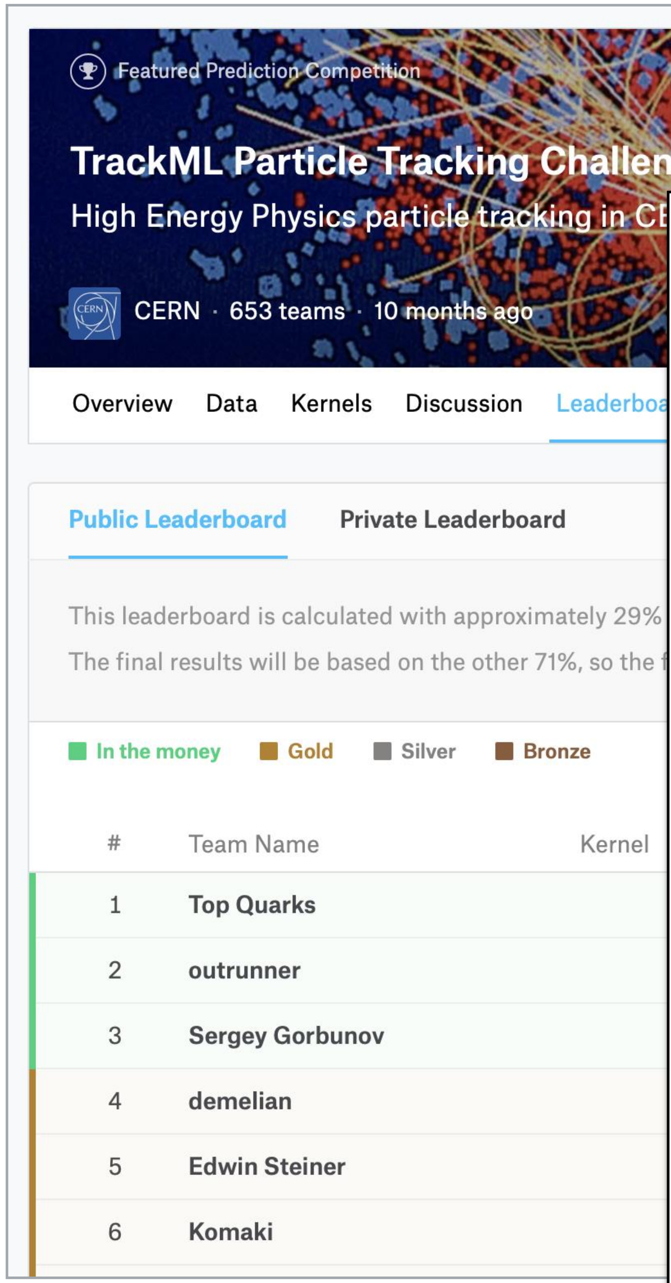
Reproducible Open Benchmarks for Data Analysis

