HEP ML Lab

An end-to-end framework for machine learning application in high energy physics

Jing Li, Hao Sun
Dalian University of Technology, Liaoning, China
ML4Jets (2023)

Preparing for ArXiv
https://github.com/Star9daisy/hep-ml-lab

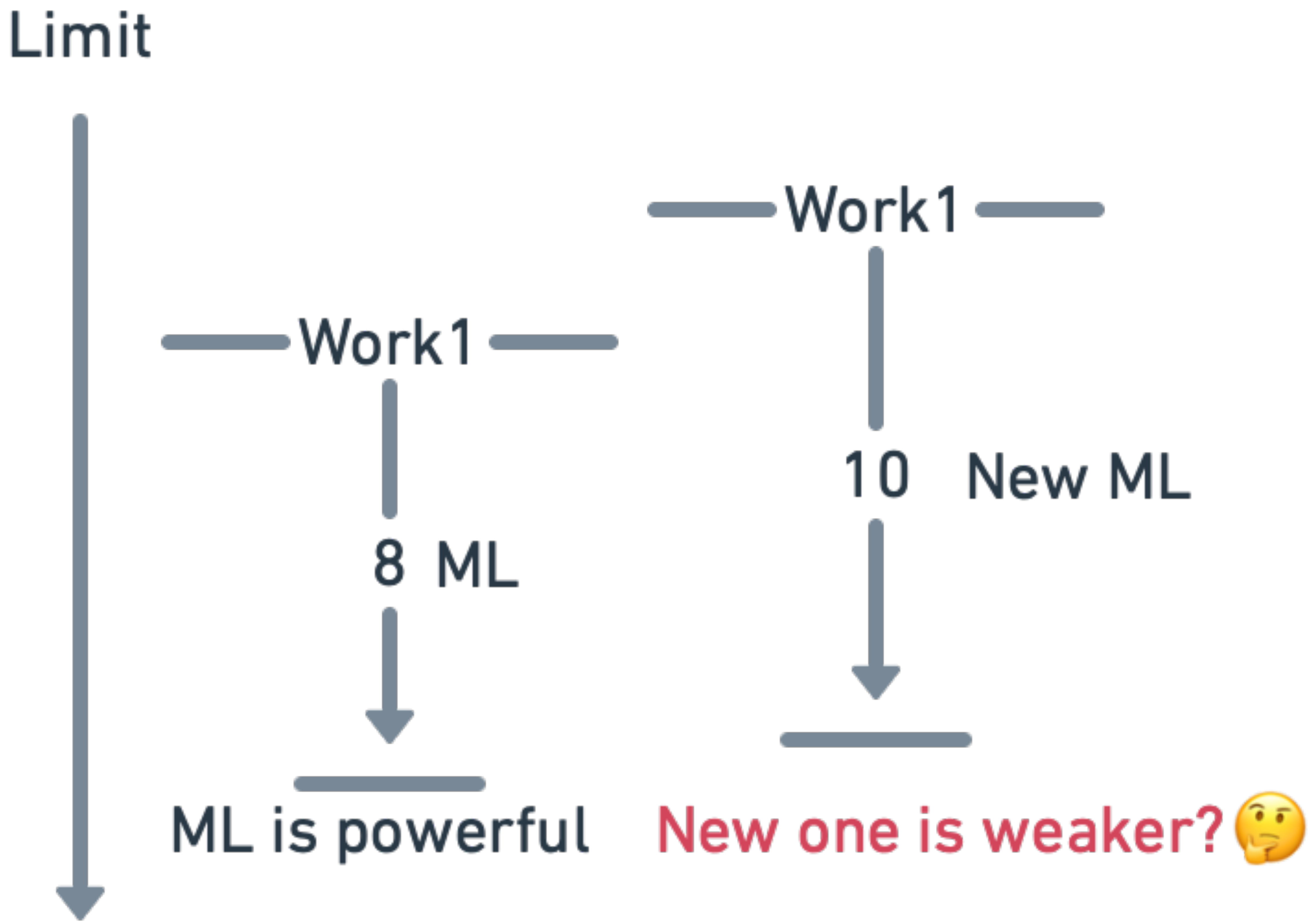
Table of contents

- Introduction: why we need an end-to-end framework?
- Quick start: generate events, create datasets, apply approaches
- Future: roadmap

Introduction

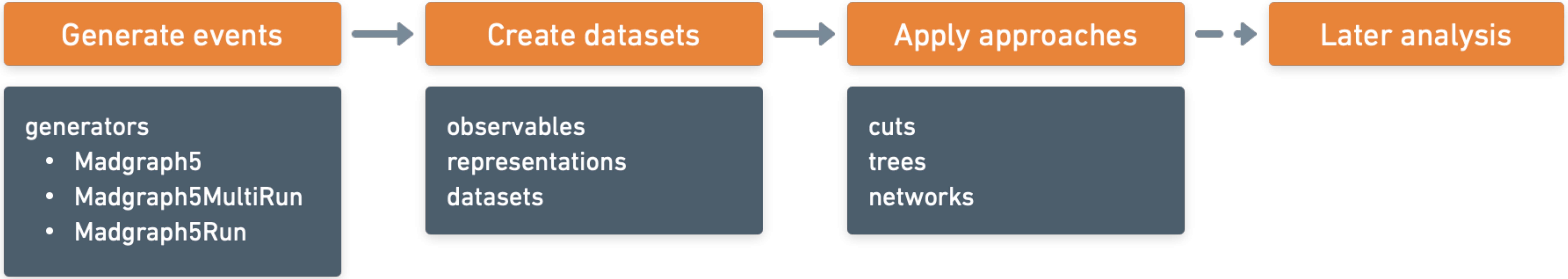
Reproduction issues

- A lot of works have explore the potential of machine learning approaches.
- The lack of source codes leads to reproduction problems.
- It’s crucial to compare different algorithms on the same baseline.



Introduction

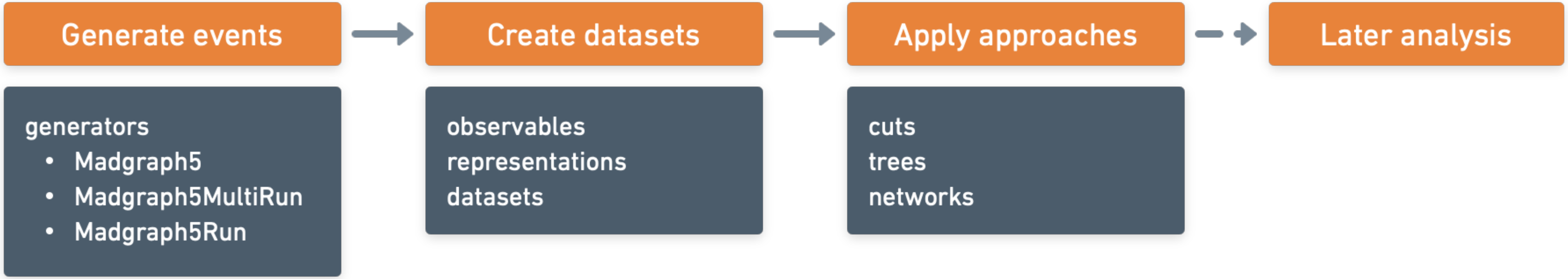
Control from end to end



- HEP ML Lab (HML) helps apply different approaches into studies.
- An end-to-end framework ensure reproducibility.

Introduction

Highlights



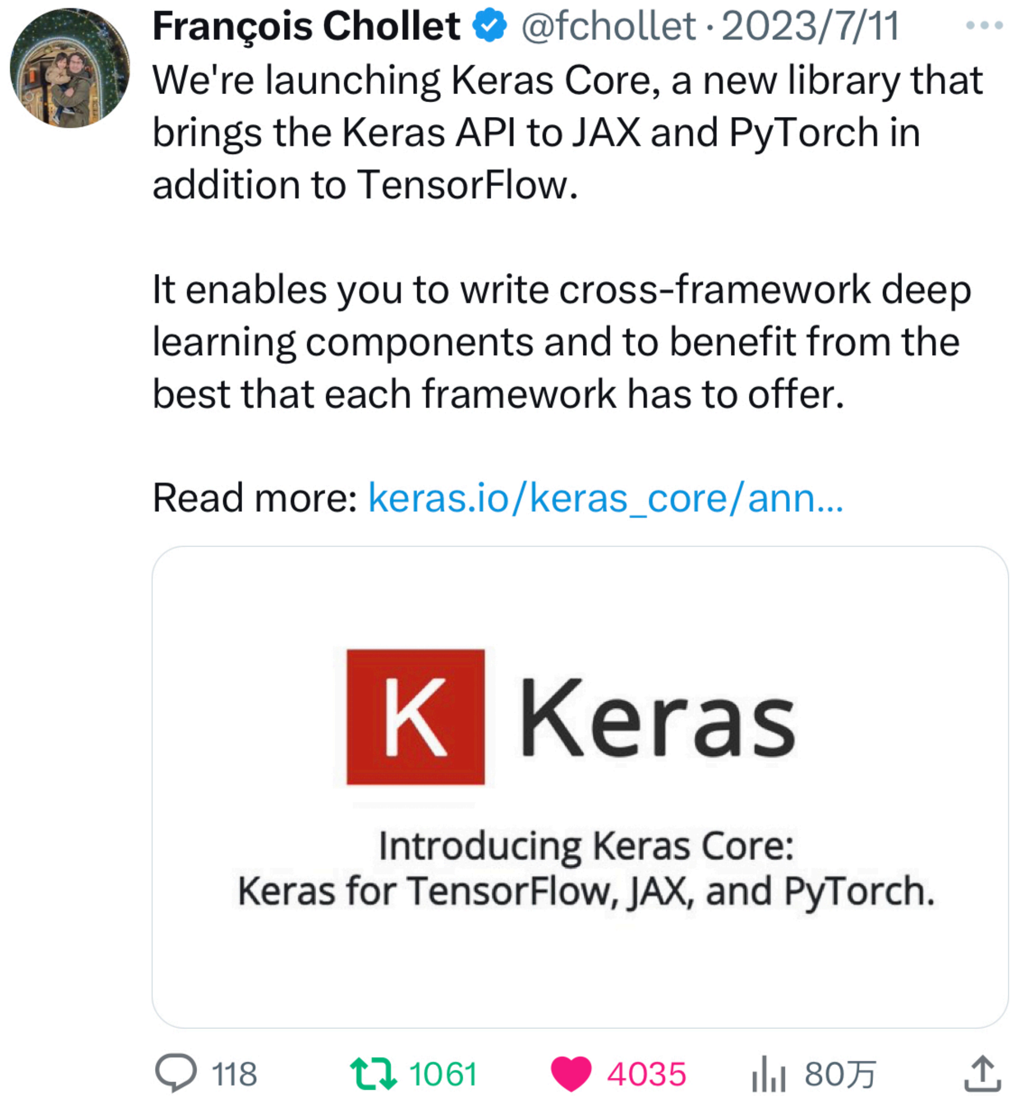
Generate events → Create datasets → Apply approaches → Later analysis

generators
- Madgraph5
- Madgraph5MultiRun
- Madgraph5Run

observables
representations
datasets

cuts
trees
networks

- Data control from the very beginning.
- Approaches in **Keras** style.
  - Cross-framework support 👉



François Chollet @fchollet · 2023/7/11

We're launching Keras Core, a new library that brings the Keras API to JAX and PyTorch in addition to TensorFlow.

It enables you to write cross-framework deep learning components and to benefit from the best that each framework has to offer.

Read more: keras.io/keras_core/ann...

