates (cont.)



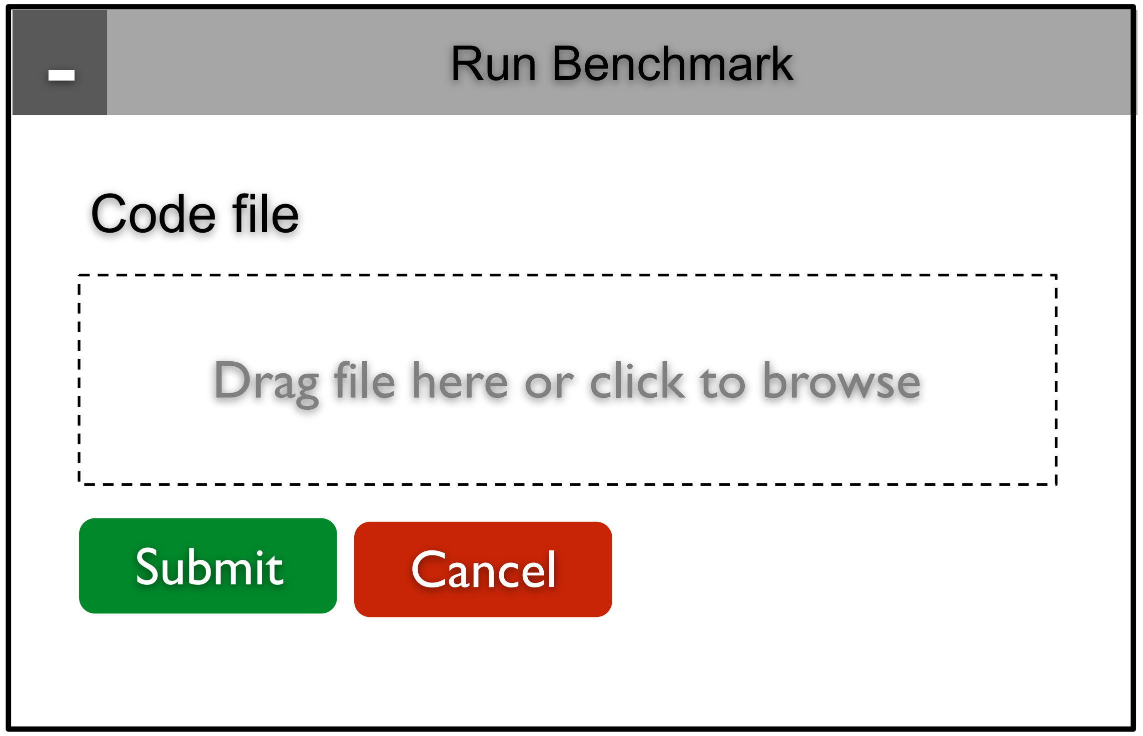
```yaml
workflow:
    version: 0.3.0
    inputs:
      files:
        - $[[code]]
        - code/analyze.py
        - data/sequences.txt
      parameters:
        codefile: $[[code]]
        inputfile: data/sequences.txt
        outputfile: results/predictions.txt
    workflow:
      type: serial
      specification:
        steps:
          - environment: 'python:3.7'
            commands:
              - python "${codefile}"
                  --inputfile "${inputfile}"
                  --outputfile "${outputfile}"
              - python code/analyze.py
                  --inputfile "${outputfile}"
                  --outputfile results/eval.json
    outputs:
      files:
       - results/predictions.txt
       - results/eval.json
```

