# Implementation of Jupyter Notebooks into The Reproducible Open Benchmarks for Data Analysis Platform (ROB)

Aaron Wang, Heiko Müller, Shih-Chieh Hsu, Ajay Rawat
University of Washington
6/7/2021

# Background

- Inspired by the “Machine Learning Landscape of Top Taggers Paper”
  - https://arxiv.org/abs/1902.09914v2
  - They compiled and compared multiple top tagging models
- A significant amount of time is required to organize and evaluate large benchmarks
- Goal of the ROB is to provide a platform that **automates the collection, execution and comparison of submissions** of participants in the benchmark

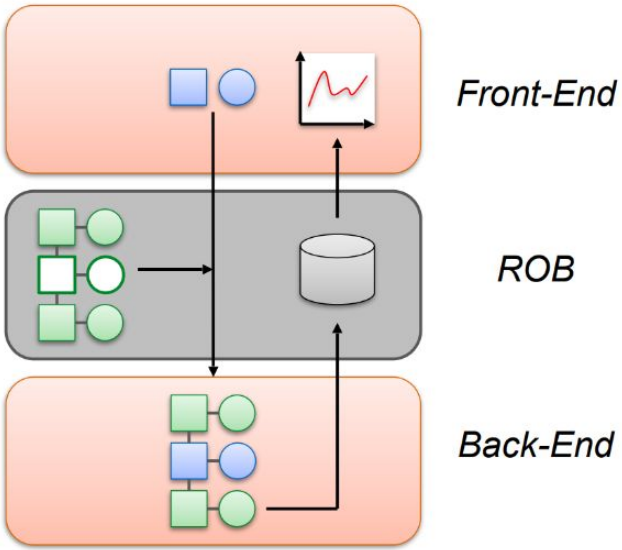
| | AUC | Acc | $1/\epsilon_B$ ($\epsilon_S = 0.3$) | | | #Param |
|---|---|---|---|---|---|---|
| | | | single | mean | median | |
| CNN [16] | 0.981 | 0.930 | 914±14 | 995±15 | 975±18 | 610k |
| ResNeXt [30] | 0.984 | 0.936 | 1122±47 | 1270±28 | 1286±31 | 1.46M |
| TopoDNN [18] | 0.972 | 0.916 | 295±5 | 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 for Data Analysis Platform (ROB)

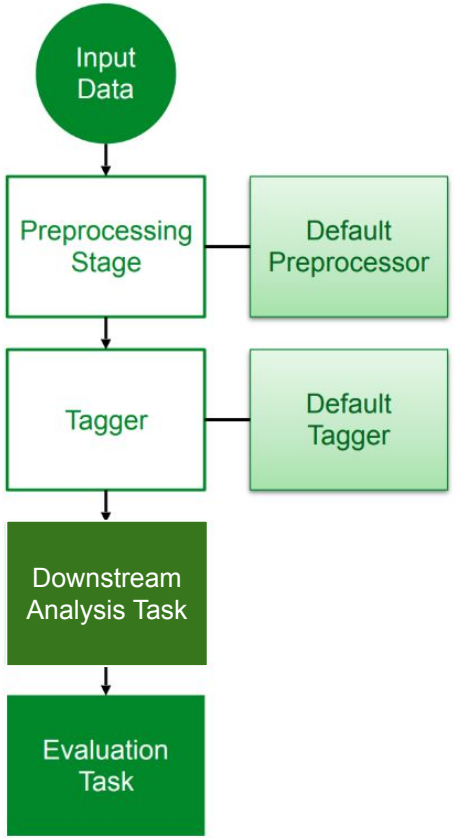
Exploratory work for enabling such community benchmarks.