Keras

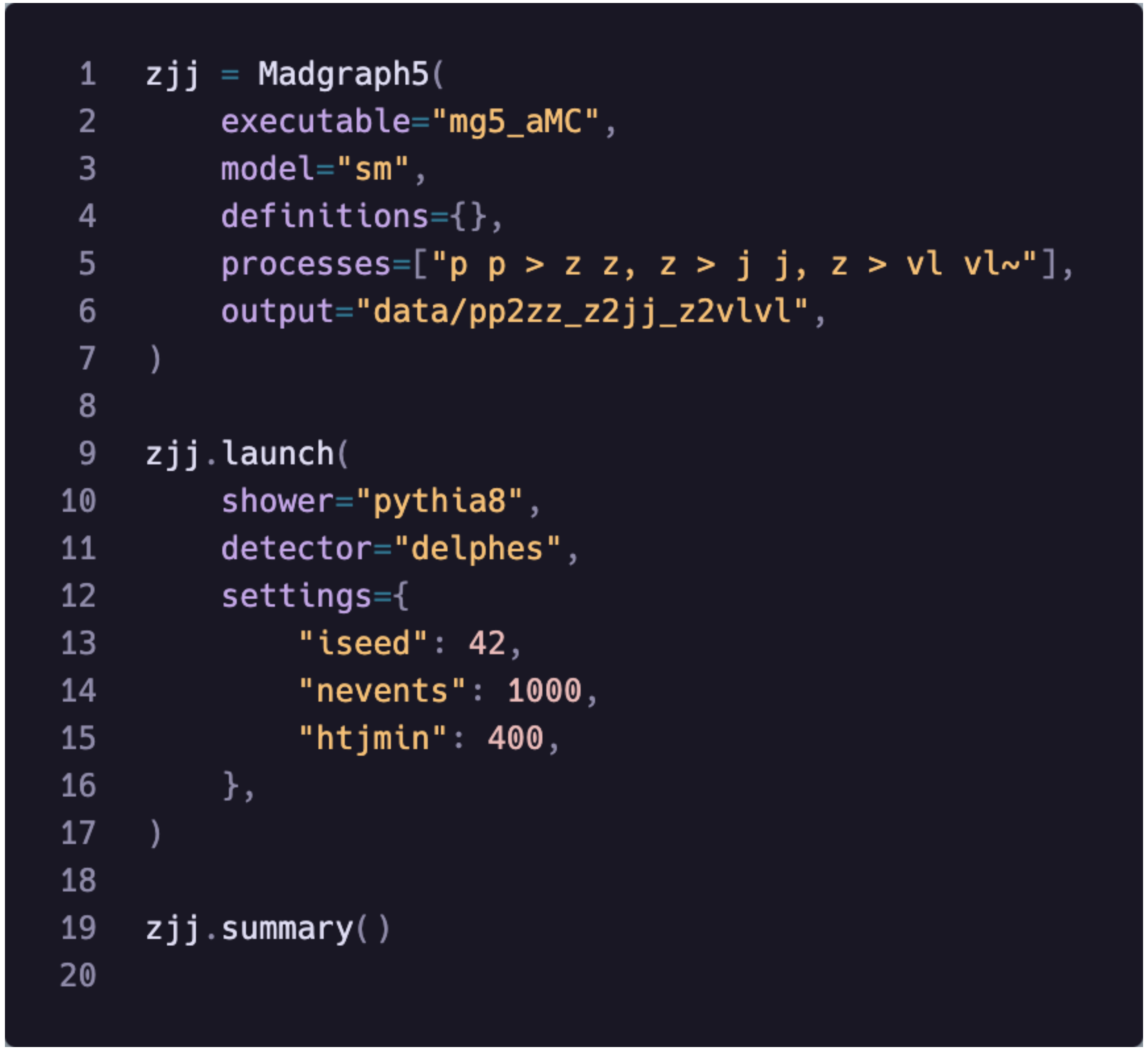
Introducing Keras Core:
Keras for TensorFlow, JAX, and PyTorch.

118 1061 4035 80万

Quick start

Generate events

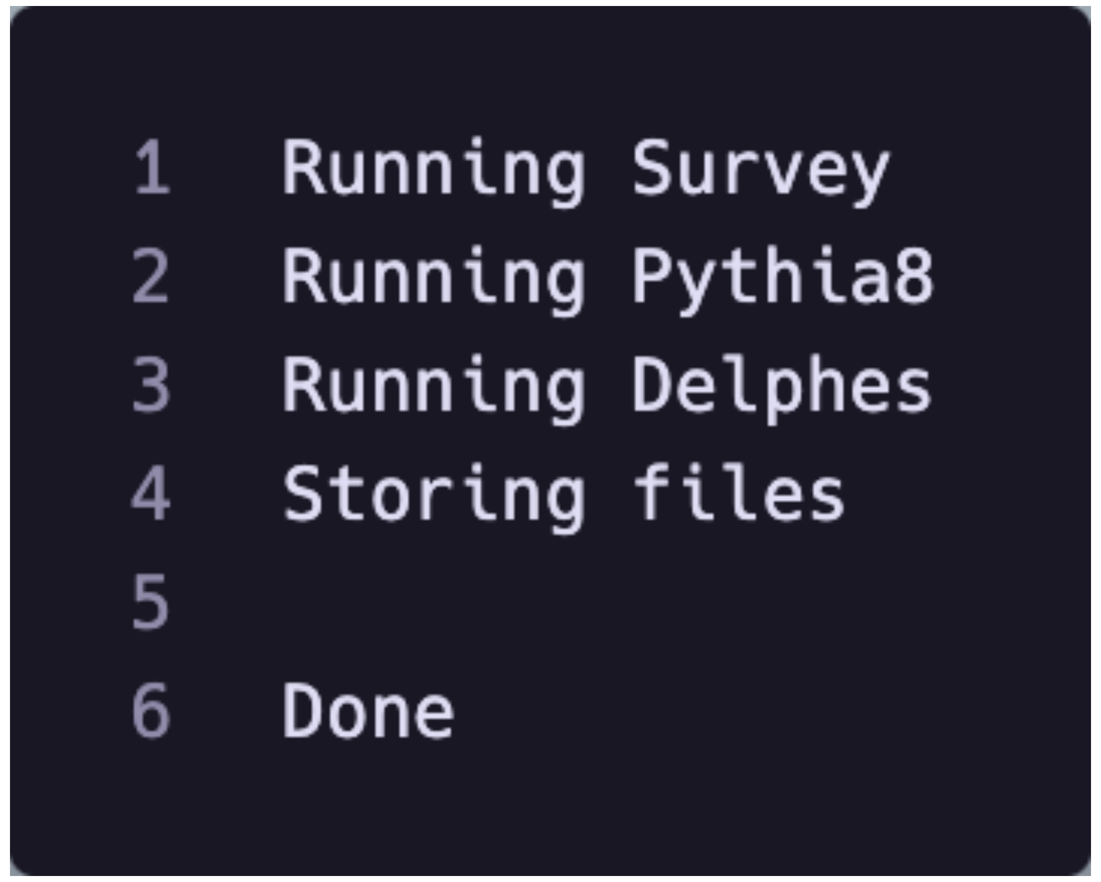
- Classical three commands
    - **generate, output, launch**
- Optimized parameters
    - **definitions, processes**
    - **settings**
- **summary** to show all runs' info



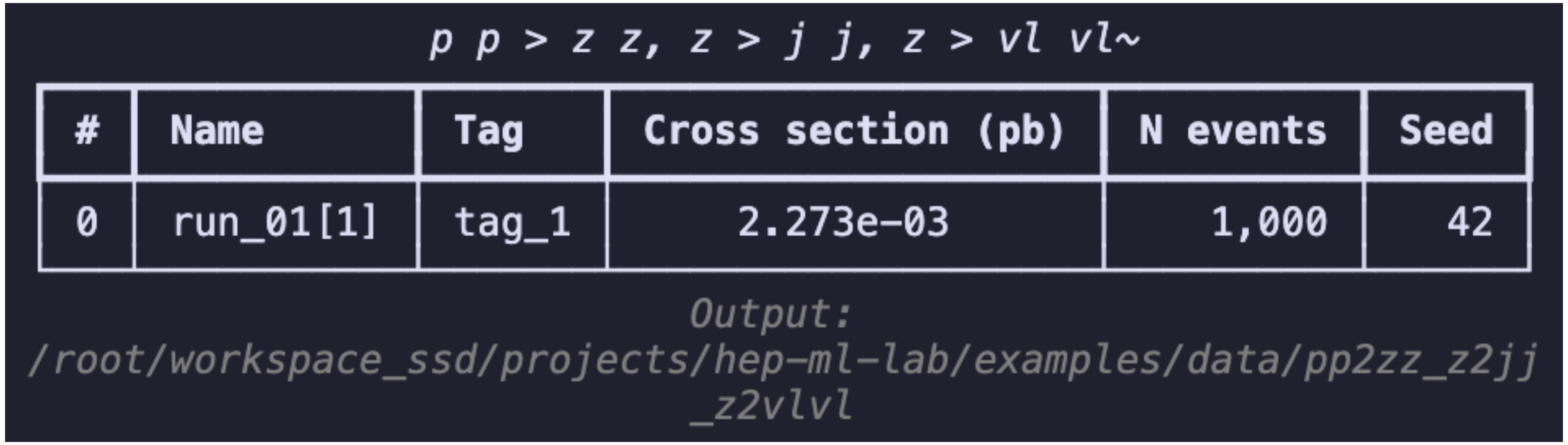
```python
zjj = Madgraph5(
    executable="mg5_aMC",
    model="sm",
    definitions={},
    processes=["p p > z z, z > j j, z > vl vl~"],
    output="data/pp2zz_z2jj_z2vlvl",
)

zjj.launch(
    shower="pythia8",
    detector="delphes",
    settings={
        "iseed": 42,
        "nevents": 1000,
        "htjmin": 400,
    },
)

zjj.summary()
```

Quick start

Generate events



```
Running Survey
Running Pythia8
Running Delphes
Storing files

Done
```

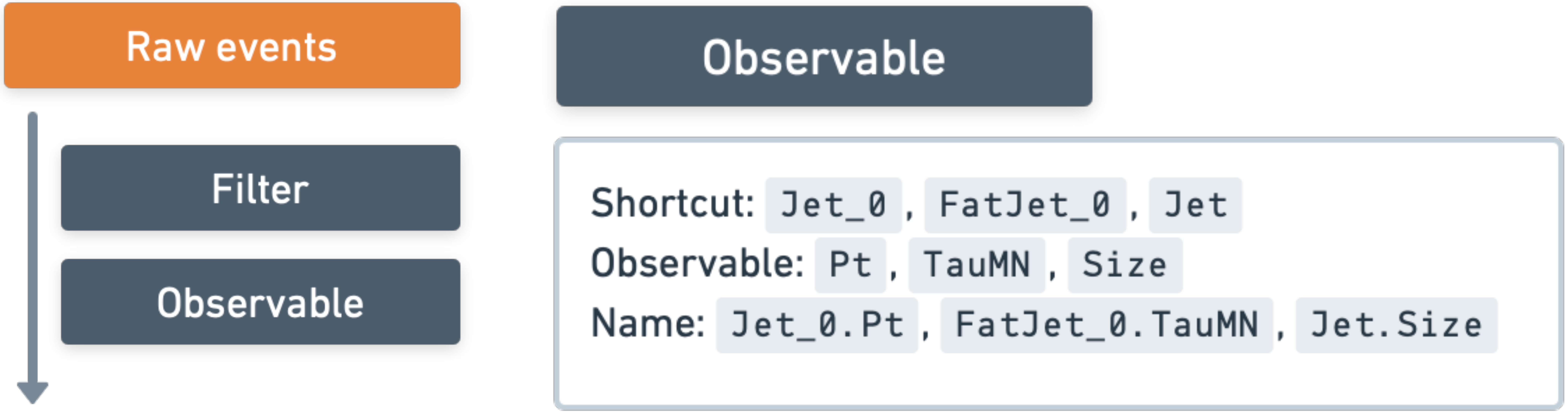


```
                      p p > z z, z > j j, z > vl vl~
┌───┬───────────┬───────┬────────────────────┬──────────┬──────┐
│ # │ Name      │ Tag   │ Cross section (pb) │ N events │ Seed │
├───┼───────────┼───────┼────────────────────┼──────────┼──────┤
│ 0 │ run_01[1] │ tag_1 │     2.273e-03      │    1,000 │   42 │
└───┴───────────┴───────┴────────────────────┴──────────┴──────┘
                                Output:
/root/workspace_ssd/projects/hep-ml-lab/examples/data/pp2zz_z2jj
                               _z2vlvl
```