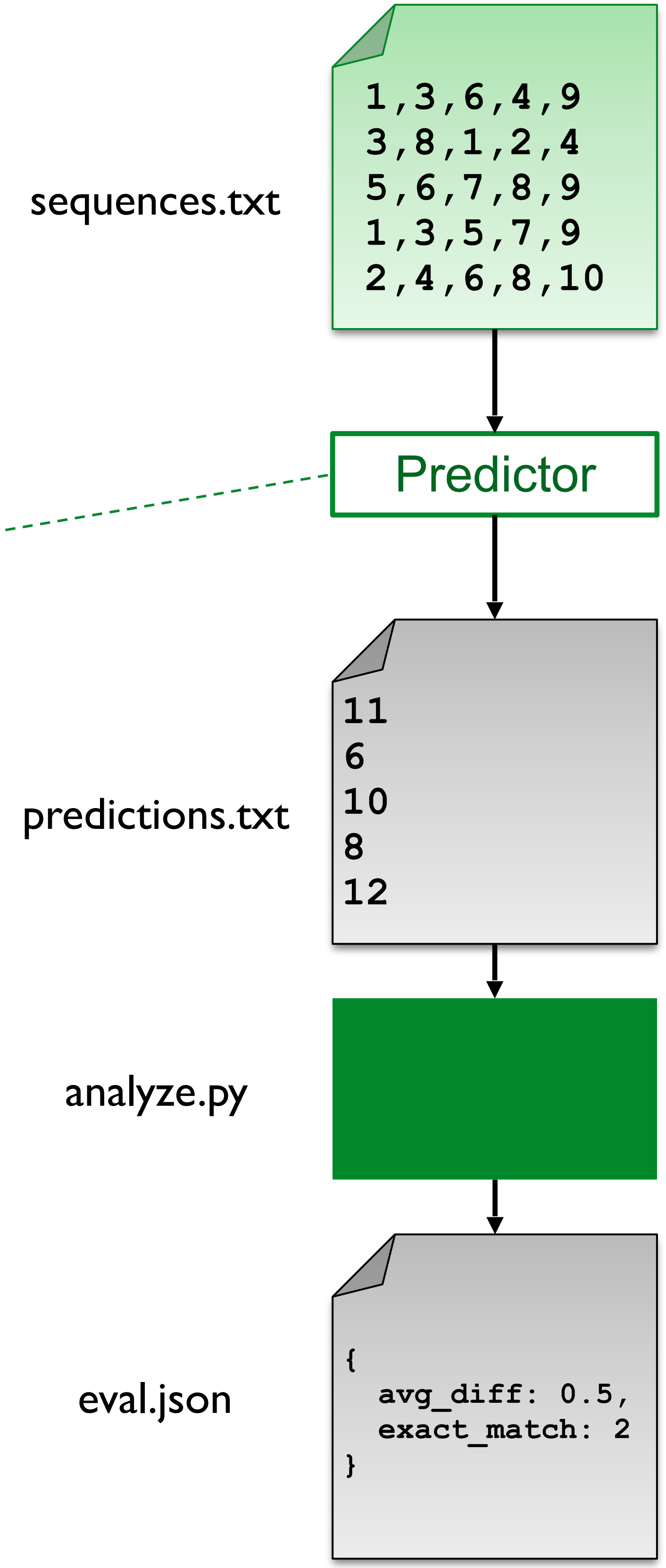


Workflow Templates (cont.)

Render input form from parameter declarations

```
parameters:
    - id: code
      name: 'Code file'
      datatype: file
```





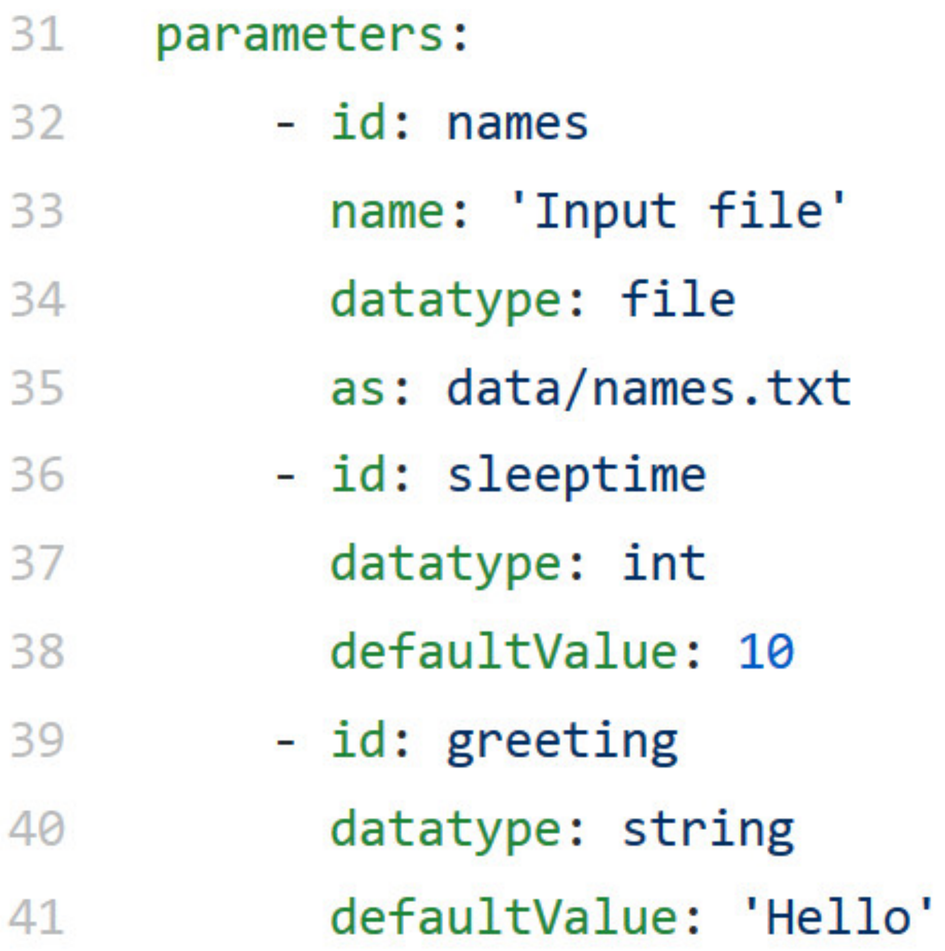
```
1,3,6,4,9
3,8,1,2,4
5,6,7,8,9
1,3,5,7,9
2,4,6,8,10
```

```
11
6
10
8
12
```

```
{
  avg_diff: 0.5,
  exact_match: 2
}
```

Workflow Templates (cont.)