***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

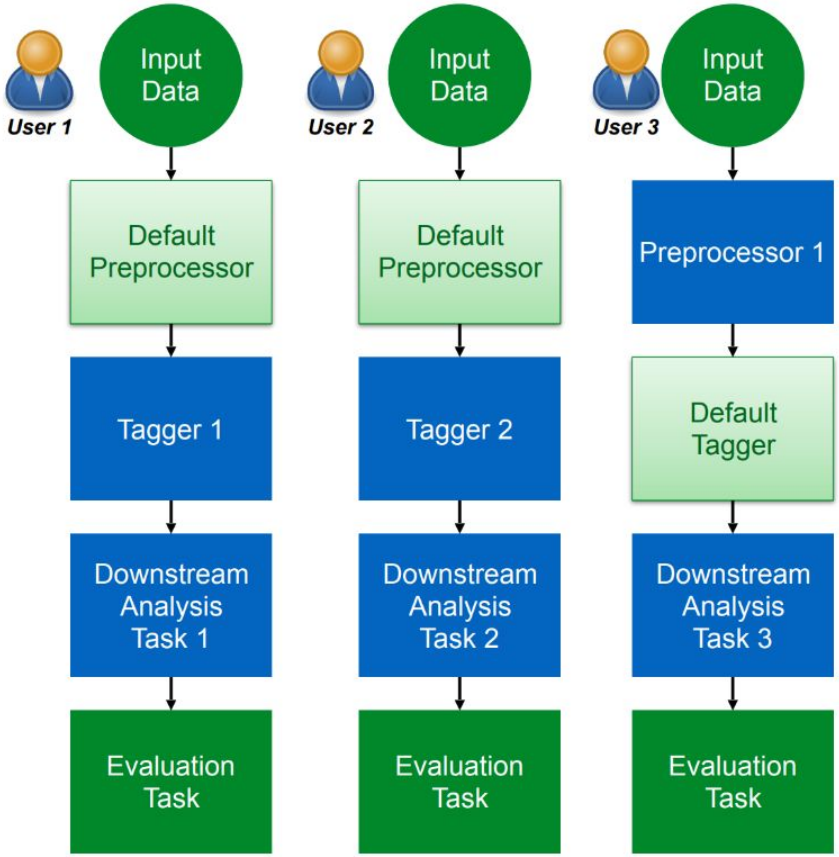



***Workflow Templates***

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).

# Current Problem:

- ROB currently requires participants to package their submissions into docker containers
  - Steep learning curve for people unfamiliar

# Solution

- In order to increase ease of use, we implement support for the commonly used Jupyter Notebooks
  - We allow users to submit their implementations directly as Jupyter Notebooks

# How to run a Jupyter Notebook?

- Jupyter Notebook executions are cell by cell
- Cannot be run like a python script

Issues we needed to solve:

- How to run the same jupyter notebook with new parameters without needing to open and edit the notebook it self
- How to run the Jupyter Notebook within the ROB without the need to open the python notebook using the Jupyter Notebook Editor

# Papermill!


- Papermill is a convenient way to run Jupyter Notebooks
  - It is a tool that parametrizes, and executes notebooks in its entirety
  - https://papermill.readthedocs.io/en/latest/

# How Papermill Works

- Users are required to define one cell in their Jupyter Notebook with the tag “parameters”


- Users need to ensure that the entire notebook runs with no errors (just like any python script)
- Ensure that all parameters that can or might be changed in the future is included in the parameters cell

# How Papermill Works Continued.

- Users can then execute the notebook using the papermill “execute notebook function”
  - Inputs are:
    - Path to the input
    - Path to the output notebook
    - Dictionary of parameters

```
execute_notebook(<input notebook>, <output notebook>, <dictionary of parameters>)
```

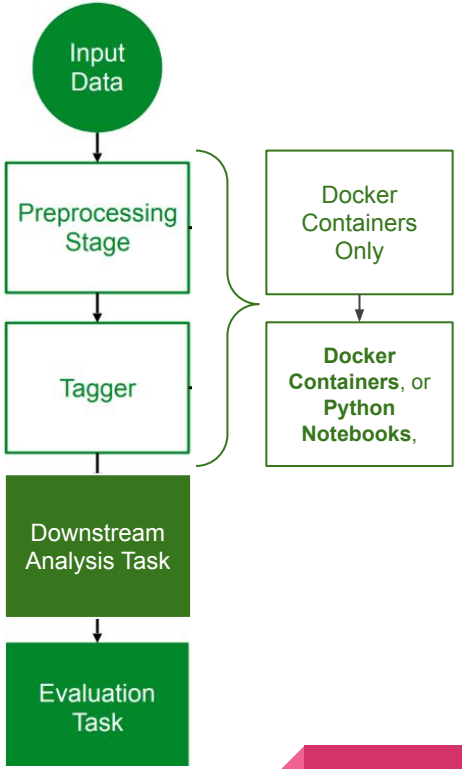
```
import papermill as pm

pm.execute_notebook(
   'path/to/input.ipynb',
   'path/to/output.ipynb',
   parameters=dict(alpha=0.6, ratio=0.1)
)
```