- Simple status checker
- Summary table

Quick start

Create datasets



Shortcut: `Jet_0`, `FatJet_0`, `Jet`

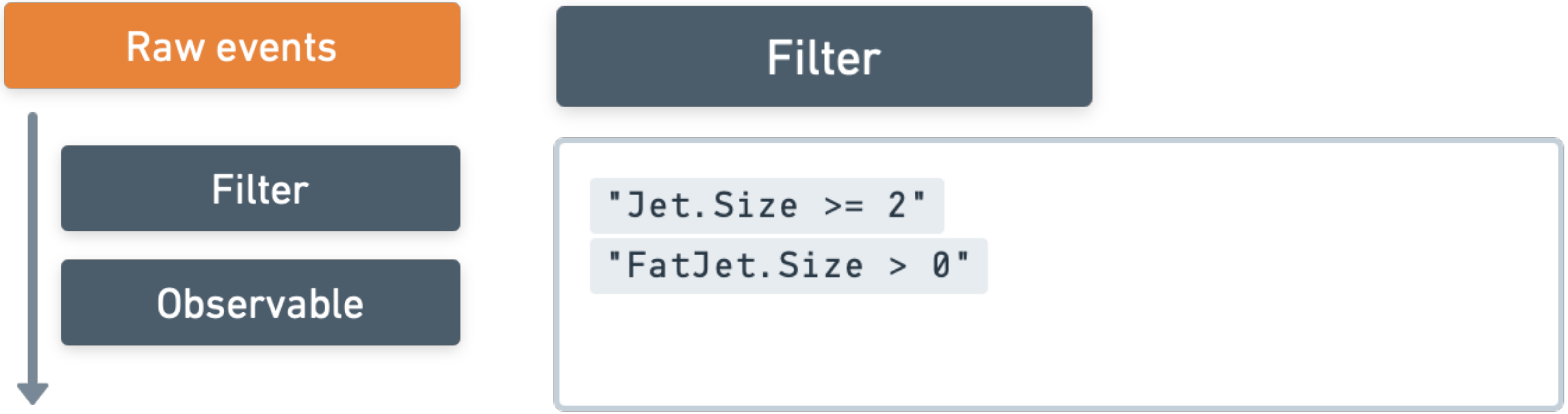
Observable: `Pt`, `TauMN`, `Size`

Name: `Jet_0.Pt`, `FatJet_0.TauMN`, `Jet.Size`

- Observable parsing system: name = shortcut + observable.

Quick start

Create datasets

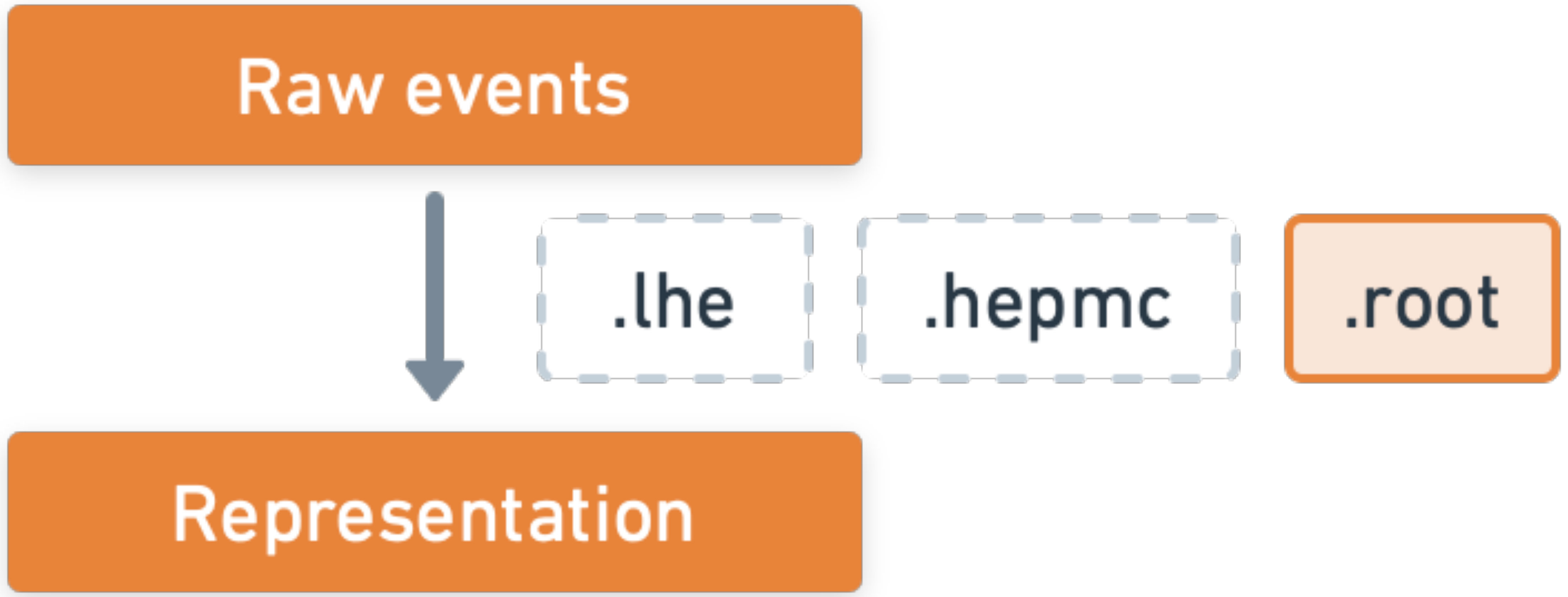


```
"Jet.Size >= 2"
"FatJet.Size > 0"
```

- **Observable** parsing system: name = shortcut + observable.
- **Filter** accepts a list of logical "and" conditions.

Quick start

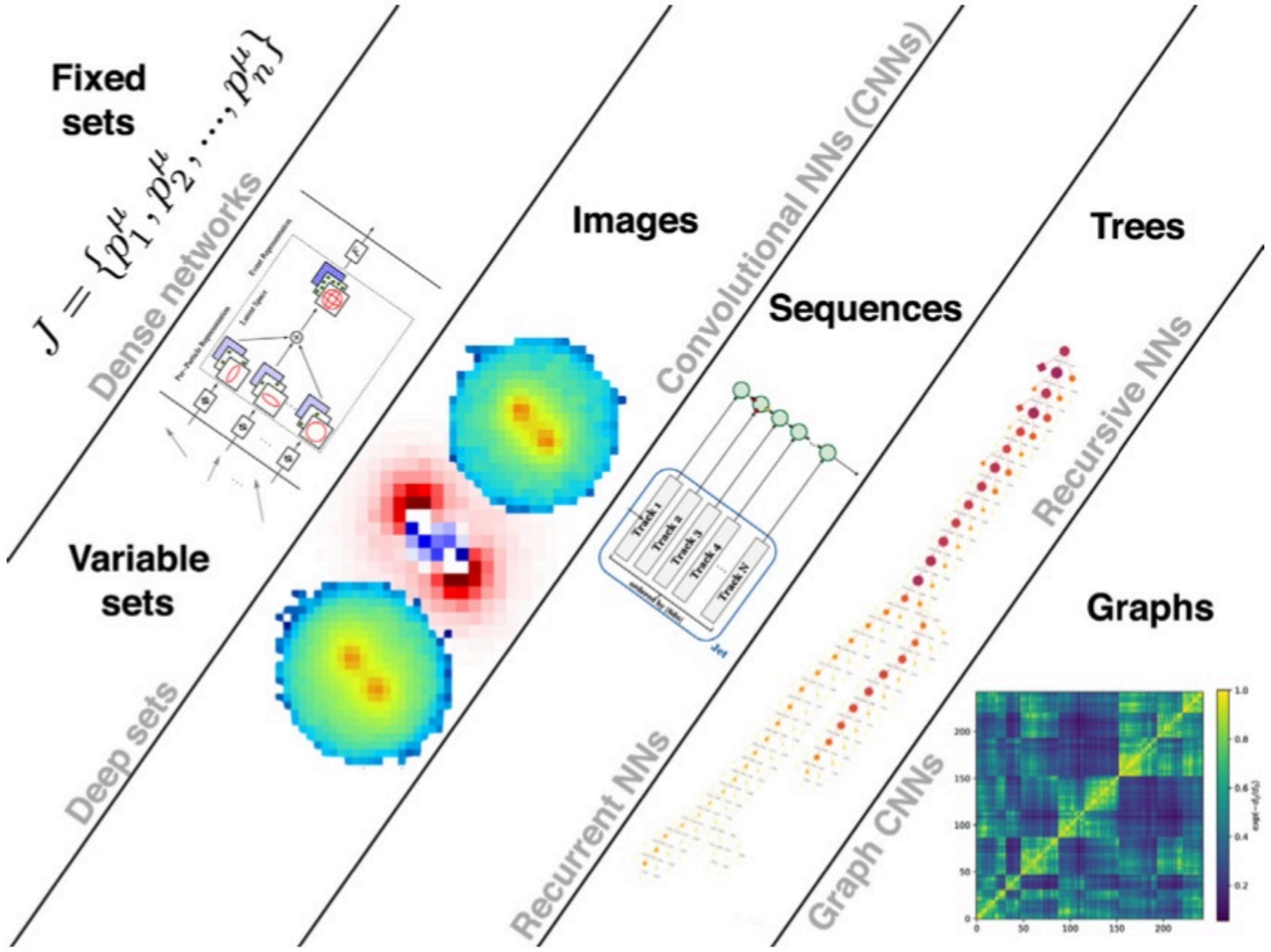
Create datasets



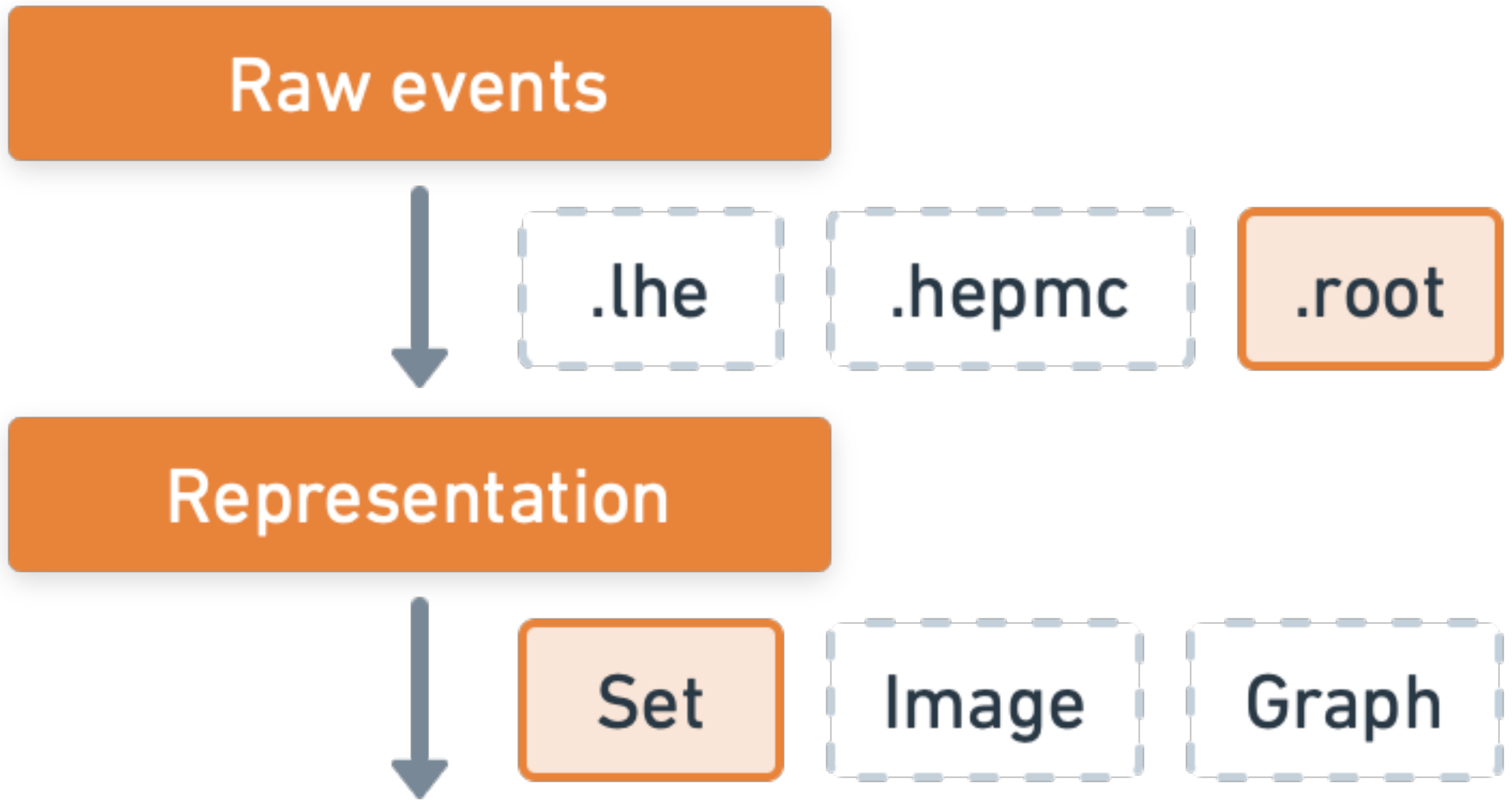
- **Observable** parsing system: name = shortcut + observable.
- **Filter** accepts a list of logical "and" conditions.
- Currently only supports root format.