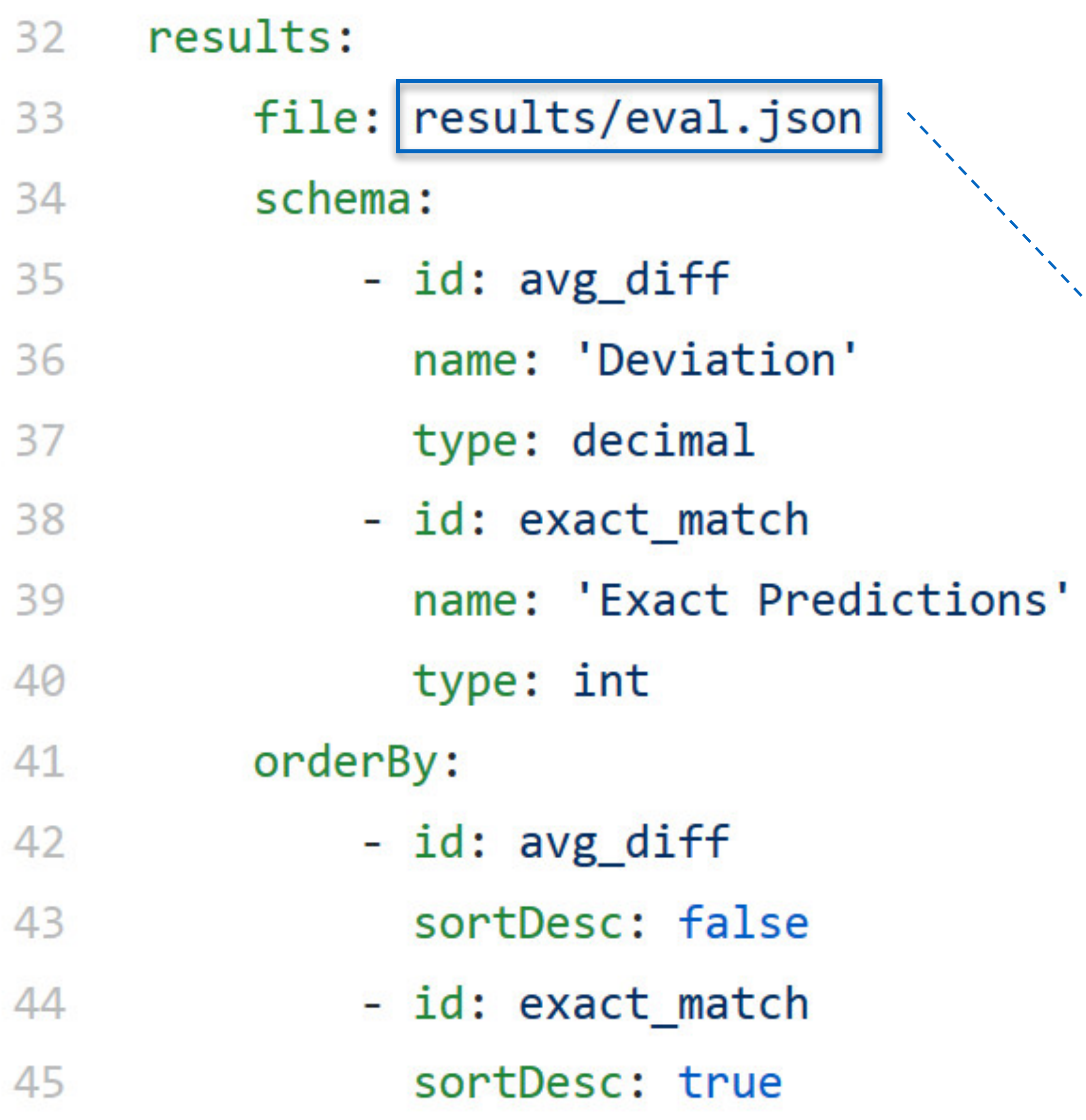
Parameters for 'Hello World'



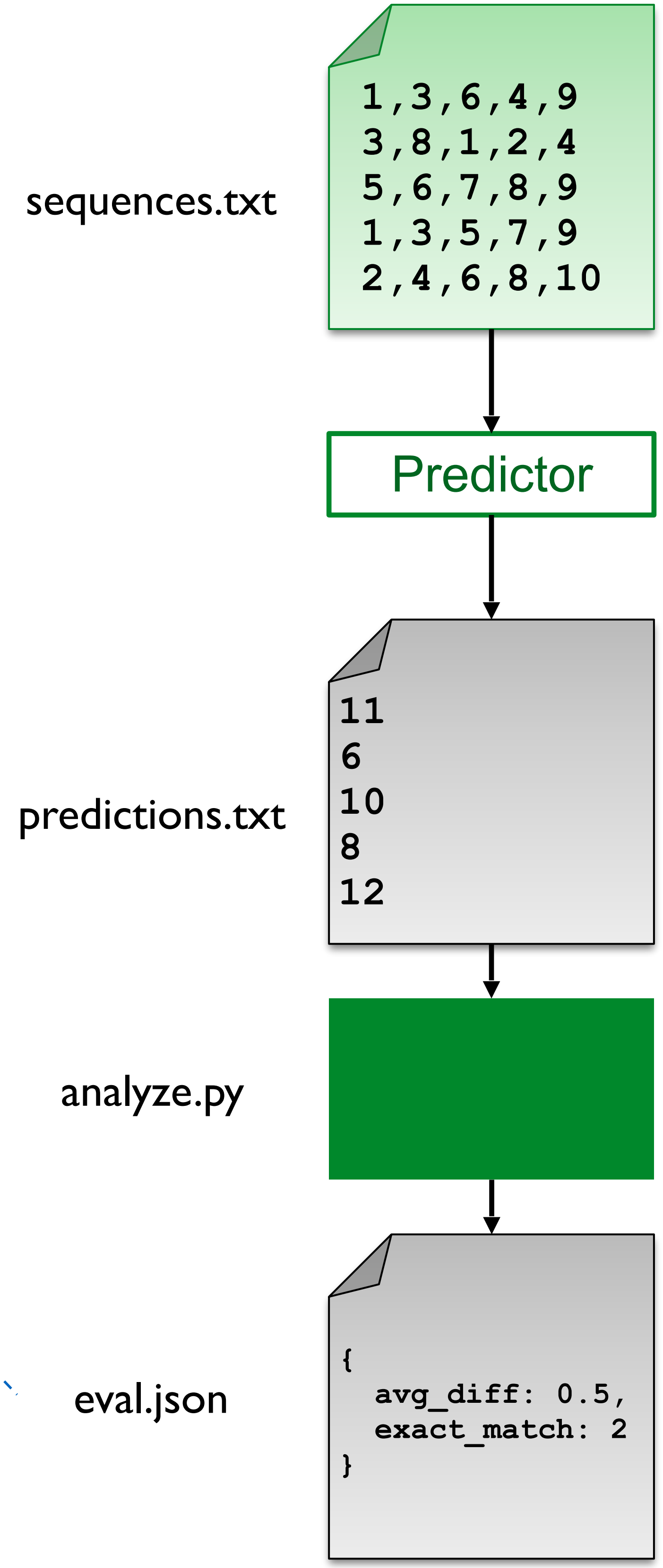
```
parameters:
    - id: names
      name: 'Input file'
      datatype: file
      as: data/names.txt
    - id: sleeptime
      datatype: int
      defaultValue: 10
    - id: greeting
      datatype: string
      defaultValue: 'Hello'
```

