INTRODUCTION TO COMPUTATIONAL PHYSICS

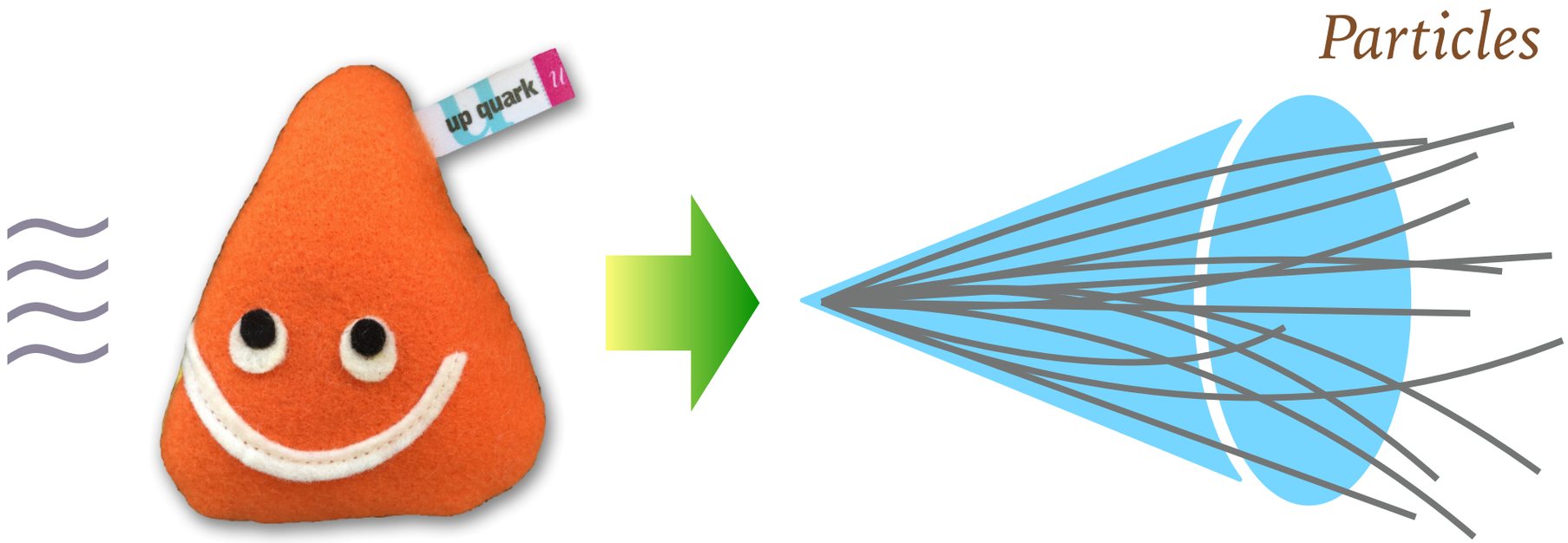
Kai-Feng Chen
National Taiwan University

WHAT ARE WE GOING TO DO?

- We will have a **Machine Learning Competition**, as well!
- The subject is to classify particle data, e.g. a collection of high momentum **Higgs boson** events versus high momentum **light quarks**, as measured in the detector.
- We are going to run this competition, and who obtains the **higher accuracy** from the test sample will win!

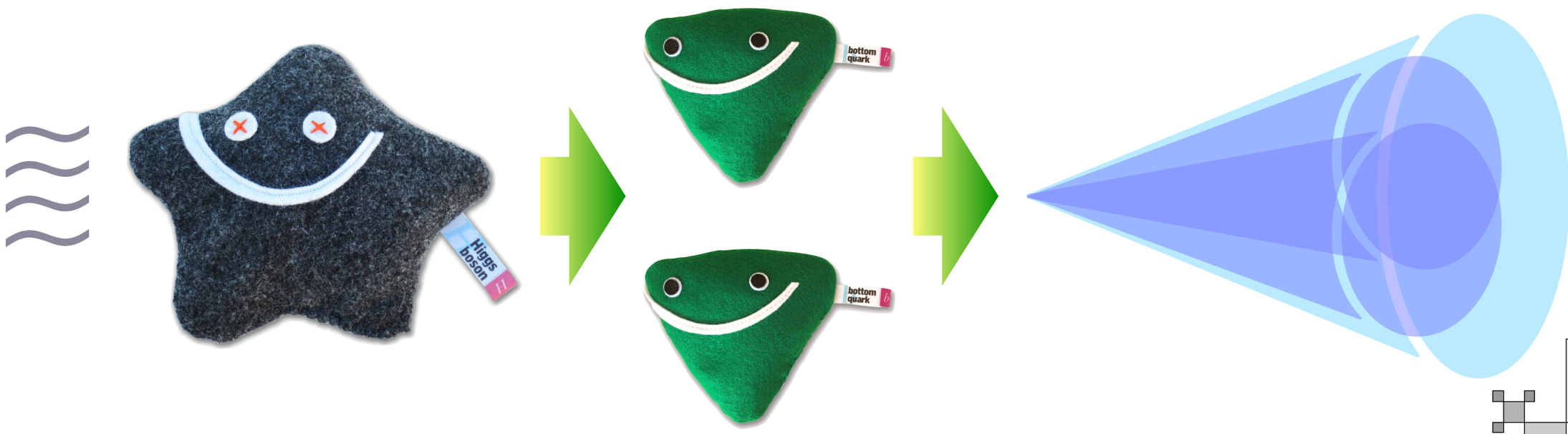
A LITTLE BIT OF PHYSICS

- When a (*high-momentum*) quark is produced, it will hadronize and produce a lot of particles afterwards. These particles would form an object which is called a **"Jet"**, which is the product we can see in the experiments.
- A typical description of the jet is a collection of particles distributed in a cone. The particles are mostly hadrons, such as kaons, pions, can be either charged or neutral.