Quick start

Create datasets



[1709.04464] Jet Substructure at the Large Hadron Collider:
A Review of Recent Advances in Theory and Machine Learning



Raw events

.lhe .hepmc .root

Representation

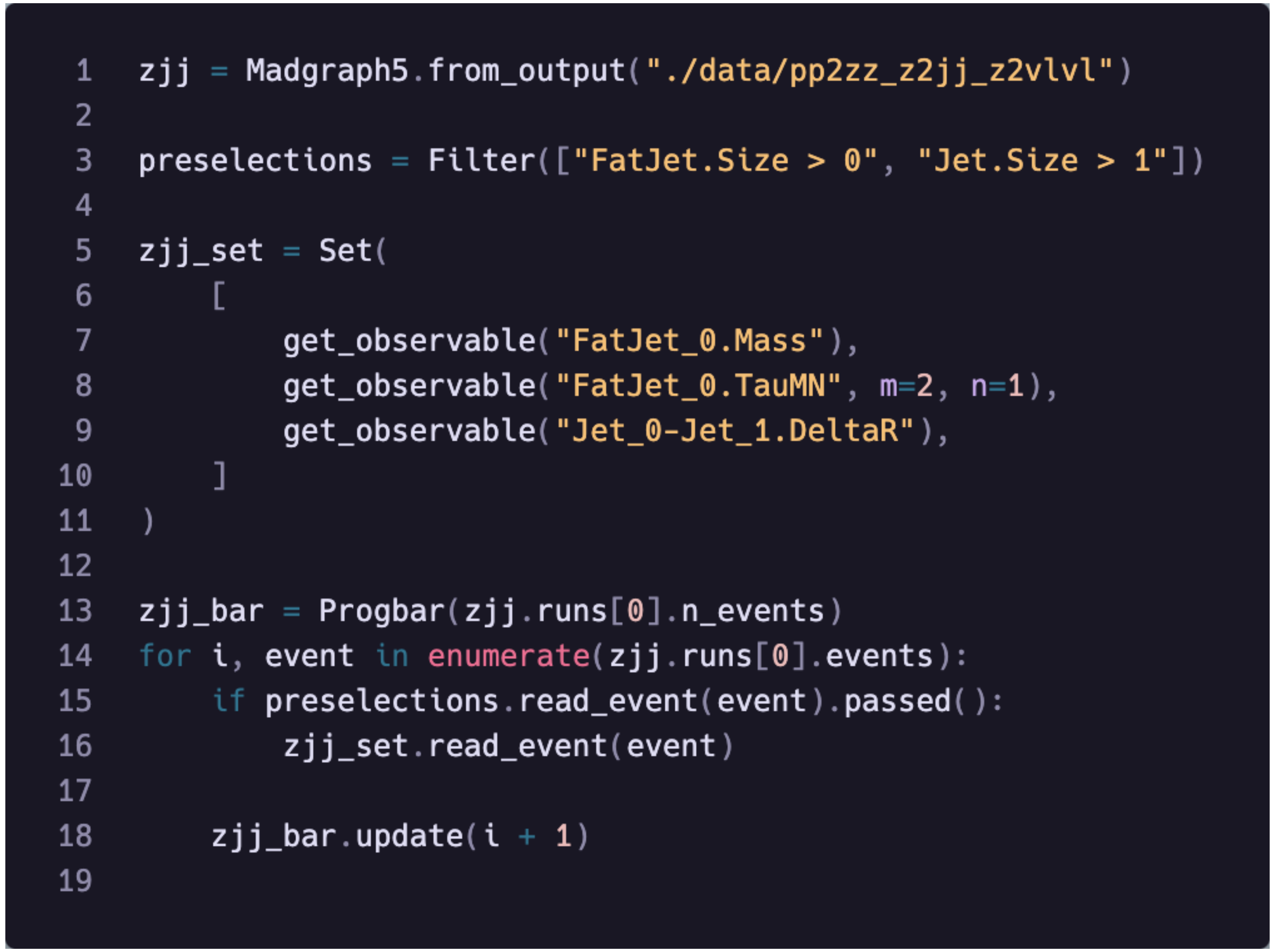
Set Image Graph

- Currently only supports **Set**.
- Expanding to **Image** and **Graph**.

Quick start

Create datasets

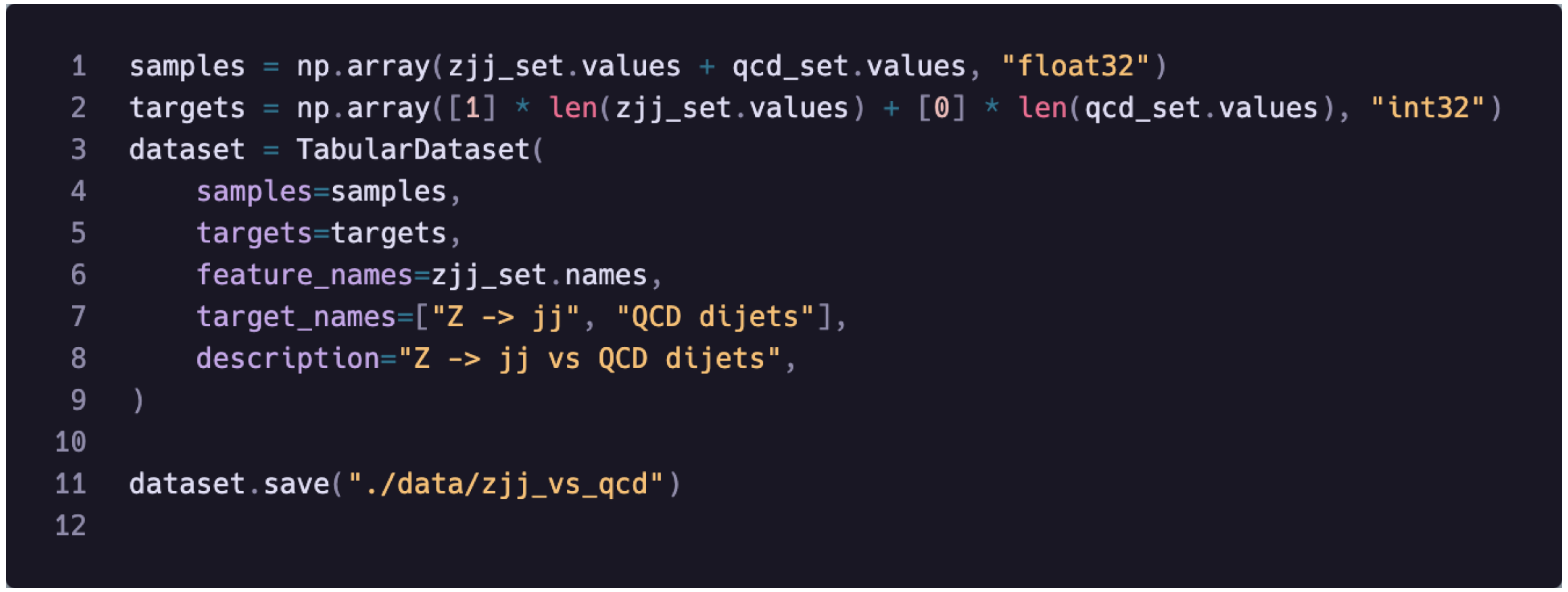
- **from_output** to load the existing runs.
- **get_observable** to parse the name.
    - Could pass strings directly to **Set** here.
- **passed** to check if the event is valid to preselection conditions



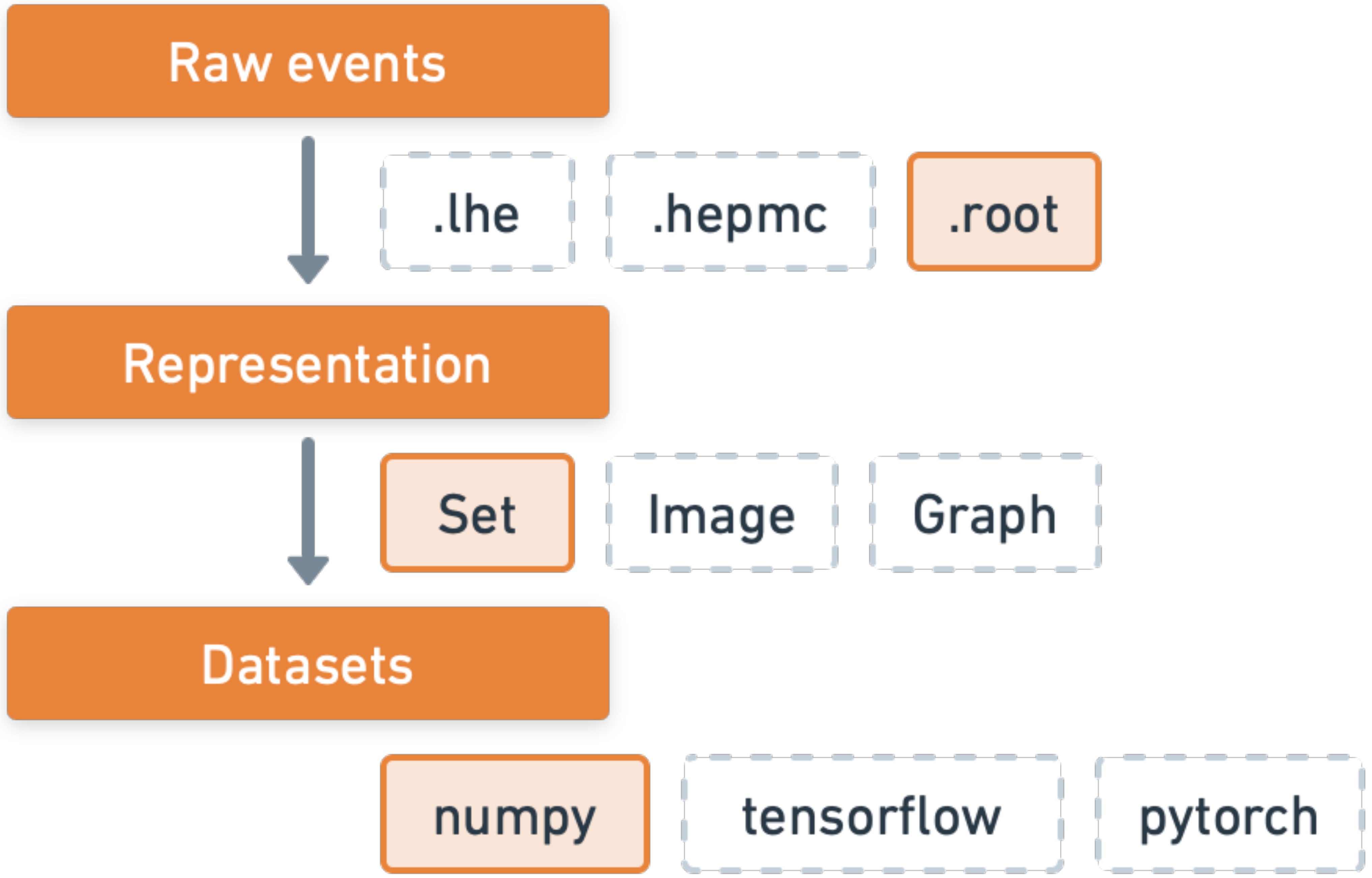
```python
zjj = Madgraph5.from_output("./data/pp2zz_z2jj_z2vlvl")

preselections = Filter(["FatJet.Size > 0", "Jet.Size > 1"])

zjj_set = Set(
    [
        get_observable("FatJet_0.Mass"),
        get_observable("FatJet_0.TauMN", m=2, n=1),
        get_observable("Jet_0-Jet_1.DeltaR"),
    ]
)

zjj_bar = Progbar(zjj.runs[0].n_events)
for i, event in enumerate(zjj.runs[0].events):
    if preselections.read_event(event).passed():
        zjj_set.read_event(event)

    zjj_bar.update(i + 1)