Workflow Templates (cont.)

Result schema to store benchmark results in database and to generate ranking



```yaml
results:
    file: results/eval.json
    schema:
        - id: avg_diff
          name: 'Deviation'
          type: decimal
        - id: exact_match
          name: 'Exact Predictions'
          type: int
    orderBy:
        - id: avg_diff
          sortDesc: false
        - id: exact_match
          sortDesc: true
```



sequences.txt

```
1,3,6,4,9
3,8,1,2,4
5,6,7,8,9
1,3,5,7,9
2,4,6,8,10
```

Predictor

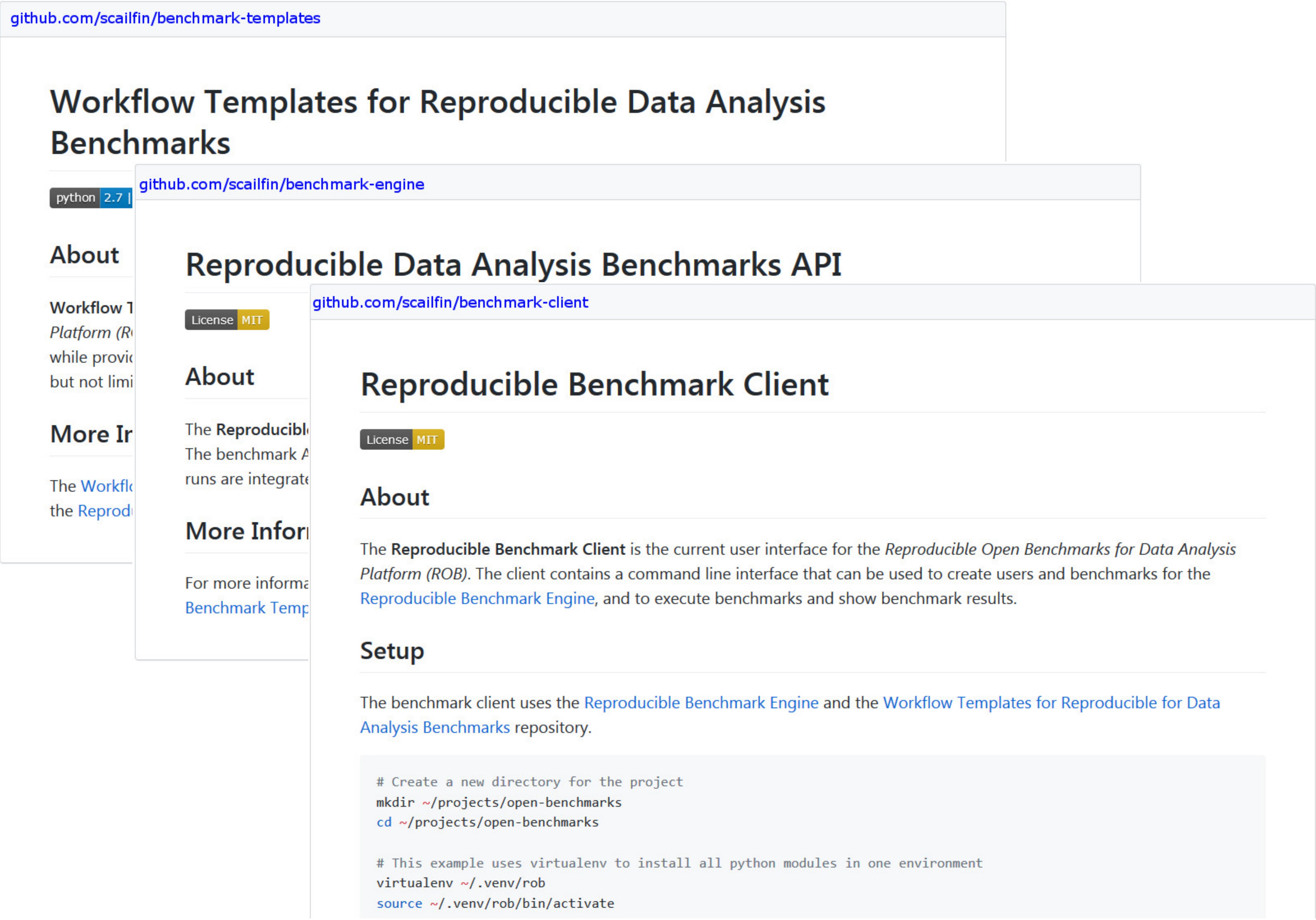
predictions.txt

```
11
6
10
8
12
```

analyze.py

eval.json

```
{
  avg_diff: 0.5,
  exact_match: 2
}
```



github.com/scailfin/benchmark-templates

Workflow Templates for Reproducible Data Analysis Benchmarks