Heiko Mueller, Irina Espejo,
Kyle Cranmer and Sebastian Macaluso

NYU

Benchmarks & Challenges



Featured Prediction Competition

TrackML Particle Tracking Challenge

High Energy Physics particle tracking in CE

$25,000

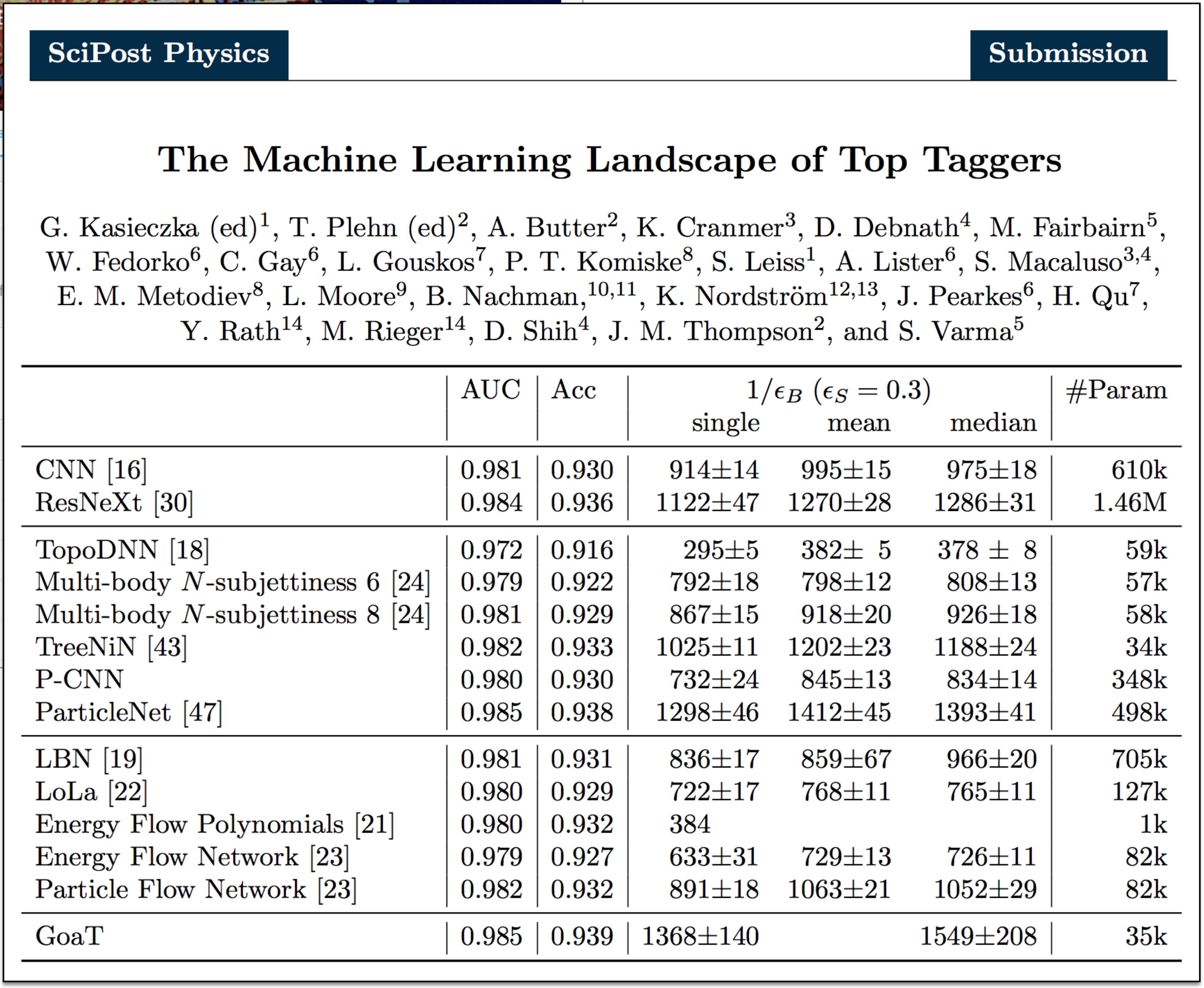
CERN · 653 teams · 10 months ago

This leaderboard is calculated with approximately 29%

The final results will be based on the other 71%, so the f

🟩 In the money 🟫 Gold ⬜ Silver 🟫 Bronze

#	Team Name	Kernel
1	Top Quarks	
2	outrunner	
3	Sergey Gorbunov	
4	demelian	
5	Edwin Steiner	
6	Komaki	



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]

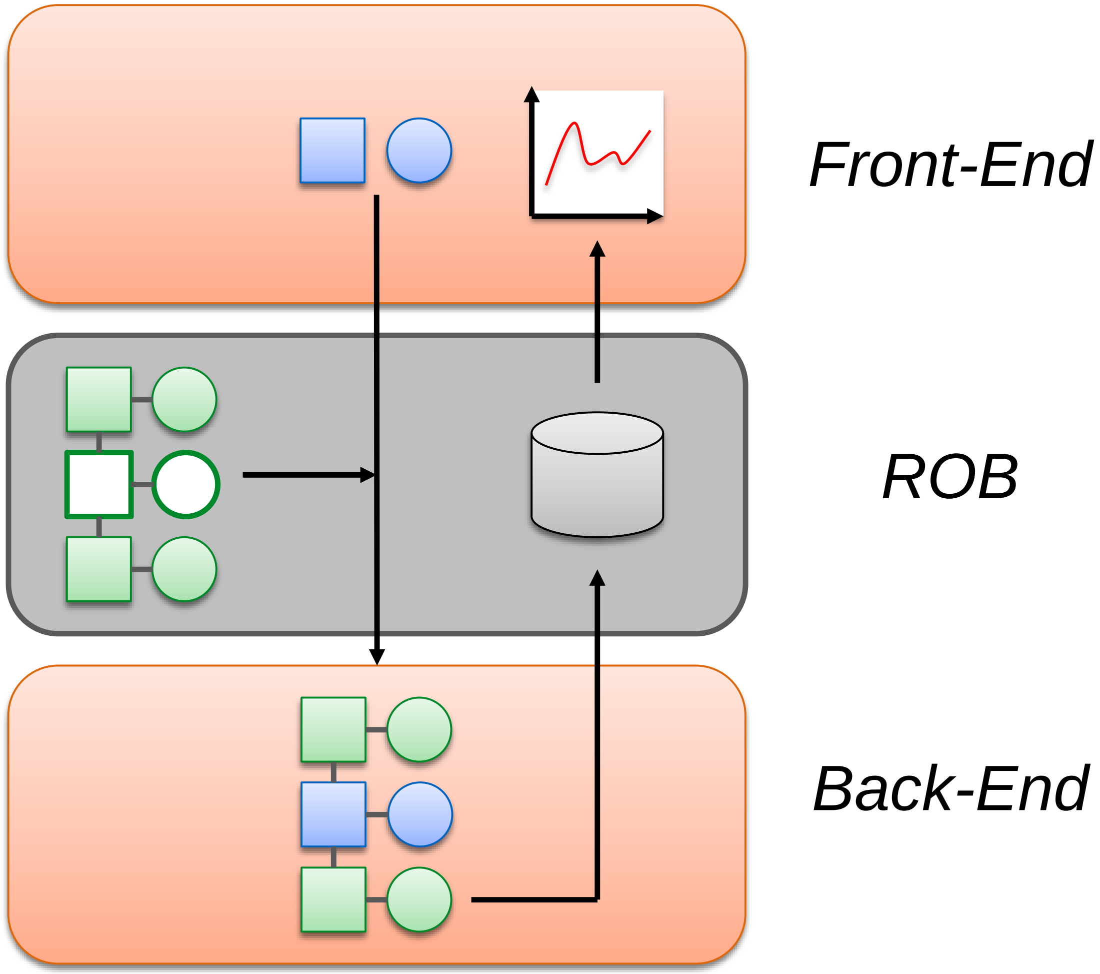
	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$)			#Param
			single	mean	median	
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [30]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body 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

Reproducible Open Benchmarks Platform

Exploratory work for enabling such community benchmarks.