```

Quick start

Create datasets



```
samples = np.array(zjj_set.values + qcd_set.values, "float32")
targets = np.array([1] * len(zjj_set.values) + [0] * len(qcd_set.values), "int32")
dataset = TabularDataset(
    samples=samples,
    targets=targets,
    feature_names=zjj_set.names,
    target_names=["Z -> jj", "QCD dijets"],
    description="Z -> jj vs QCD dijets",
)

dataset.save("./data/zjj_vs_qcd")

```



Raw events

.lhe

.hepmc

.root

Representation

Set

Image

Graph

Datasets

numpy

tensorflow

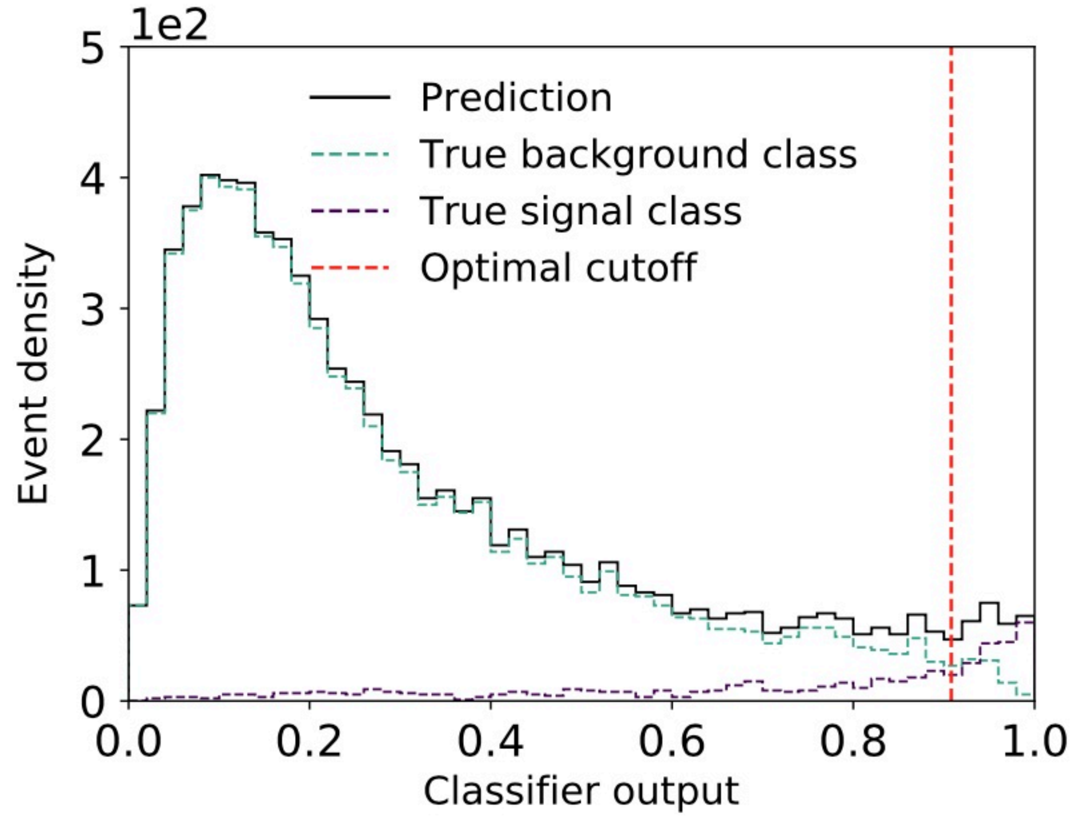
pytorch

- **TabularDataset** for **Set**.
- Currently supports saving to **.npz** files

Quick start

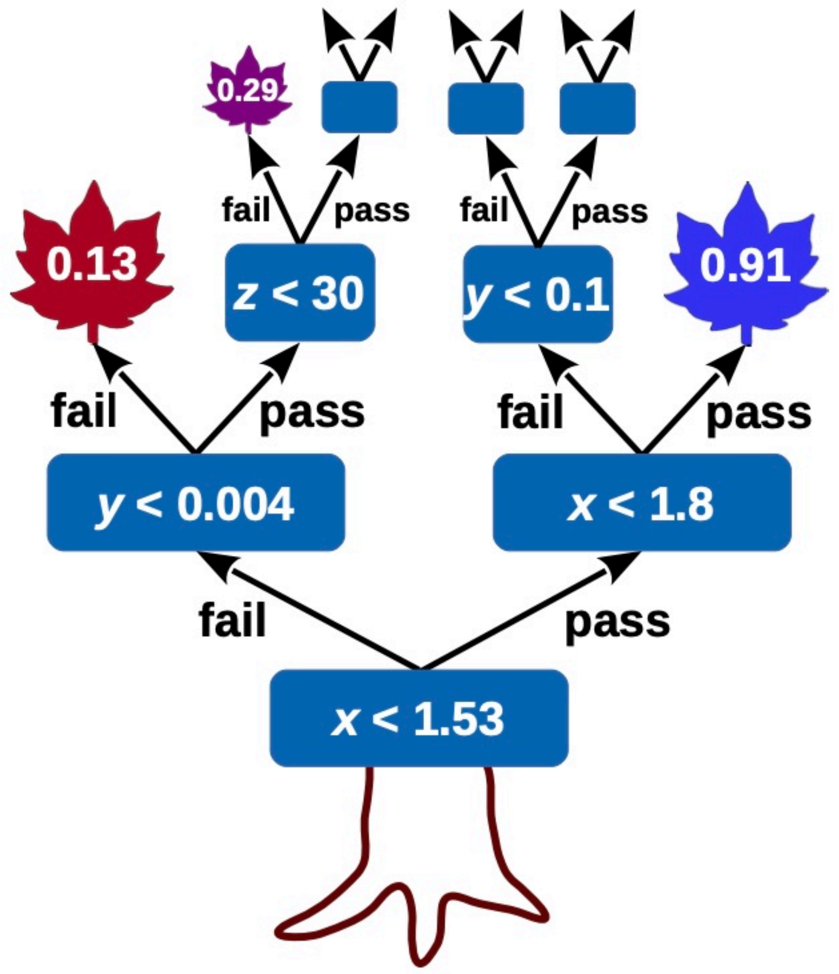
Apply approaches

- Three most-used approaches
  - cuts
  - trees
  - neural networks

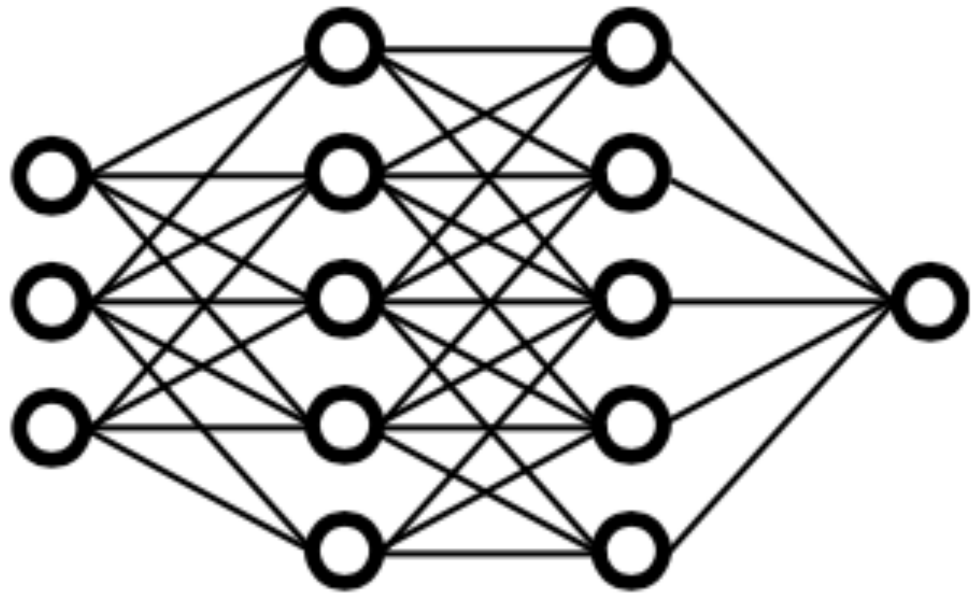


(a) XGBoost with optimized cutoff at 0.9081.

[2108.03125] Beyond Cuts in Small Signal Scenarios



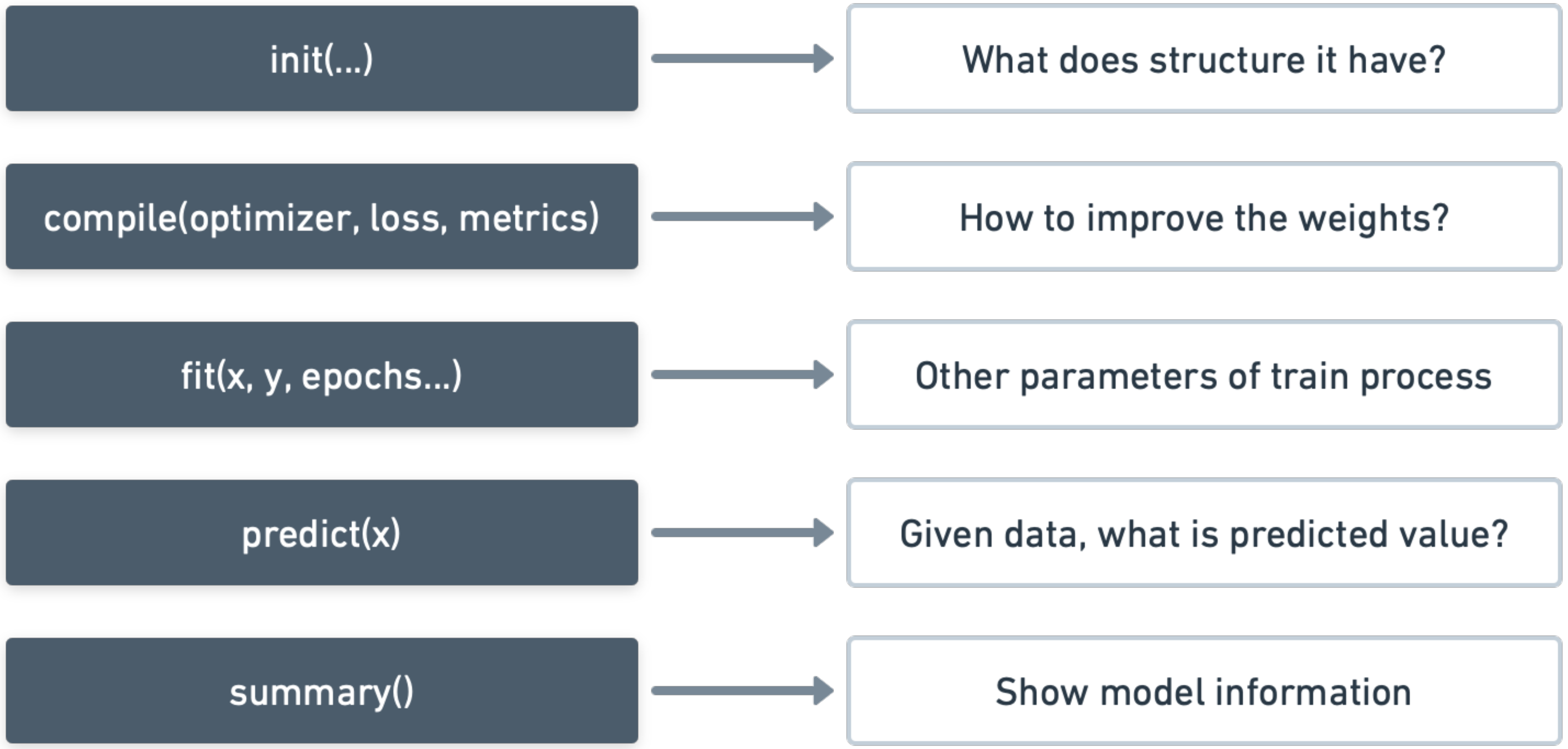
[2206.09645] Boosted decision trees



[1709.04464] Jet Substructure at the Large Hadron Collider

Quick start

Apply approaches



init(...) → What does structure it have?