python 2.7 |

About

Workflow T
Platform (R
while provic
but not limi

More Ir

The Workflo
the Reprodu

github.com/scailfin/benchmark-engine

Reproducible Data Analysis Benchmarks API

License MIT

About

The **Reproducibl**
The benchmark A
runs are integrat

More Infor

For more informa
Benchmark Temp

github.com/scailfin/benchmark-client

Reproducible Benchmark Client

License MIT

About

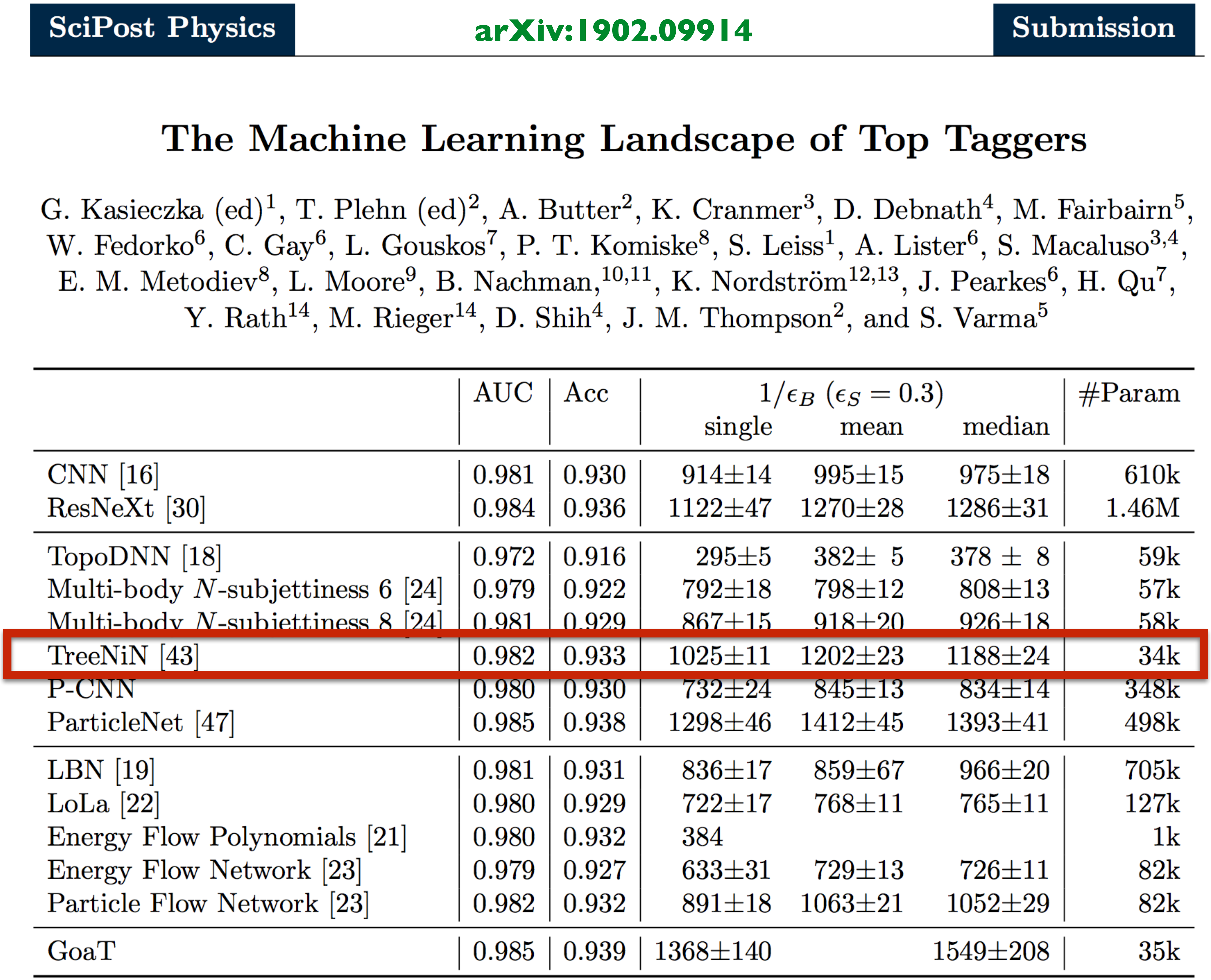
The **Reproducible Benchmark Client** is the current user interface for the *Reproducible Open Benchmarks for Data Analysis Platform (ROB)*. The client contains a command line interface that can be used to create users and benchmarks for the Reproducible Benchmark Engine, and to execute benchmarks and show benchmark results.