Components and Actors in ROB

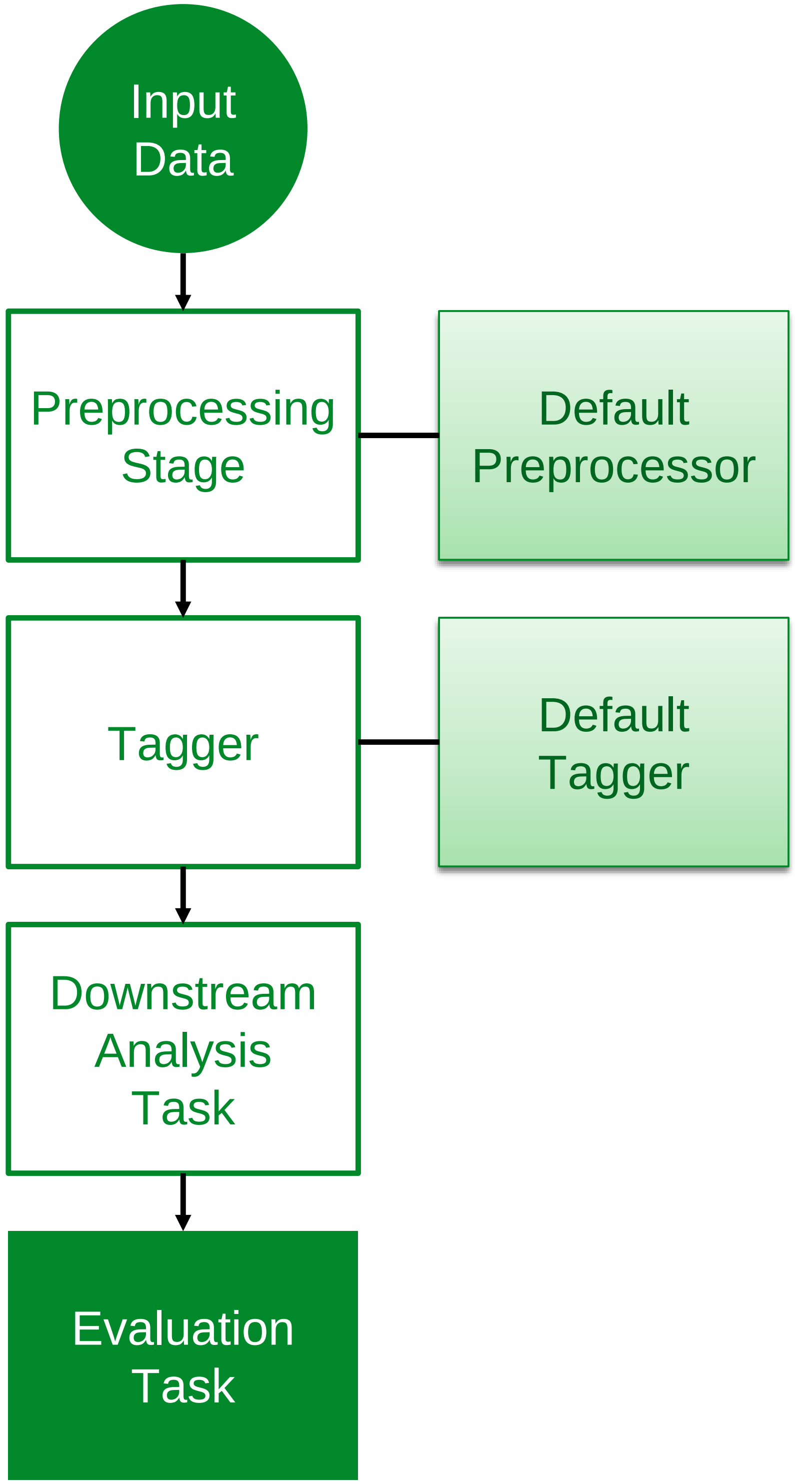
1. Benchmark workflow defined by **coordinator** along with input data.
2. **Users** provide code (e.g. docker containers) that satisfy workflow stages, input parameters, and input data (file upload).
3. **Back-end** processes workflows and evaluates metrics (powered for example by REANA).
4. **Front-end**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