compile(optimizer, loss, metrics) → How to improve the weights?

fit(x, y, epochs...) → Other parameters of train process

predict(x) → Given data, what is predicted value?

summary() → Show model information

scikit learn

Keras

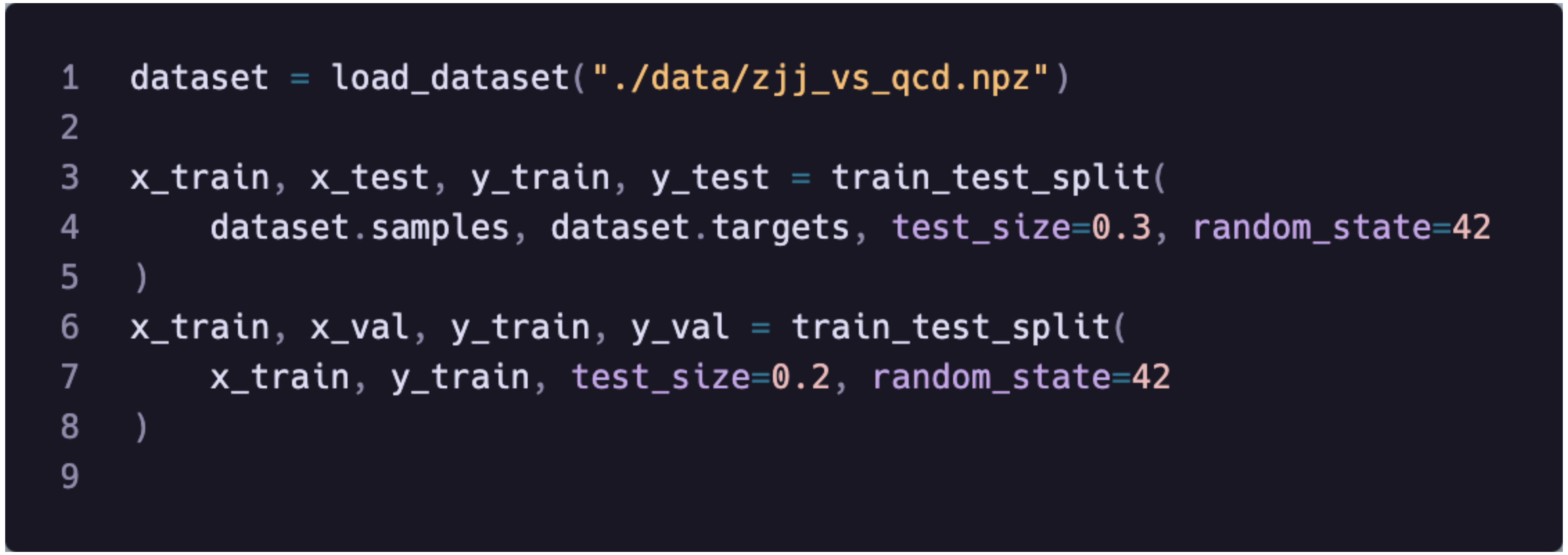
TensorFlow

PyTorch

Approach protocol accepts any models with these member functions

Quick start

Apply methods



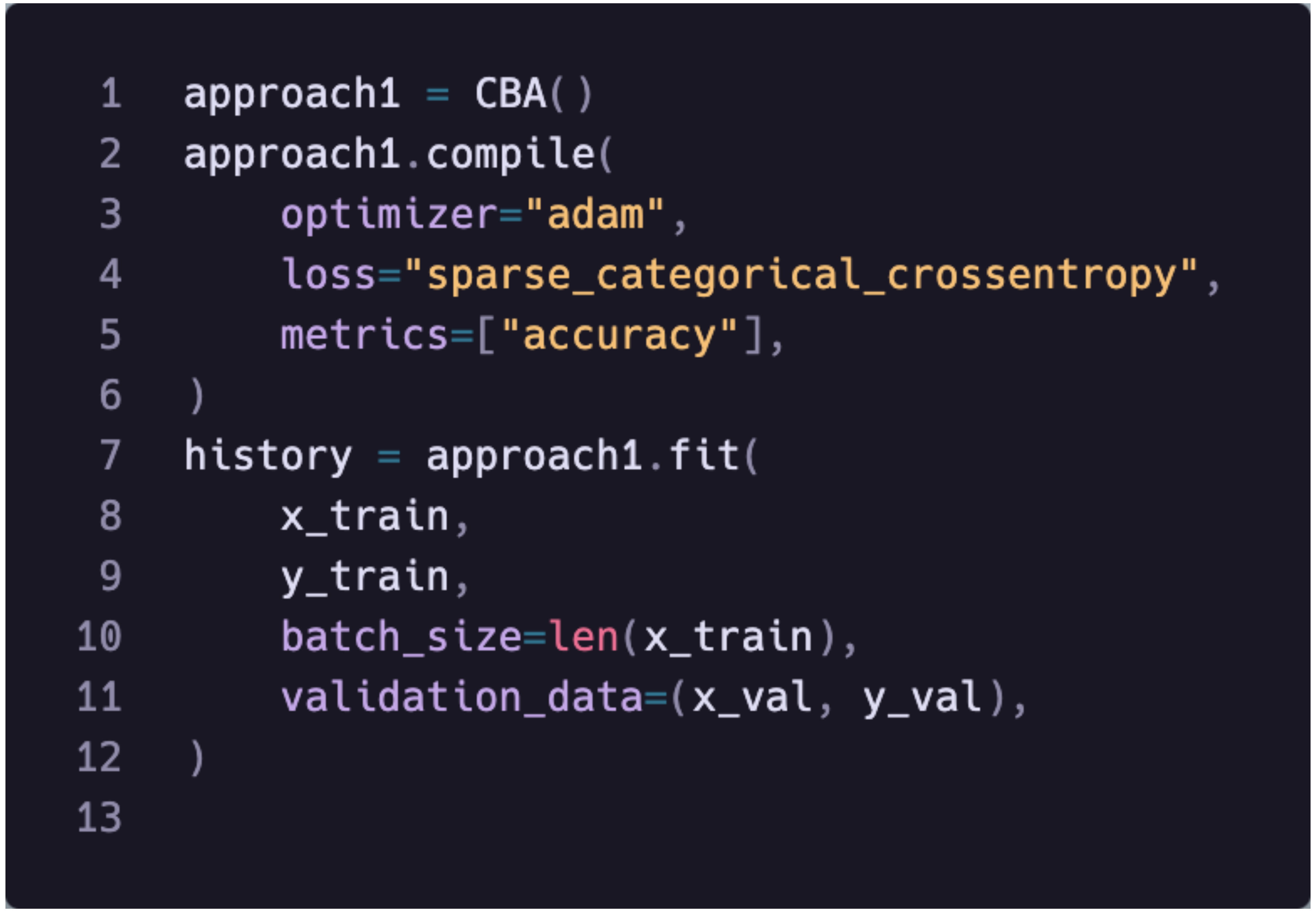
```python
dataset = load_dataset("./data/zjj_vs_qcd.npz")

x_train, x_test, y_train, y_test = train_test_split(
    dataset.samples, dataset.targets, test_size=0.3, random_state=42
)
x_train, x_val, y_train, y_val = train_test_split(
    x_train, y_train, test_size=0.2, random_state=42
)

```

- Load dataset from previous saved location.
- Split train/val/test sets with fixed random seed.

Quick start

Apply methods

- CutAndCount as a cut-based analysis (CBA).
- **batch_size** should be the whole train set.
- **epochs** have no effect.

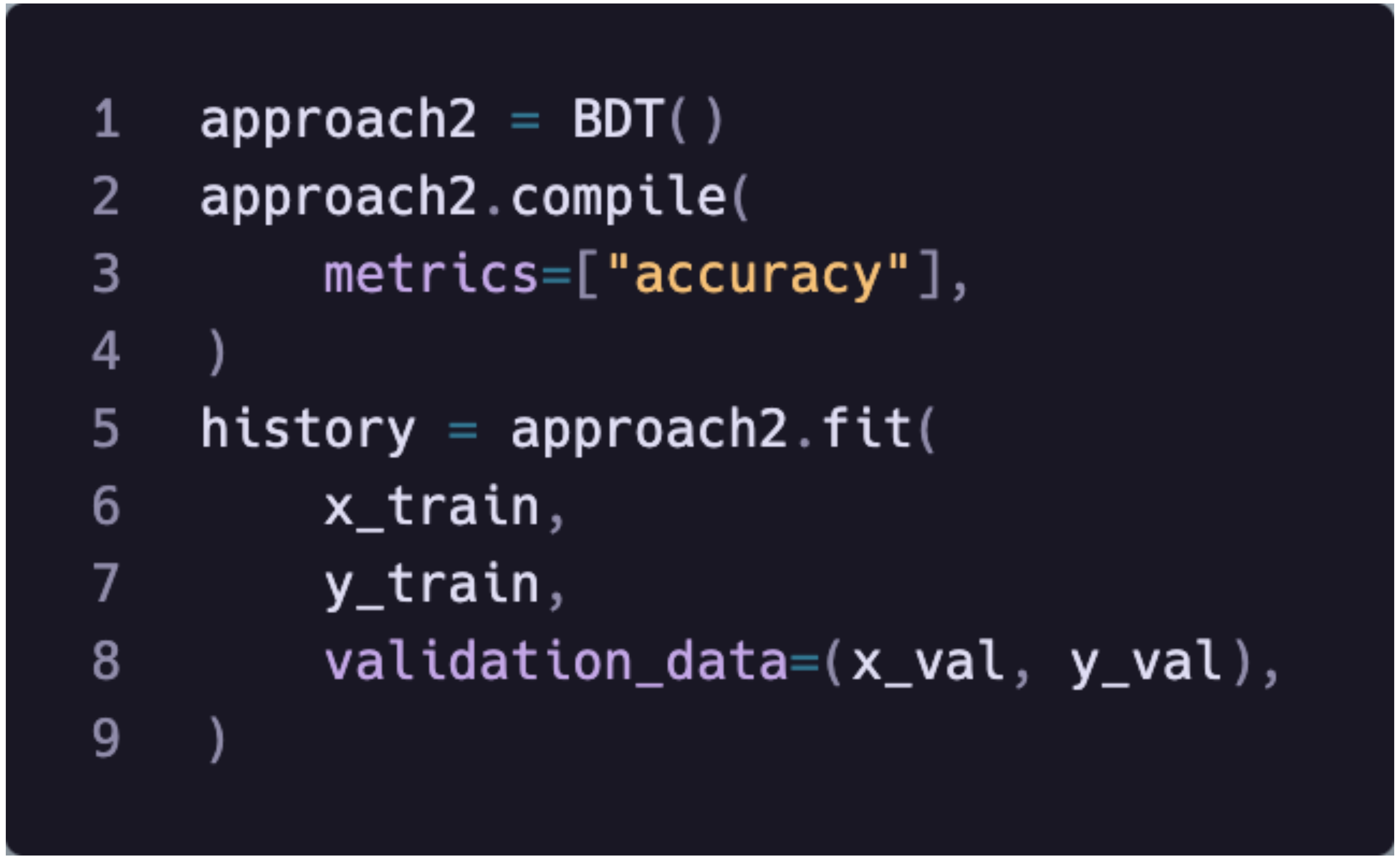


```
approach1 = CBA()
approach1.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
history = approach1.fit(
    x_train,
    y_train,
    batch_size=len(x_train),
    validation_data=(x_val, y_val),
)

```

Quick start

Apply methods

- **GradientBoostedDecisionTree** as a boosted decision tree (BDT).
- **optimizer**, **loss** have no effect in **compile**.
- **batch_size**, **epochs** have no effect in **fit**.