Setup

The benchmark client uses the Reproducible Benchmark Engine and the Workflow Templates for Reproducible for Data Analysis Benchmarks repository.

```
# Create a new directory for the project
mkdir ~/projects/open-benchmarks
cd ~/projects/open-benchmarks

# This example uses virtualenv to install all python modules in one environment
virtualenv ~/.venv/rob
source ~/.venv/rob/bin/activate
```



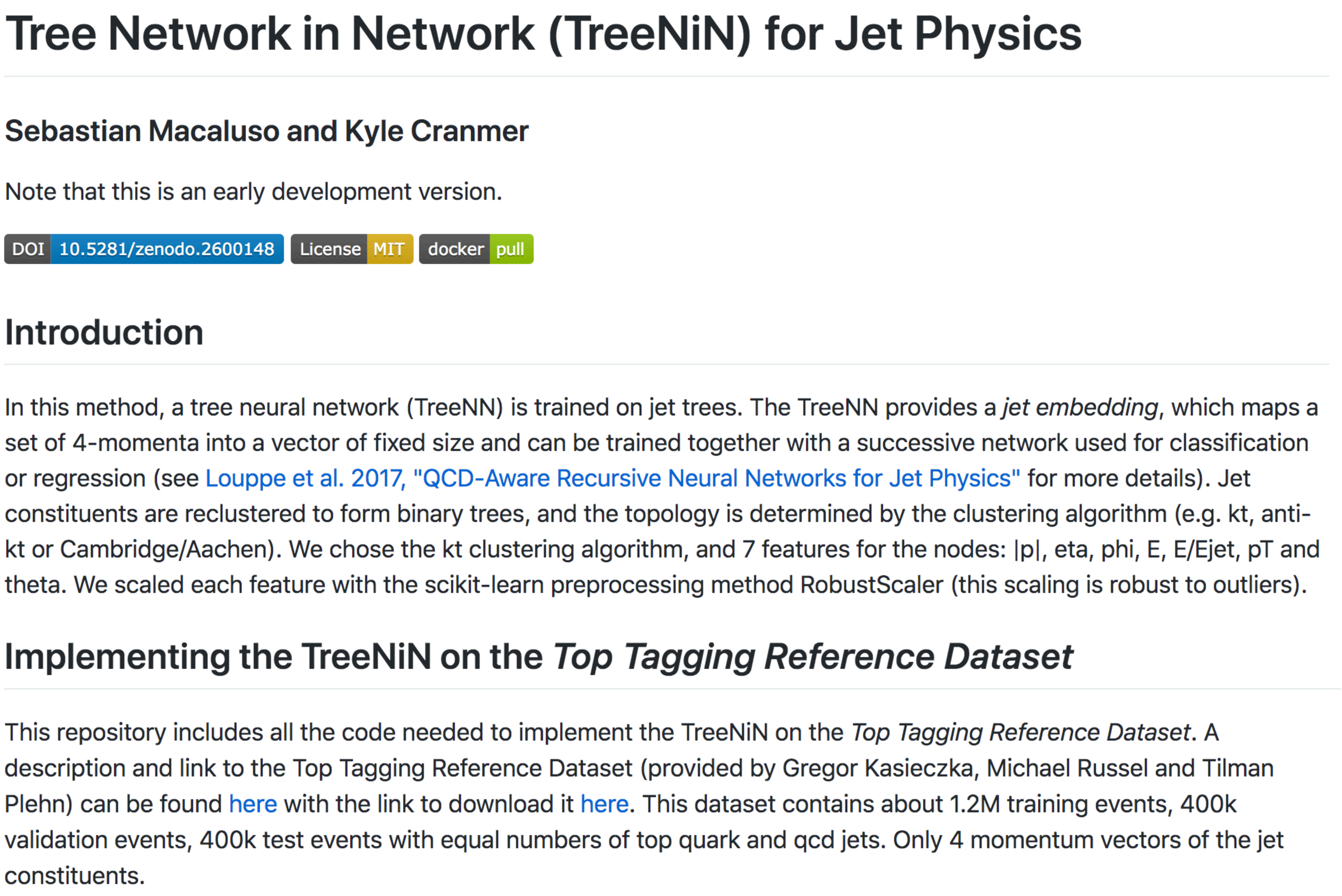
SciPost Physics arXiv:1902.09914 Submission

The Machine Learning Landscape of Top Taggers

G. Kasieczka (ed)[1], T. Plehn (ed)[2], A. Butter[2], K. Cranmer[3], D. Debnath[4], M. Fairbairn[5], W. Fedorko[6], C. Gay[6], L. Gouskos[7], P. T. Komiske[8], S. Leiss[1], A. Lister[6], S. Macaluso[3,4], E. M. Metodiev[8], L. Moore[9], B. Nachman,[10,11], K. Nordström[12,13], J. Pearkes[6], H. Qu[7], Y. Rath[14], M. Rieger[14], D. Shih[4], J. M. Thompson[2], and S. Varma[5]

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$)			#Param
			single	mean	median	
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [30]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k

https://github.com/diana-hep/TreeNiN



Tree Network in Network (TreeNiN) for Jet Physics

Sebastian Macaluso and Kyle Cranmer

Note that this is an early development version.