reana

Reproducible research data analysis platform

Benchmark Workflow Example



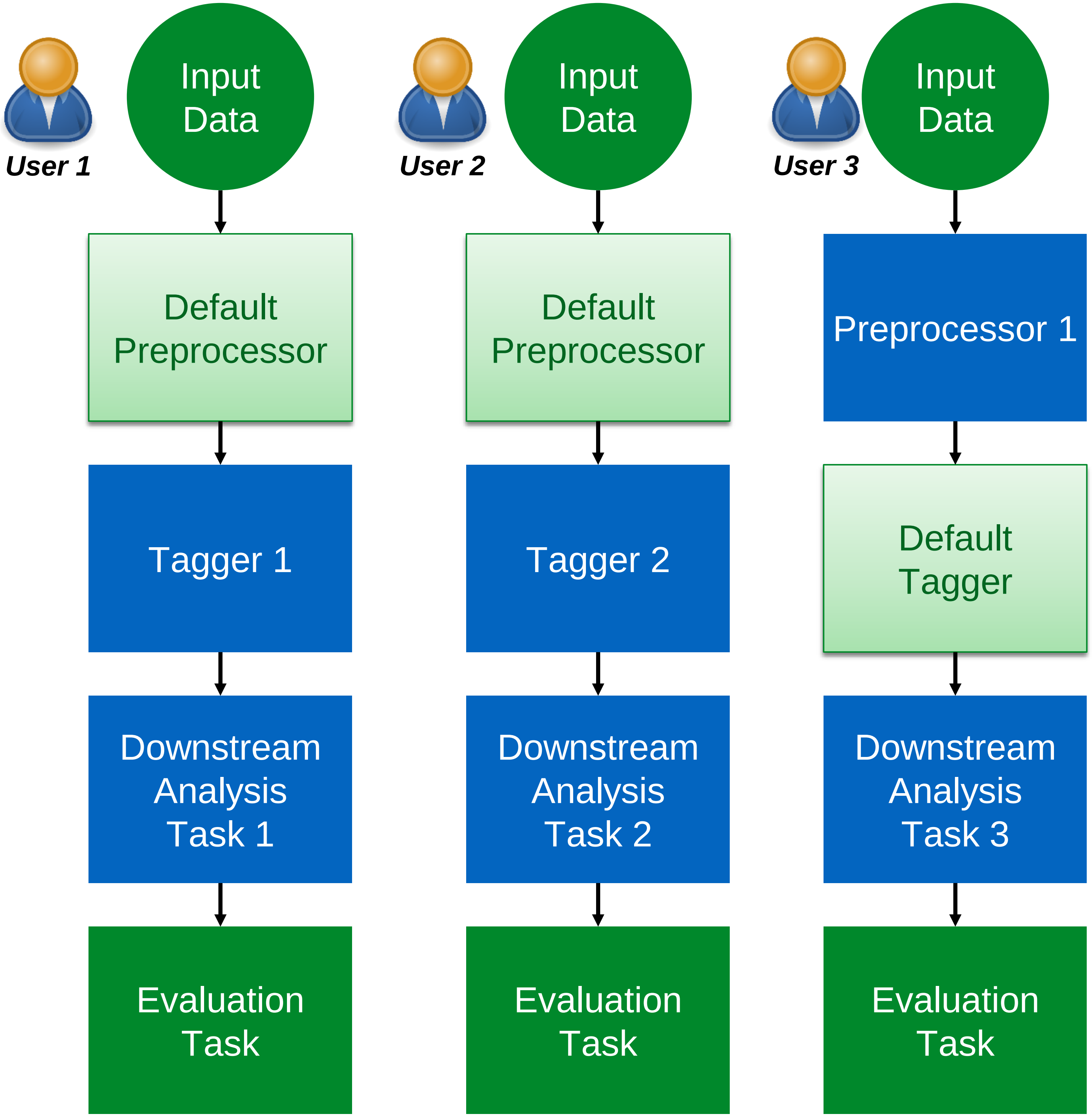
Workflow Templates

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

NYU

Benchmark Workflow Example (cont.)