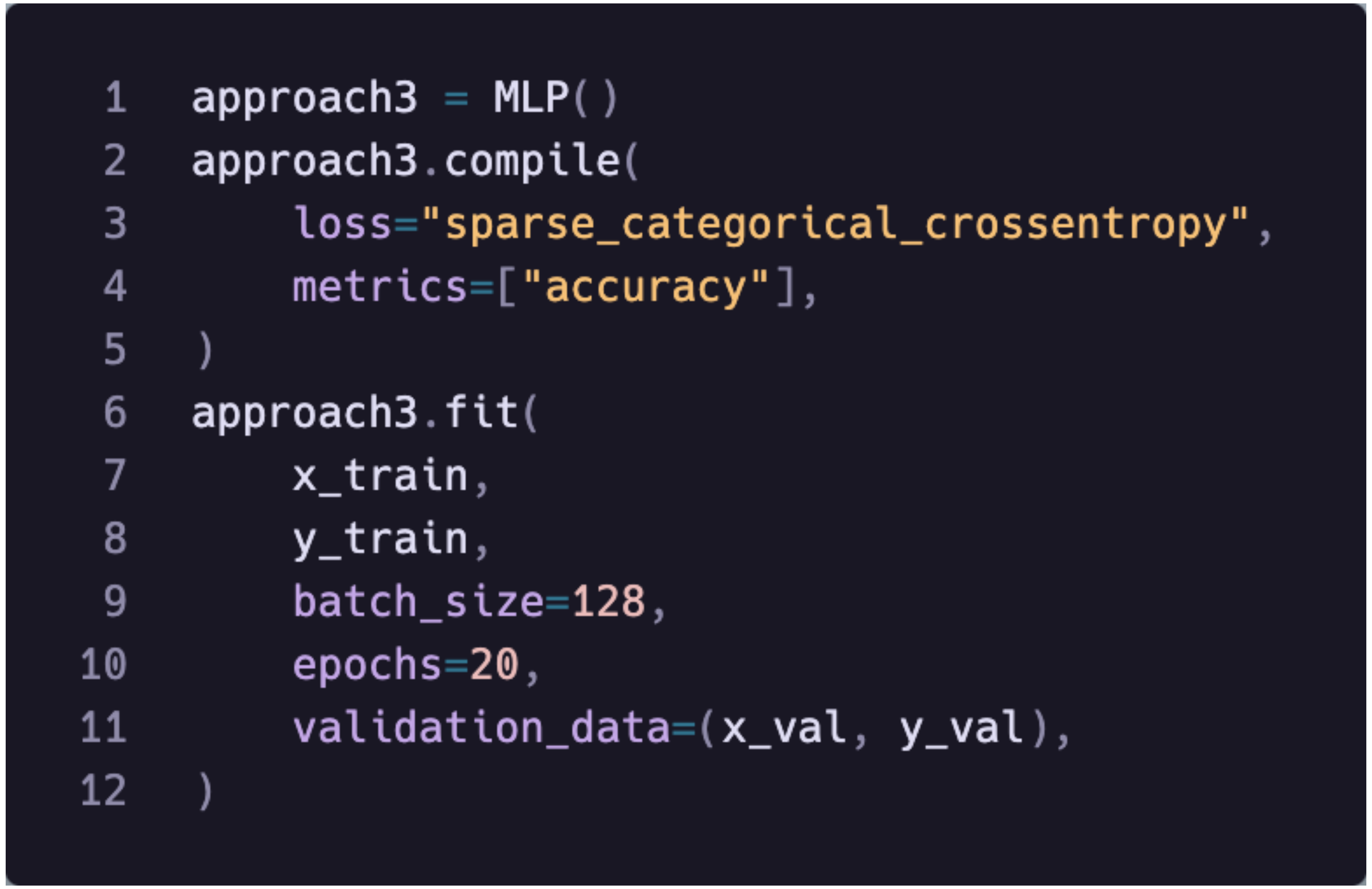


```
approach2 = BDT()
approach2.compile(
    metrics=["accuracy"],
)
history = approach2.fit(
    x_train,
    y_train,
    validation_data=(x_val, y_val),
)
```

Quick start

Apply methods

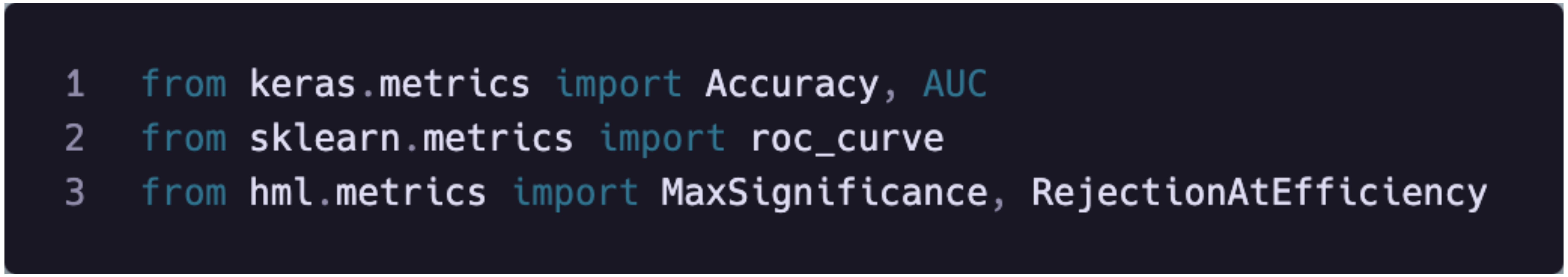
- A built-in **ToyMultilayerPerceptron** as MLP for demonstration.
- Normal **Keras** models.



```
approach3 = MLP()
approach3.compile(
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
approach3.fit(
    x_train,
    y_train,
    batch_size=128,
    epochs=20,
    validation_data=(x_val, y_val),
)
```

Quick start

Apply methods



```python
from keras.metrics import Accuracy, AUC
from sklearn.metrics import roc_curve
from hml.metrics import MaxSignificance, RejectionAtEfficiency
```

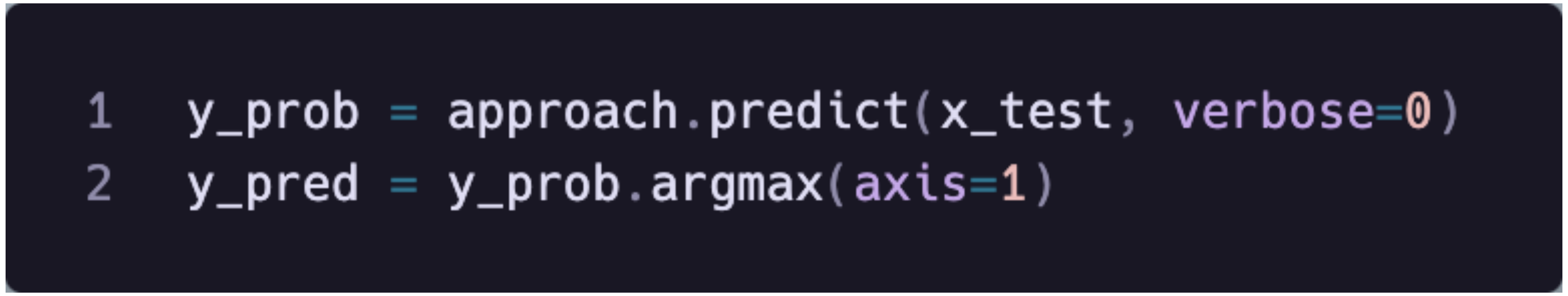
- **MaxSignificance** calculates the maximum significance under uniform distributed thresholds.

$$\text{significance} = \sqrt{S/\left(S+B\right)}$$

- **RejectionAtEfficiency** $(1/\varepsilon_b$ at $\varepsilon_s = 50\,\%)$ calculates the background rejection at a given signal efficiency.

Quick start

Apply methods

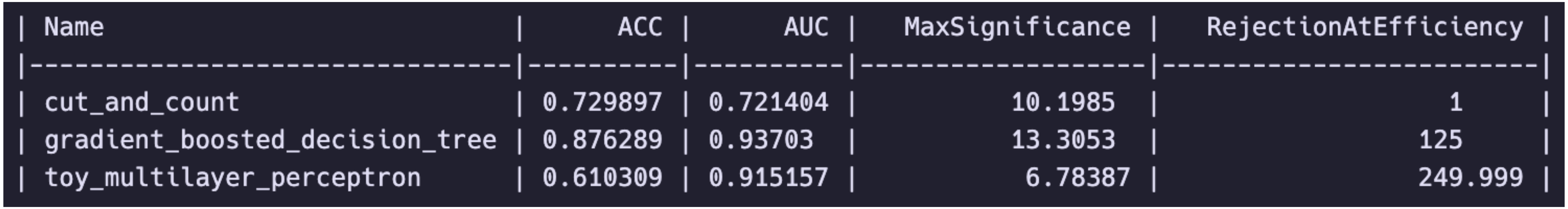


```
y_prob = approach.predict(x_test, verbose=0)
y_pred = y_prob.argmax(axis=1)
```

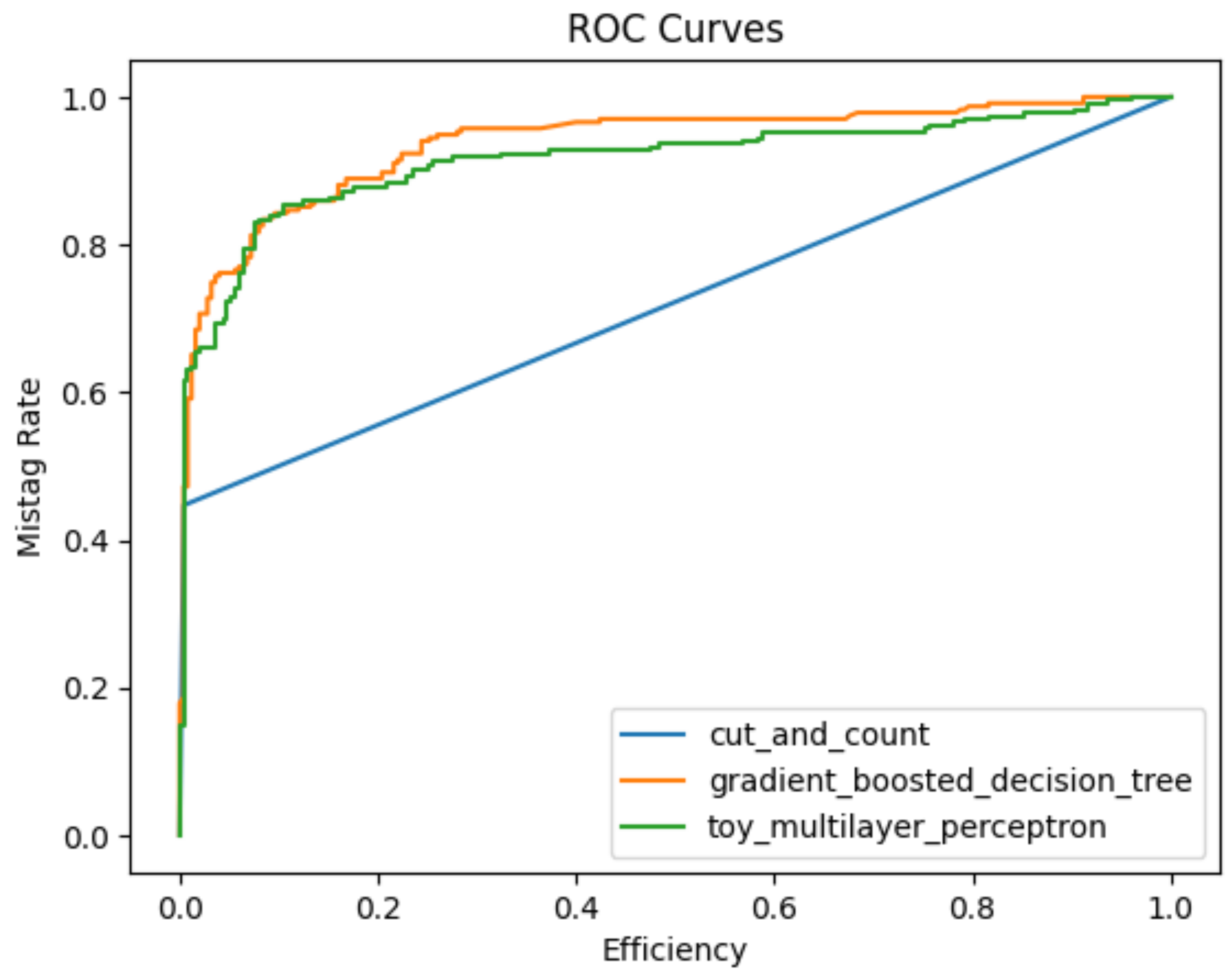
- Get class probability and predictions to evaluate all approaches.