DOI 10.5281/zenodo.2600148 License MIT docker pull

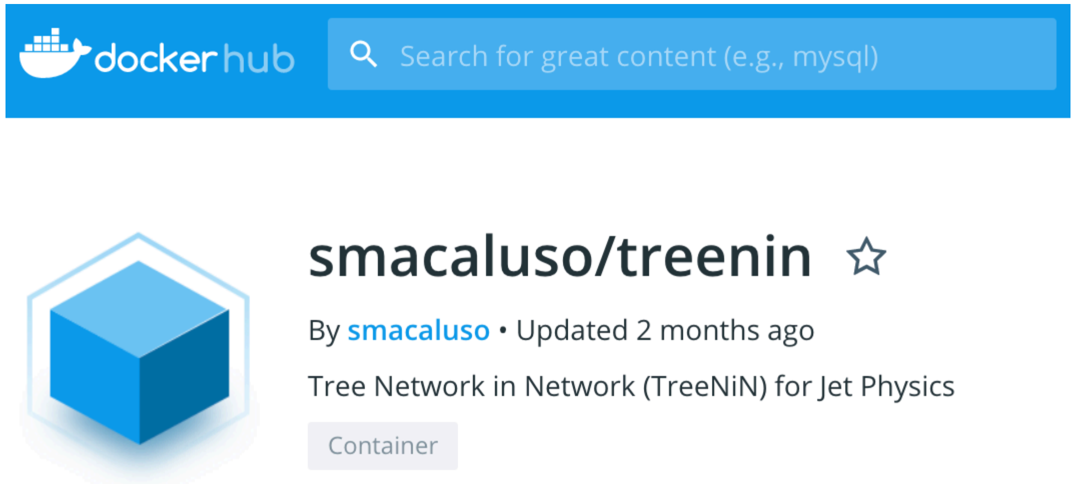
Introduction

In this method, a tree neural network (TreeNN) is trained on jet trees. The TreeNN provides a *jet embedding*, which maps a set of 4-momenta into a vector of fixed size and can be trained together with a successive network used for classification or regression (see Louppe et al. 2017, "QCD-Aware Recursive Neural Networks for Jet Physics" for more details). Jet constituents are reclustered to form binary trees, and the topology is determined by the clustering algorithm (e.g. kt, anti-kt or Cambridge/Aachen). We chose the kt clustering algorithm, and 7 features for the nodes: |p|, eta, phi, E, E/Ejet, pT and theta. We scaled each feature with the scikit-learn preprocessing method RobustScaler (this scaling is robust to outliers).

Implementing the TreeNiN on the *Top Tagging Reference Dataset*

This repository includes all the code needed to implement the TreeNiN on the *Top Tagging Reference Dataset*. A description and link to the Top Tagging Reference Dataset (provided by Gregor Kasieczka, Michael Russel and Tilman Plehn) can be found here with the link to download it here. This dataset contains about 1.2M training events, 400k validation events, 400k test events with equal numbers of top quark and qcd jets. Only 4 momentum vectors of the jet constituents.