- Will execute your notebook similar to how scripts can be executed
- Will generate an output notebook at the path of the outback notebook, with an output that is exactly like if you pressed “run all cells” in the Jupyter Notebooks
- Parameter values in the notebook will be replaced by the values defined in the function

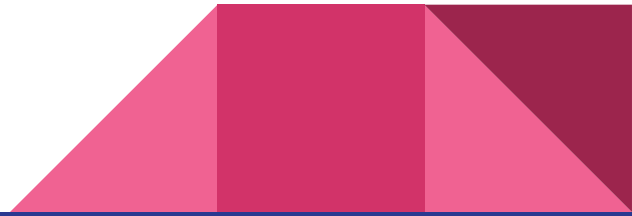
# Implementing Papermill Into ROB

- Create a new **Step Type** that uses papermill to execute the notebook
- Create a new “worker” that executes steps of the given context
  - A function that calls the step and executes the function
- Create another worker that is able to run Jupyter Notebooks contained within docker containers



# Jet Flavor Demo

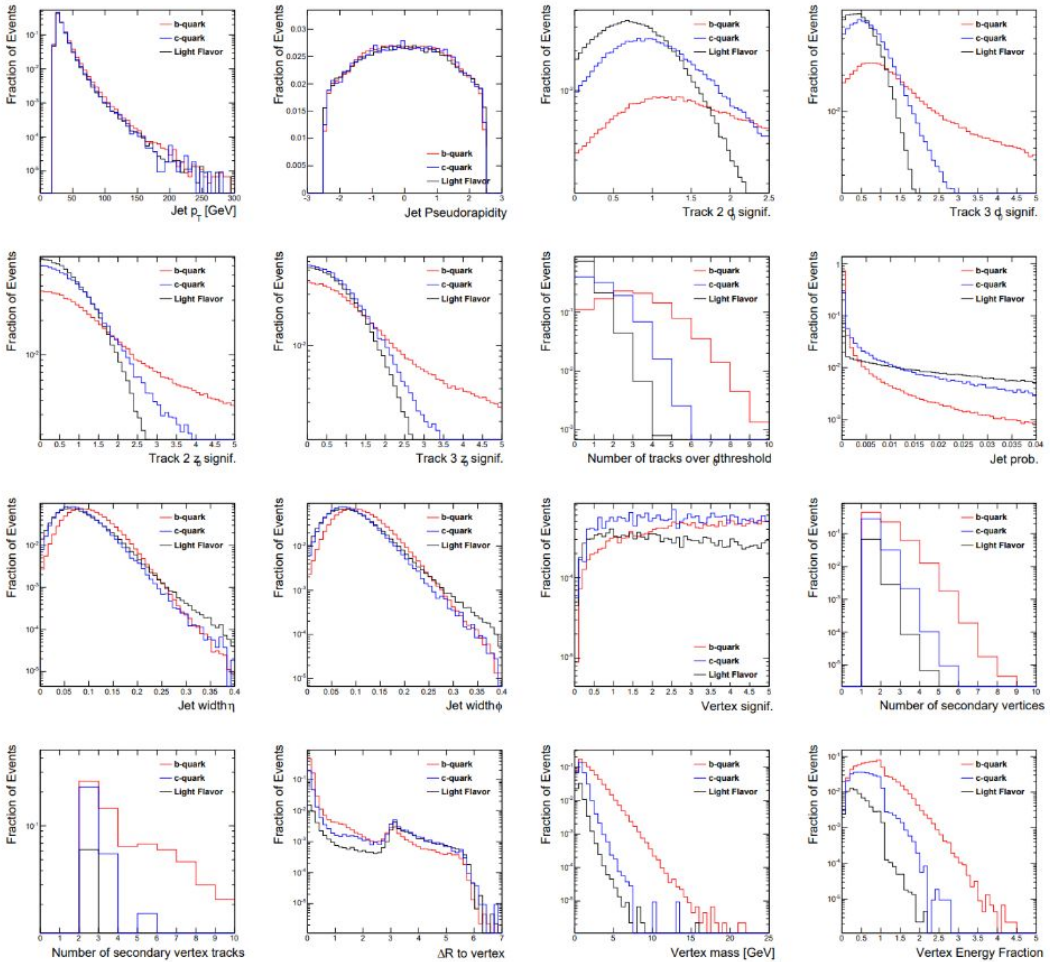
- Implementation of the Jet Flavor Dataset into the ROB
  - http://mlphysics.ics.uci.edu/data/hb_jet_flavor_2016/
- Inspired by the “Jet Flavor Classification in High-Energy Physics with Deep Neural Networks” paper
  - https://arxiv.org/pdf/1607.08633.pdf