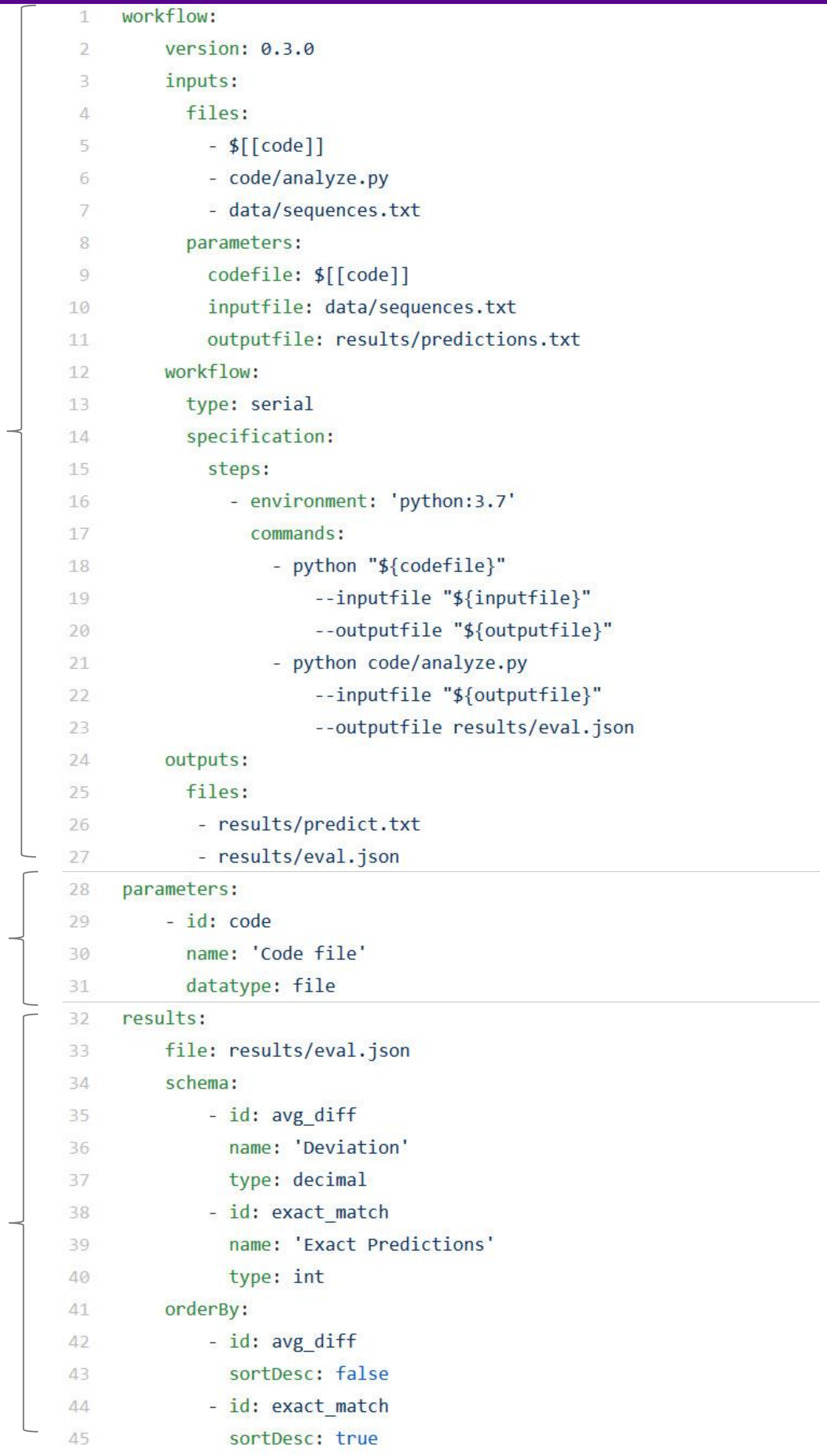
Benchmark Participants

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

Workflow Templates

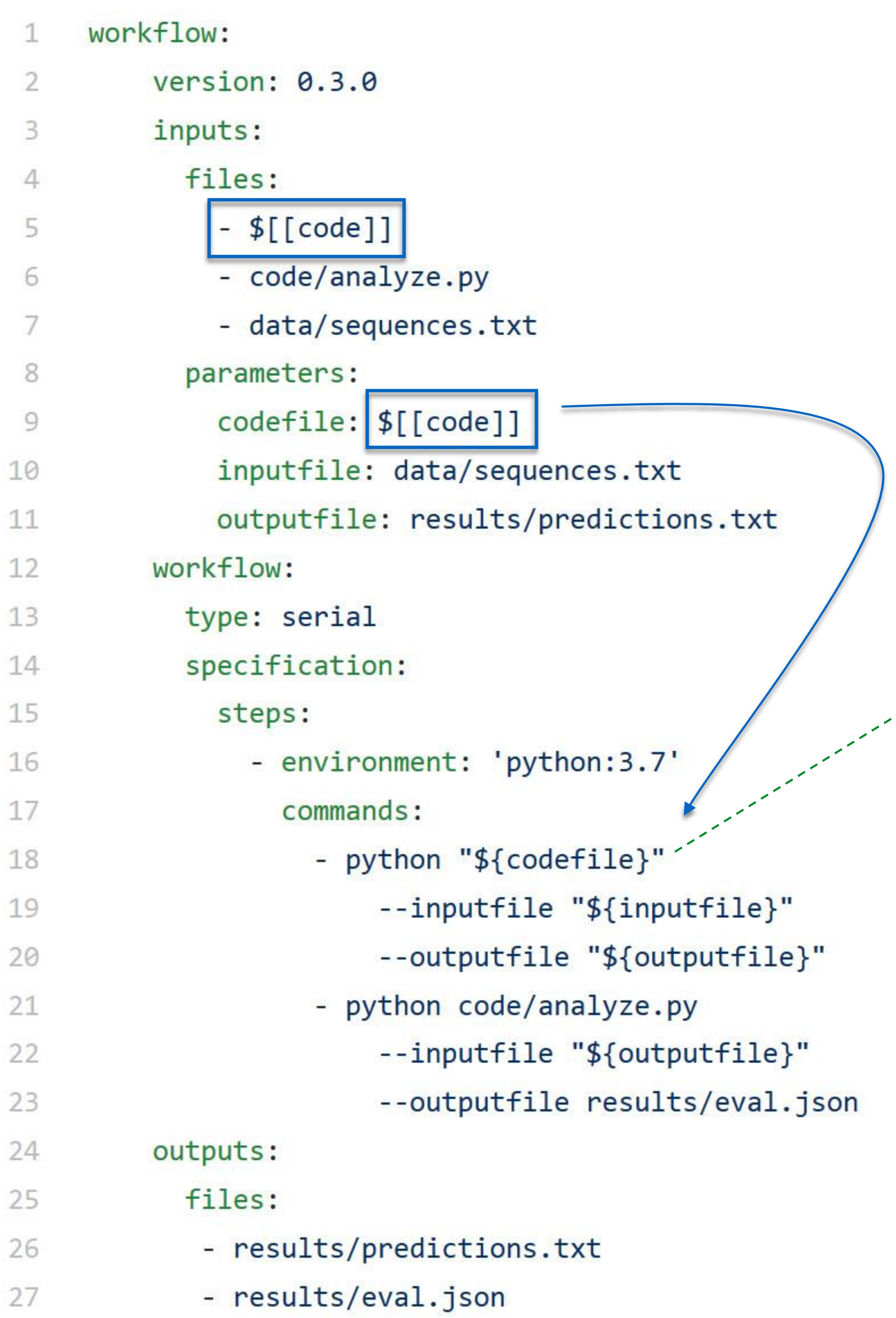
Components of Workflow Templates

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*'.

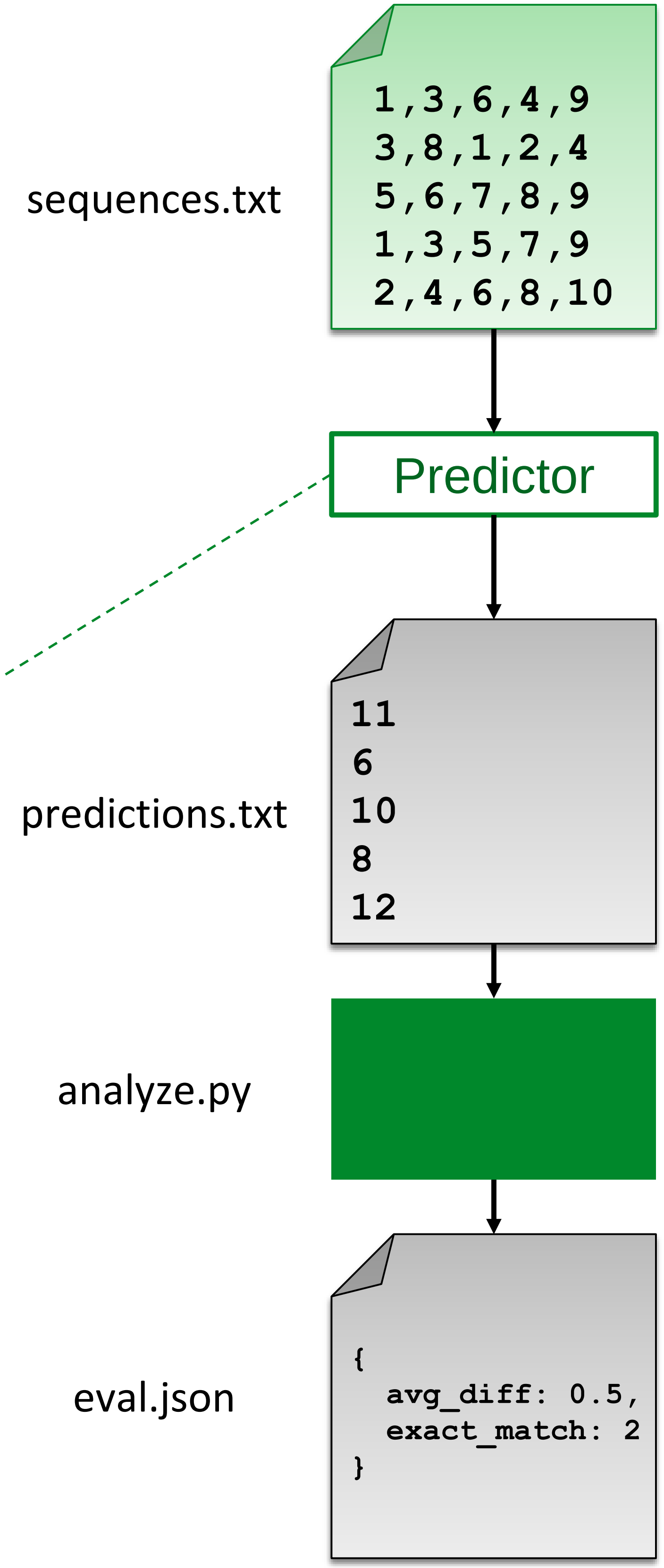


```
workflow:
    version: 0.3.0
    inputs:
      files:
        - $[[code]]
        - code/analyze.py
        - data/sequences.txt
      parameters:
        codefile: $[[code]]
        inputfile: data/sequences.txt
        outputfile: results/predictions.txt
    workflow:
      type: serial
      specification: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