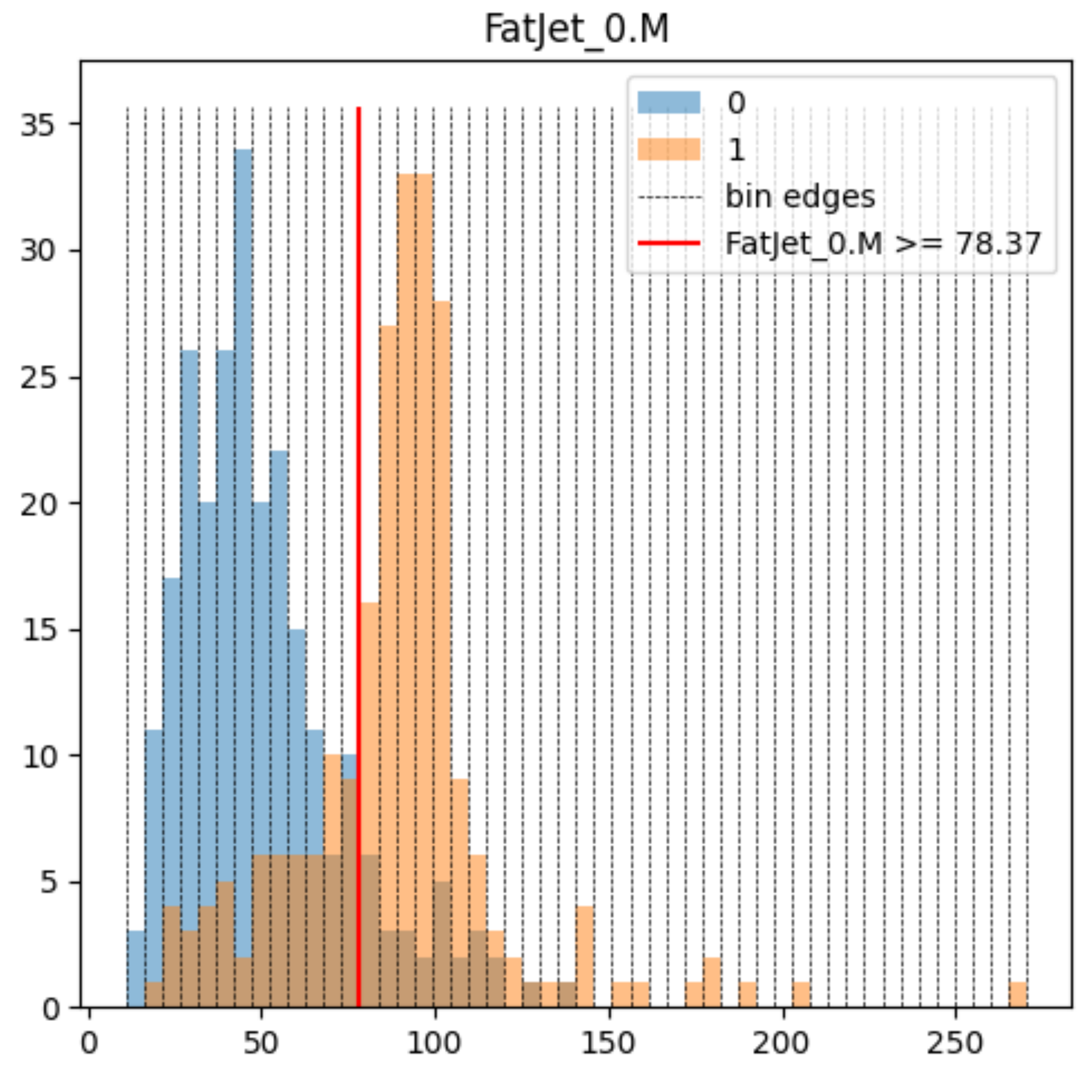
Quick start

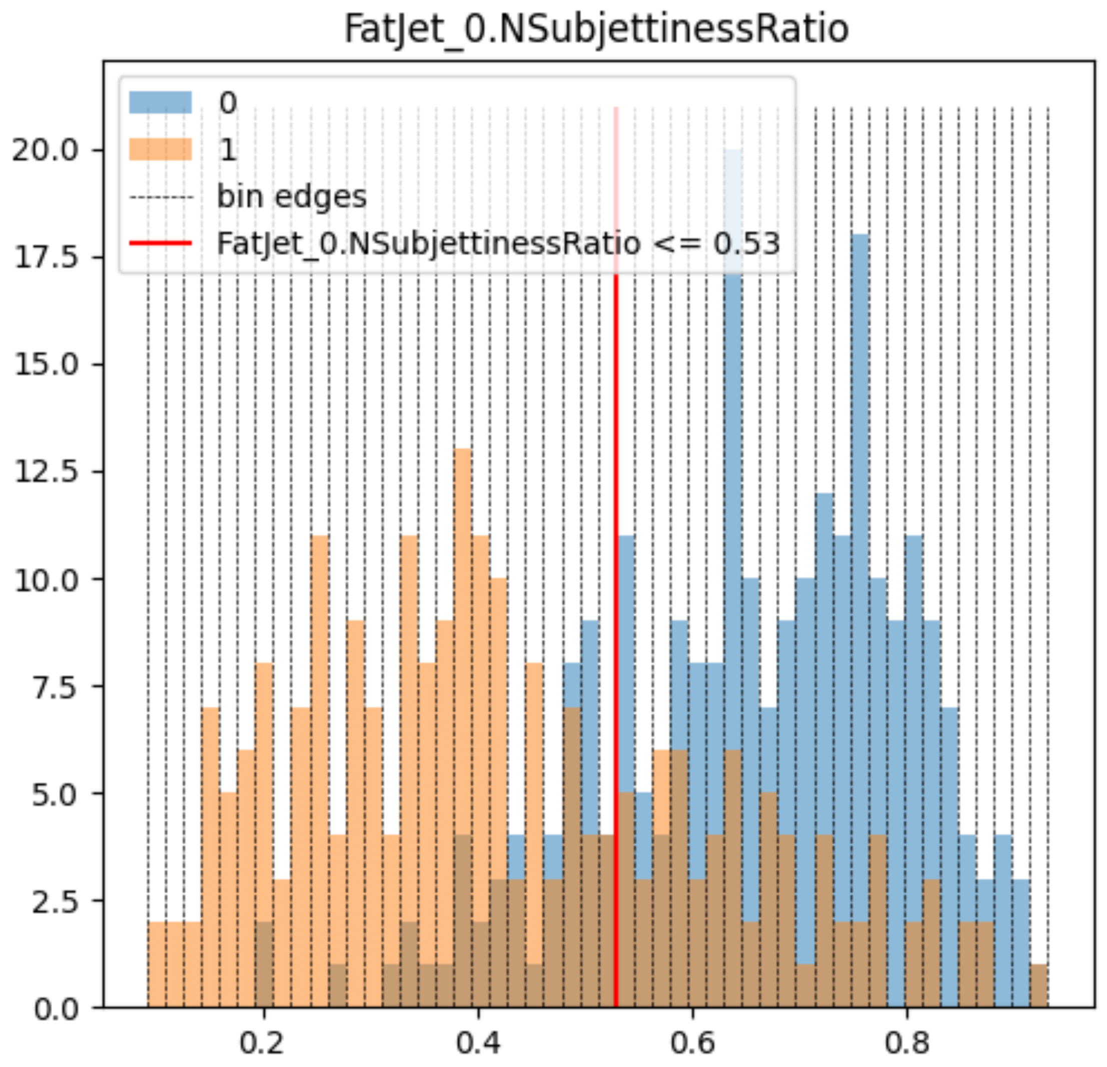
Apply methods

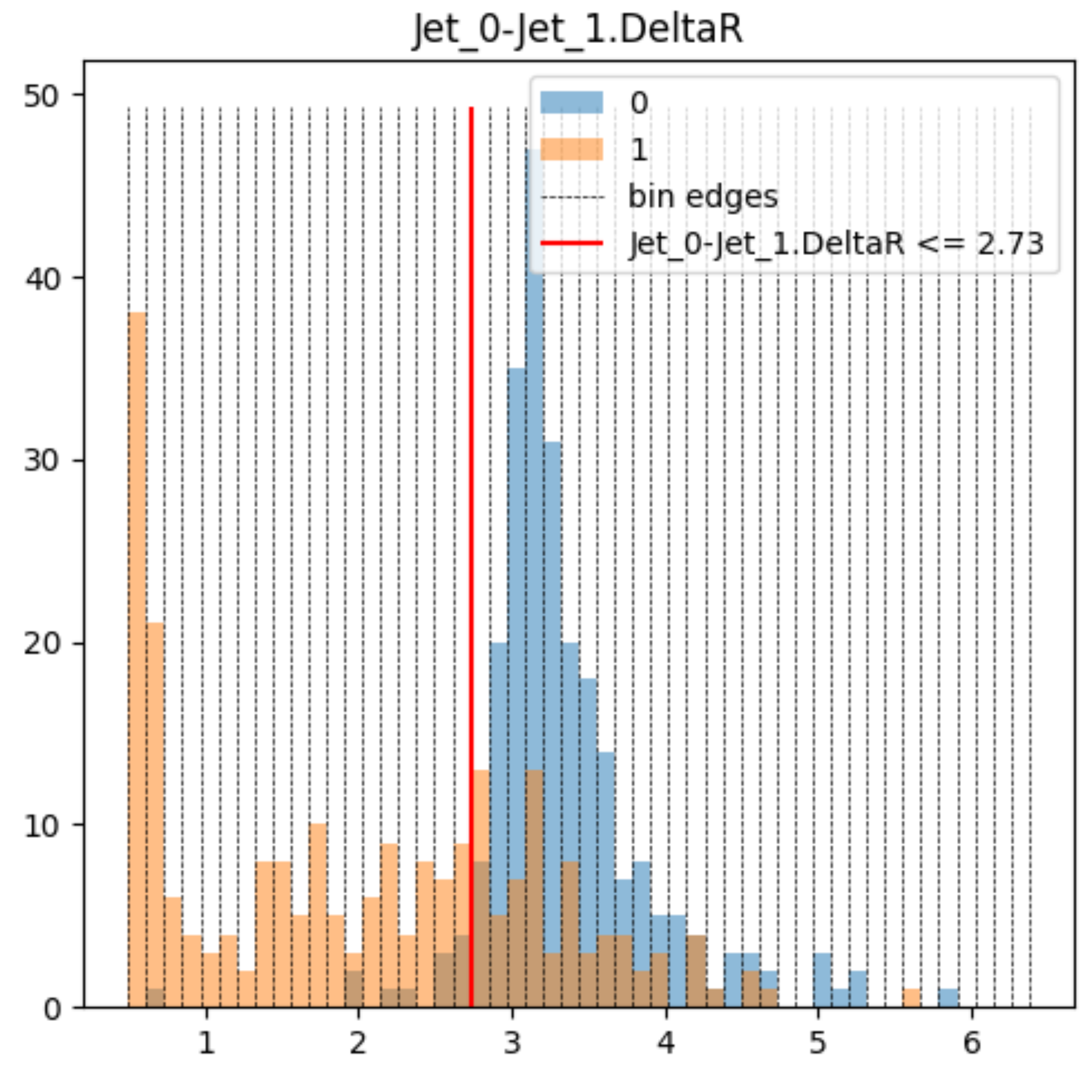


```
| Name                          |      ACC |      AUC |   MaxSignificance |   RejectionAtEfficiency |
|-------------------------------|----------|----------|-------------------|-------------------------|
| cut_and_count                 | 0.729897 | 0.721404 |           10.1985 |                       1 |
| gradient_boosted_decision_tree | 0.876289 | 0.93703  |           13.3053 |                     125 |
| toy_multilayer_perceptron     | 0.610309 | 0.915157 |           6.78387 |                 249.999 |
```









Future

Roadmap

- 0.3.x
  - Support loading parts of existing datasets from Zenodo, Hugging Face, GitHub.
  - Support image and graph representation and ToyCNN, ToyGNN to test.
- 0.4.*
  - Support for frameworks from **Scikit-HEP**

Future

Roadmap

- WELCOME your priceless contributions ❤️
- WELCOME any comments to my email: star9daisy@outlook.com
- Check the documents for more details: https://star9daisy.github.io/hep-ml-lab/
- Find source code here: https://github.com/Star9daisy/hep-ml-lab
- Have a try today! pip install hep-ml-lab

THANK YOU!