# Dataset

- 11 million jets total
- Composed of 3 classes - light, charm and bottom
  - 44:11:45
  - light and charm are background, and bottom is signal
- Composed of 4 types of variables in addition to jet pT and jet eta:
  - High level track variables
  - High level vertex variables
  - Low level Track variables
  - Low Level Vertex variables
- Each jet has between 1 and 33 tracks, with the mean being 4 tracks

http://mlphysics.ics.uci.edu/data/hb_jet_flavor_2016/readme.txt



- Lots of combinations of features that need to be compared
- Hard to compile all the models and train all of them individually
- Implement into the ROB as a workflow to make comparison and training on the same validation set easier and more organized

TABLE I: Performance results for networks using track-level, vertex-level or expert-level information. In each case the jet $p_T$ and pseudorapidity are also used. Shown for each method is the Area Under the Curve (AUC), the integral of the background efficiency versus signal efficiency, which have a statistical uncertainty of 0.001 or less. Signal efficiency and background rejections are shown in Figs. 6-10.

| Inputs | | | Technique | AUC |
|---|---|---|---|---|
| Tracks | Vertices | Expert | | |
| ✓ | | | Feedforward | 0.916 |
| ✓ | | | LSTM | 0.917 |
| ✓ | | | Outer | 0.915 |
| | ✓ | | Feedforward | 0.912 |
| | ✓ | | LSTM | 0.911 |
| | ✓ | | Outer | 0.911 |
| ✓ | ✓ | | Feedforward | 0.929 |
| ✓ | ✓ | | LSTM | 0.929 |
| ✓ | ✓ | | Outer | 0.928 |
| | | ✓ | Feedforward | 0.924 |
| | | ✓ | LSTM | 0.925 |
| | | ✓ | Outer | 0.924 |
| ✓ | | ✓ | Feedforward | 0.937 |
| ✓ | | ✓ | LSTM | 0.937 |
| ✓ | | ✓ | Outer | 0.936 |
| | ✓ | ✓ | Feedforward | 0.931 |
| | ✓ | ✓ | LSTM | 0.930 |
| | ✓ | ✓ | Outer | 0.929 |
| ✓ | ✓ | ✓ | Feedforward | 0.939 |
| ✓ | ✓ | ✓ | LSTM | 0.939 |
| ✓ | ✓ | ✓ | Outer | 0.937 |

# Workflow

User inputs:

- Model architectures
- Preprocessing script

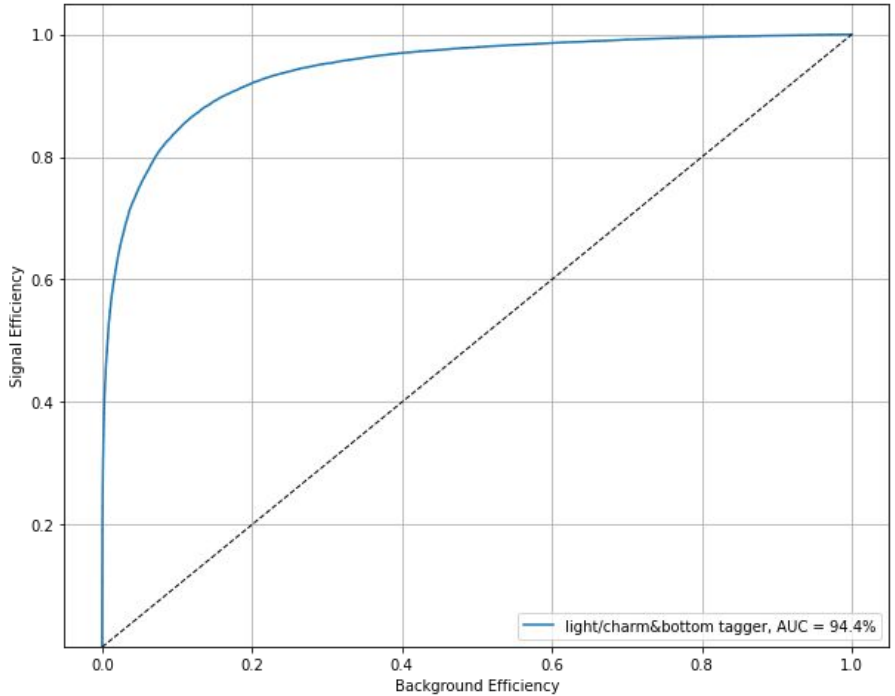
Workflow Coordinator inputs:

- Training and validation data
- Validation script

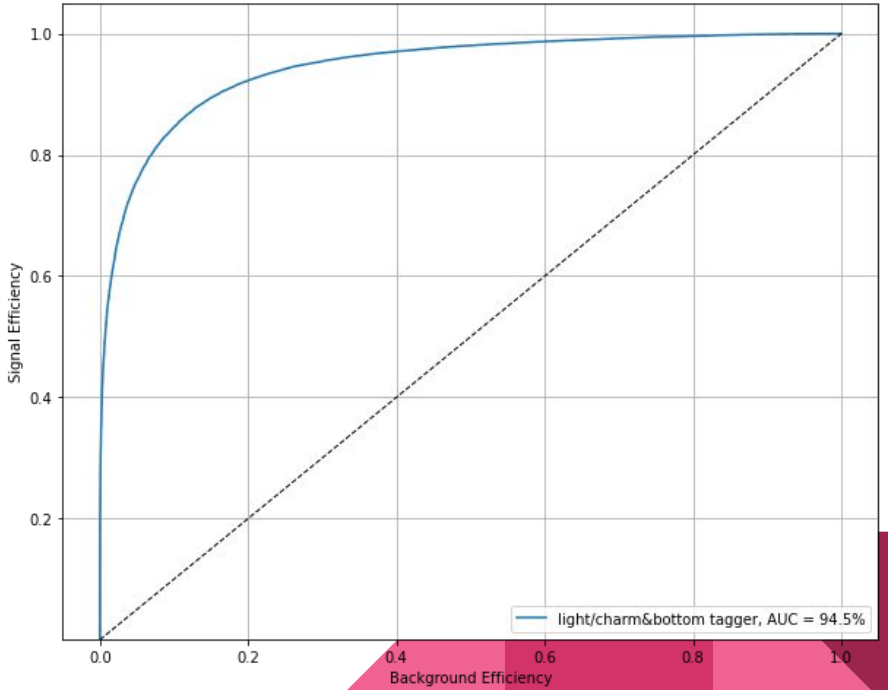
```
workflow:
    files:
        inputs:
        - code/LSTM.ipynb
        - code/preprocess.ipynb
        - data/X_test.npy
        - data/y_test.npy
        - models/dense.h5
        outputs:
        - data/X_preprocess.npy
        - data/y_preprocess.npy
        - results/roc_curve.png
```

# Example ROC Outputs:

**LSTM ROC Curve**



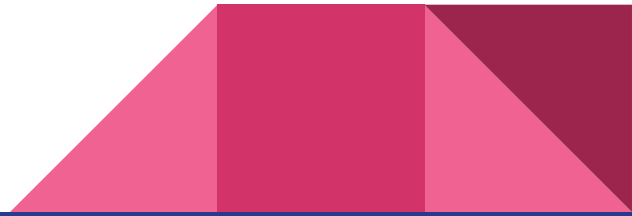

**GRU ROC Curve**



# Summary

- Implemented jupyter notebooks into the ROB as a form of input
- Created demos using Jupyter Notebooks for use of the ROB for
  - Quickdraw dataset
  - Jet Flavor Dataset
  - Top Tagging dataset
  - Mnist Dataset
- Github Repositories: https://github.com/anrunw/ROB, https://github.com/scailfin/flowserv-core
-

# Questions?



# References

[1] “Reproducible and Reusable Data Analysis Workflow Server”, https://github.com/scailfin/flowserv-core

[2] Kasieczka, Gregor, Plehn, Tilman, Butter, Anja, Cranmer, Kyle, Debnath, Dipsikha, Dillon, Barry M, . . . Varma, Sreedevi. (2019). The Machine Learning landscape of top taggers. SciPost Physics, 7(1), 014.

[3] “Jupyter Notebooks”, https://jupyter.org/

[4] “Papermill”, https://papermill.readthedocs.io/en/latest/

[5] “Particle Physics”, https://github.com/anrunw/ROB