```
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
```



```
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
```

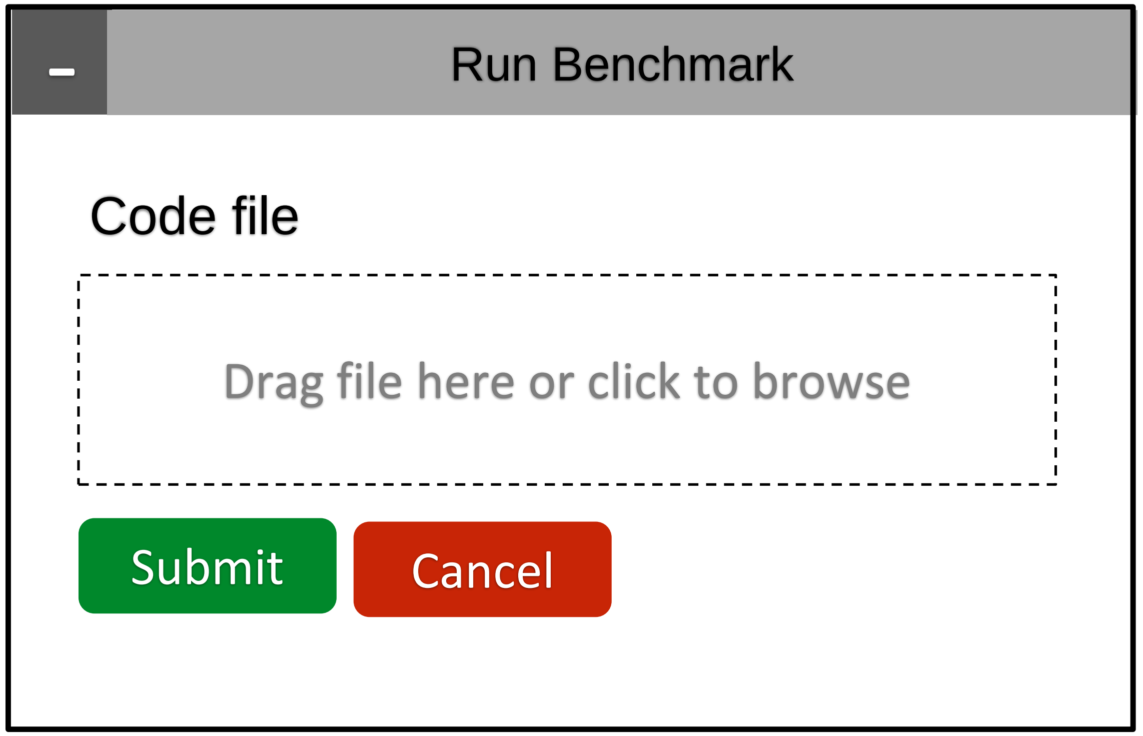
```
11
6
10
8
12
```

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

Workflow Templates (cont.)

Render input form from parameter declarations

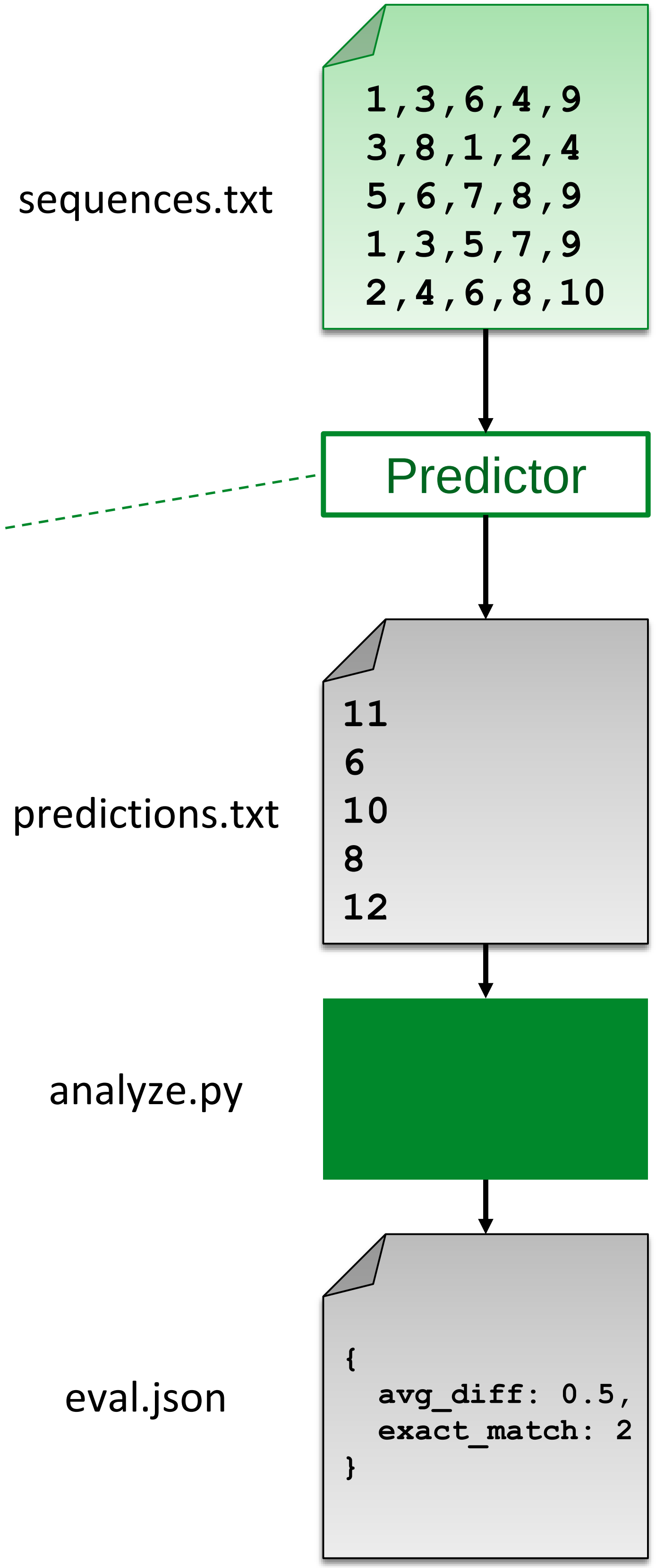
```
parameters:
    - id: code
      name: 'Code file'
      datatype: file
```



Run Benchmark

Code file

Drag file here or click to browse



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
```