TreeNiN implementation as a docker image



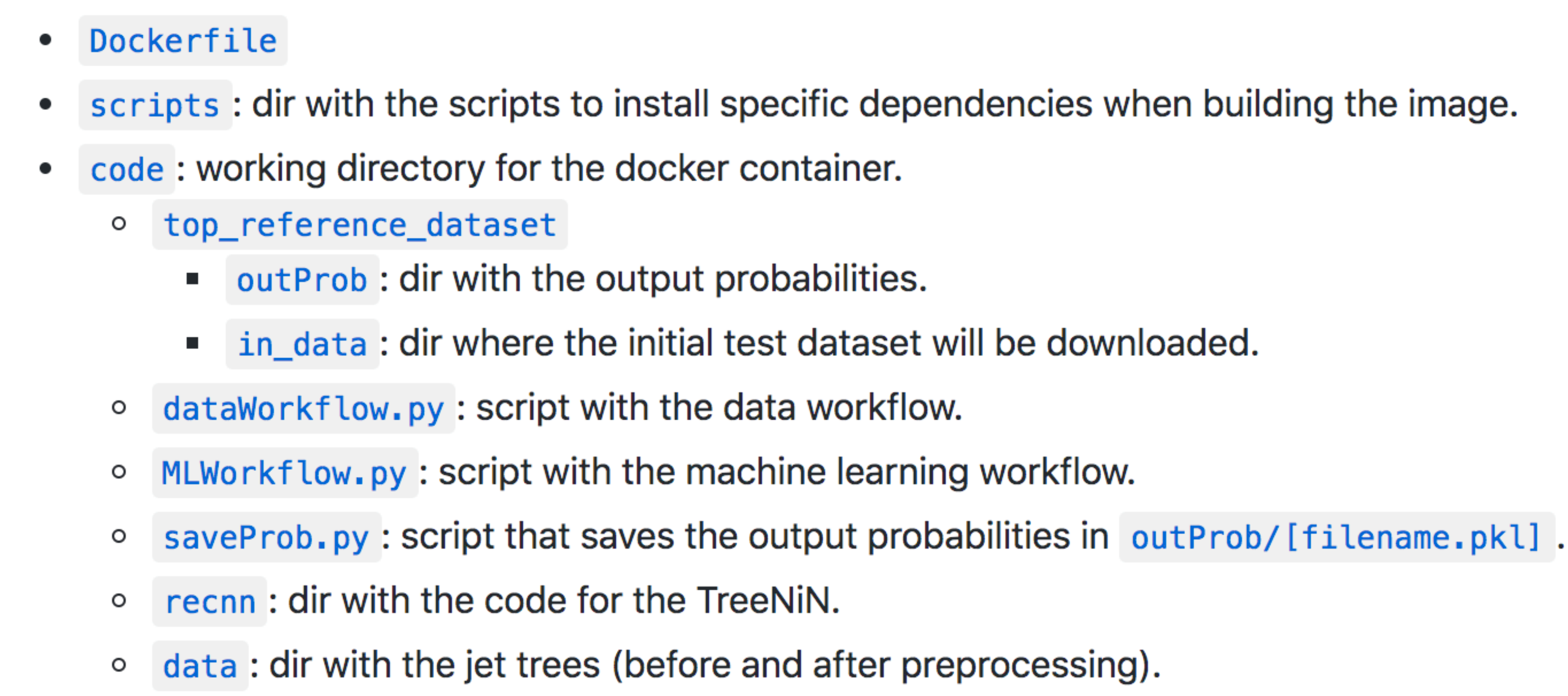
smacaluso/treenin ☆

By smacaluso • Updated 2 months ago

Tree Network in Network (TreeNiN) for Jet Physics

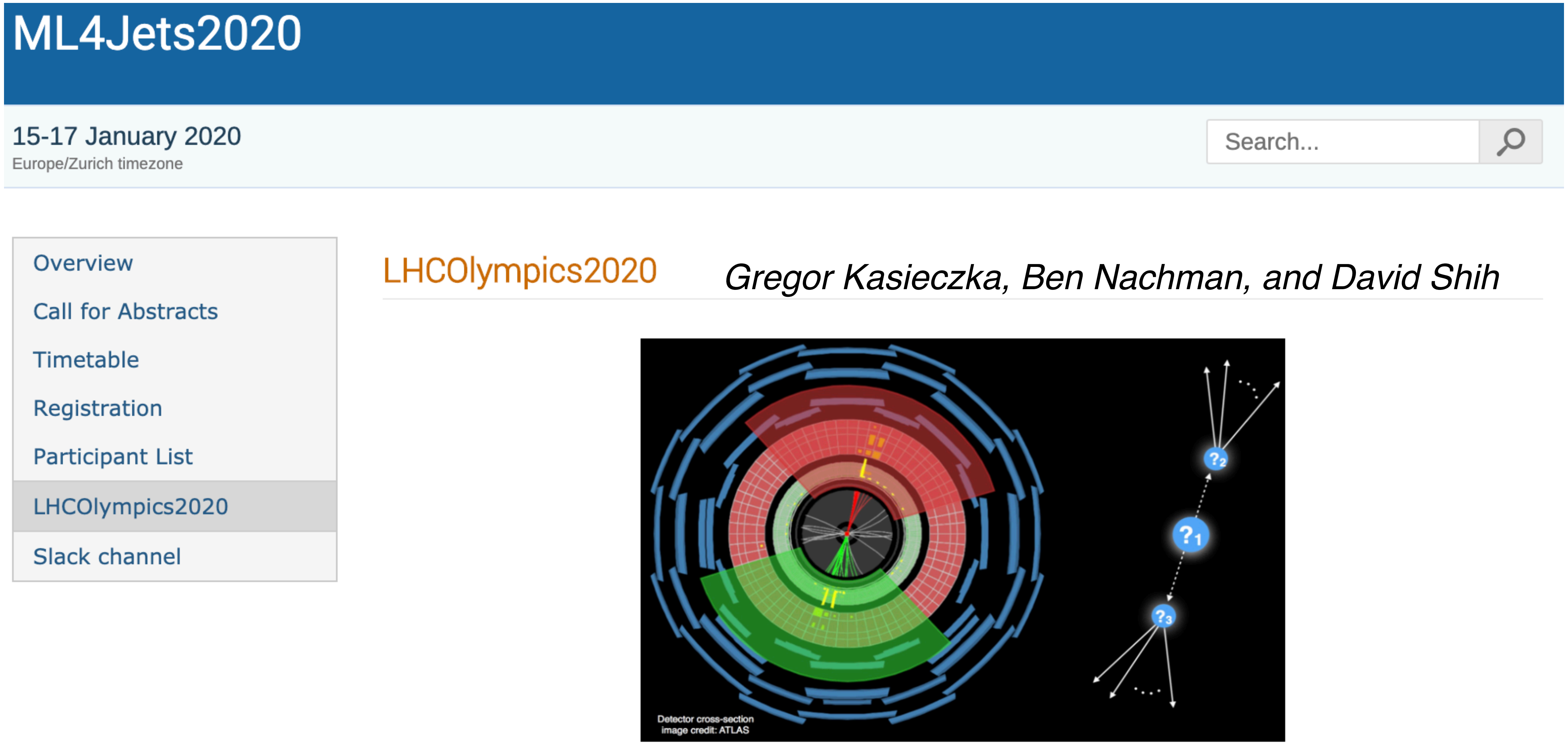
Container

Relevant Structure:



- `Dockerfile`
- `scripts` : dir with the scripts to install specific dependencies when building the image.
- `code` : working directory for the docker container.
  - `top_reference_dataset`
    - `outProb` : dir with the output probabilities.
    - `in_data` : dir where the initial test dataset will be downloaded.
  - `dataWorkflow.py` : script with the data workflow.
  - `MLWorkflow.py` : script with the machine learning workflow.
  - `saveProb.py` : script that saves the output probabilities in `outProb/[filename.pkl]` .
  - `recnn` : dir with the code for the TreeNiN.
  - `data` : dir with the jet trees (before and after preprocessing).

Aiming for demo at ML4Jets 2020 at NYU



ML4Jets2020

15-17 January 2020
Europe/Zurich timezone

- Overview
- Call for Abstracts
- Timetable
- Registration
- Participant List
- LHCOlympics2020
- Slack channel

LHCOlympics2020 *Gregor Kasieczka, Ben Nachman, and David Shih*

Detector cross-section image credit: ATLAS

New Challenge: LHCOlympics2020 focused on searches for BSM physics

- Signal: X to hadrons (dijet events), where X is a new massive particle with an O(TeV) mass.

- Goal: identify BSM physics (yes/no, what mass, what cross-section) in the dataset.

Thanks for your attention!