predictions.txt

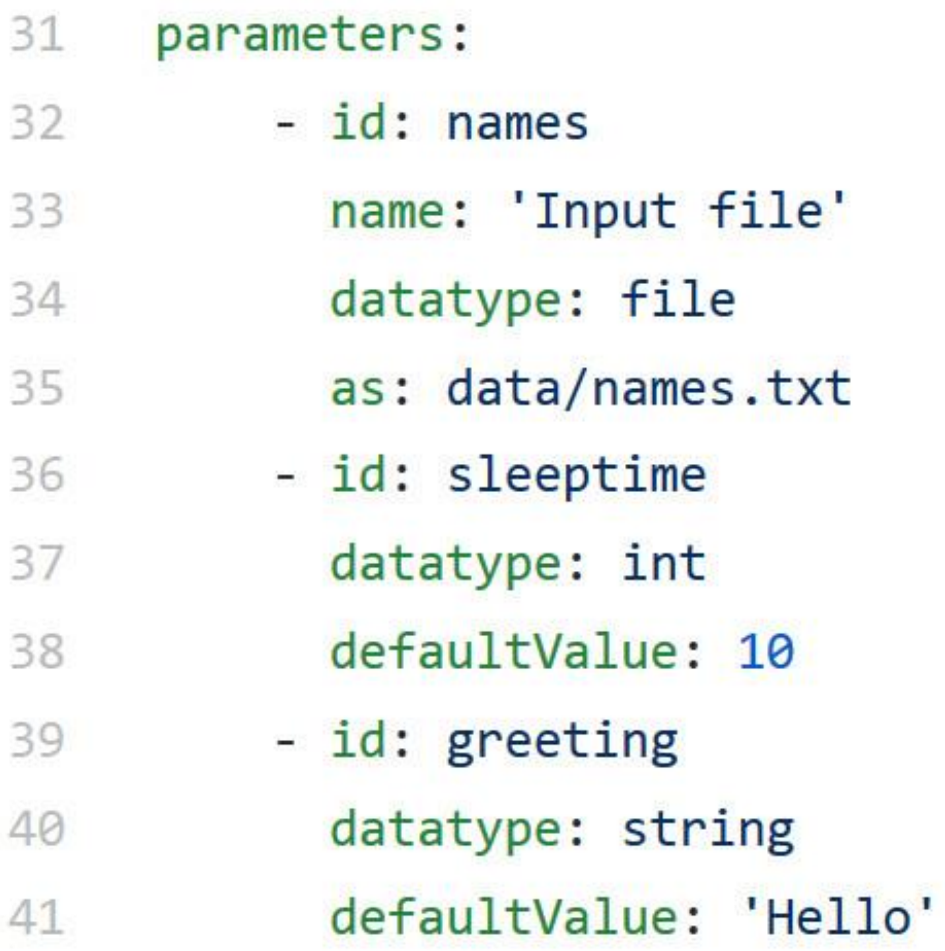
```
11
6
10
8
12
```

eval.json

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

Workflow Templates (cont.)

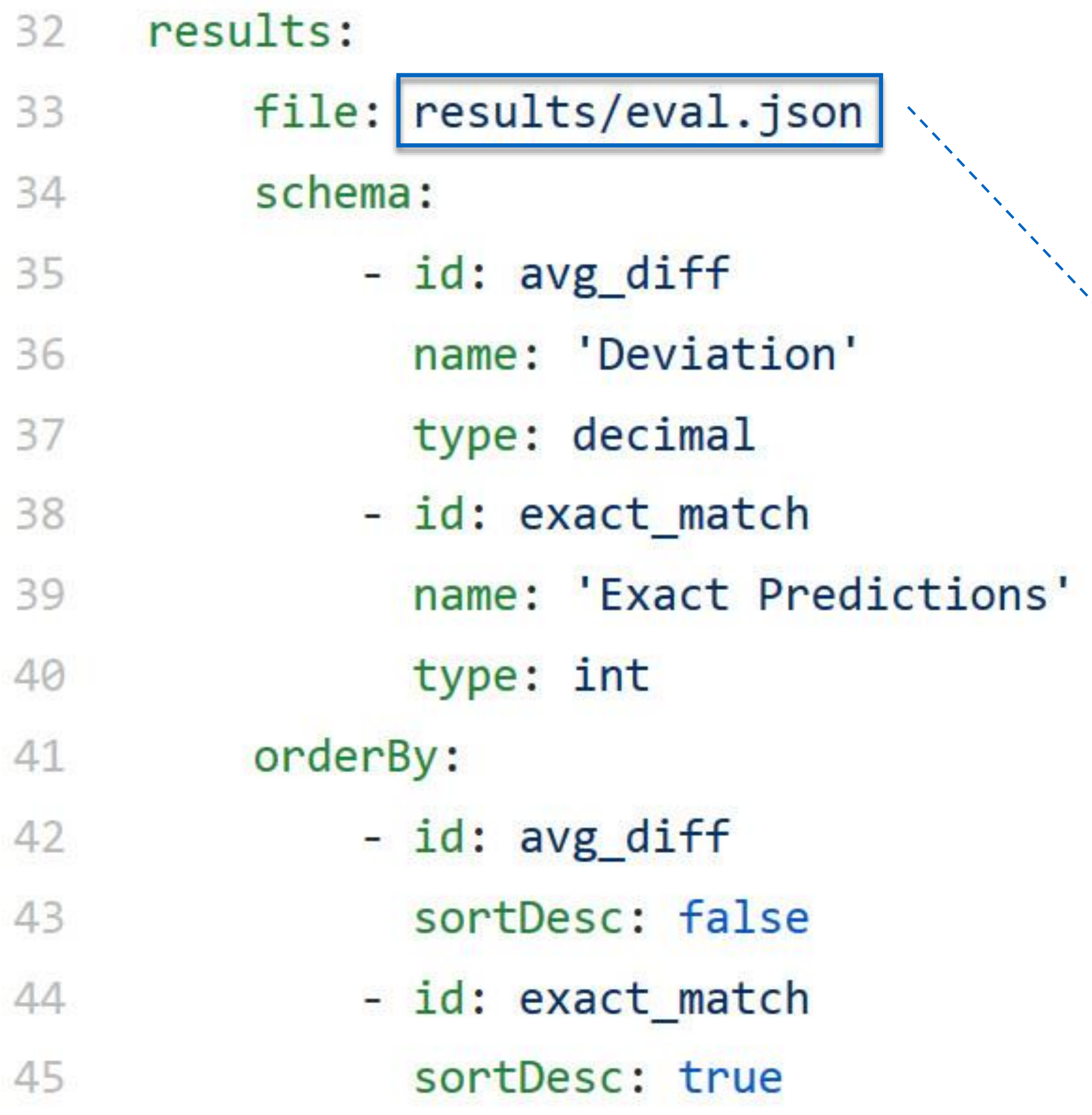
Parameters for ‘Hello World’



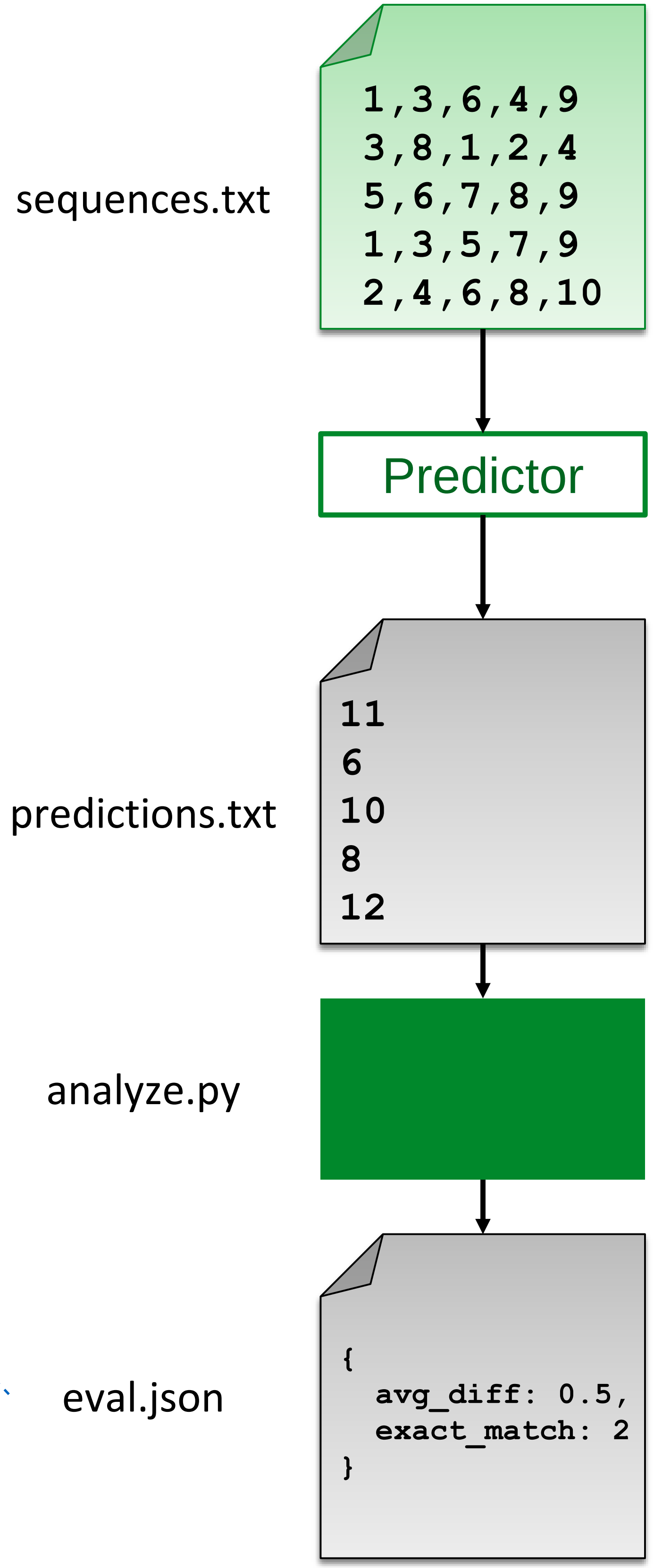
```yaml
parameters:
    - id: names
      name: 'Input file'
      datatype: file
      as: data/names.txt
    - id: sleeptime
      datatype: int
      defaultValue: 10
    - id: greeting
      datatype: string
      defaultValue: 'Hello'
```

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
```



```
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
}
```