A LITTLE BIT OF PHYSICS (II)

- When a high-momentum Higgs boson decaying into two bottom quarks, both quarks will hadronize into jets, too.
- However, since the initial momentum of the Higgs boson is high ("boosted"), the open angle of the two jets is small, and would merge into single jet in the end, e.g.
- So the Higgs jet should preserve many physics properties of the original particle, for example, the invariant mass!



THE DATA SAMPLES

- In this competition we provide the following samples:
  - **`sample_hbb.csv`**: contains 40,000 jets consisting of high momentum Higgs bosons decaying into bottom quark pairs (as signal, class = 1).
  - **`sample_q.csv`**: contains 40,000 jets from high momentum light quark (as background, class = 0).
  - **`sample_test.csv`**: contains another 40,000 jets to be processed with your machine learning algorithm.
  - **`my_predicts.csv`**: a sample submission file in the correct format of two fields:
    - ➡ **id** - an id unique to an event (0-39999).
    - ➡ **class** - the predicted classes (1 or 0)

for example:

```
id,class
0,0
1,0
2,1
3,1
. . . . . .
```

DATA FIELDS

- **id** - an id unique to an event (0-39999). All of the particles from the same event shared the same event id.
- **code** - the particle code/type:
  - **0** - photon, charge = 0
  - **1** - electron, charge = –1
  - **2** - position (anti-electron), charge = +1
  - **3** - muon, charge = –1
  - **4** - anti-muon, charge = +1
  - **5** - negatively-charged hadron (*pion, kaon, or antiproton*), charge = –1
  - **6** - positively-charged hadron (*pion, kaon, or proton*), charge = +1
  - **7** - neutral hadron (*long-lived kaon, neutron*), charge = 0
- **px** - particle momentum in x direction in unit GeV/c
- **py** - particle momentum in y direction in unit GeV/c
- **pz** - particle momentum in z direction in unit GeV/c

Common format for the training & testing samples

DATA FIELDS (II)

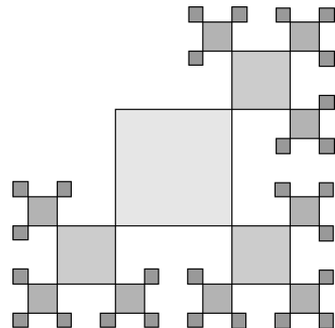
- Here are the snapshot of a couple of events:

# of particles can vary from event by event. The particles belonging to **the same event share the same id** (*the first column*), follow by the particle code, and the 3 momentum.

```
id,code,px,py,pz
0,6,5.5367,-1.4176,-24.3148
0,0,6.8353,-1.6222,-28.9080
0,0,1.2937,-0.3254,-5.3807
0,7,6.1948,2.7286,-14.6407
. . . . . .
0,0,6.9013,-1.3215,-14.3173
0,5,18.7745,-4.1912,-39.6835
0,6,13.9138,-2.8956,-29.0454
0,0,440.5334,-90.8746,-929.1404
1,5,-0.3405,0.2561,-0.0717
1,5,-0.1905,1.5290,-0.4358
1,0,-0.1894,1.0457,-0.2643
1,0,-1.2817,1.7284,-2.0255
. . . . . .
1,0,-11.9429,31.8883,-20.1425
1,5,-13.6119,34.6560,-21.9784
1,6,-51.8062,133.2360,-87.5168
1,5,-11.4774,30.1408,-19.4610
. . . . . .
```

Event #0

Event #1



A DEMO EXAMPLE: LDA W/ 2 FEATURES

- Convert the input data to the input features:

partial lda_demo.py

```
def load_samples(filename):
    print('load from',filename)
    fin = open(filename)
    lines = fin.readlines()
. . . . . .
    samples.append(evt)
    print(len(samples),'events loaded.')
    return samples

def prepare_features(samples):
    features = []
    for evt in samples:
        px = py = pz = E = 0.
        for p in evt:
            px += p[1]
            py += p[2]
            pz += p[3]
            E += (p[1]**2+p[2]**2+p[3]**2)**0.5
        M = (E**2 - (px**2+py**2+pz**2))**0.5
        features.append([len(evt), M])
    return features
```

← reading the input csv file, store the data as python list.

← phrase the python list into list of two features

← invariant mass of the whole jet

A DEMO EXAMPLE: LDA W/ 2 FEATURES (II)

- The "main" code:

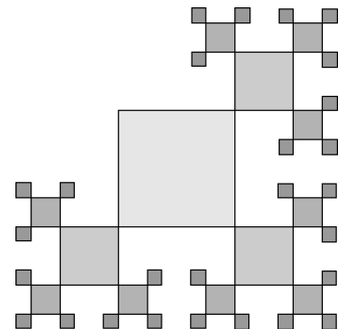
partial lda_demo.py

```
. . . . . .
sample_sig = load_samples('sample_hbb.csv')
sample_bkg = load_samples('sample_q.csv')
sample_test = load_samples('sample_test.csv')

x_train = np.array(prepare_features(sample_sig)+
                   prepare_features(sample_bkg))
y_train = np.array([1]*len(sample_sig)+[0]*len(sample_bkg))
x_test = np.array(prepare_features(sample_test))

clf = LinearDiscriminantAnalysis()
f_train = clf.fit_transform(x_train, y_train)

p_test = clf.predict(x_test)
fout = open('my_predicts.csv','w')
fout.write("id,class\n")
for idx,p in enumerate(p_test):
    fout.write('%d,%d\n' % (idx,p>0.5))
fout.close()
```

← prepare training sample

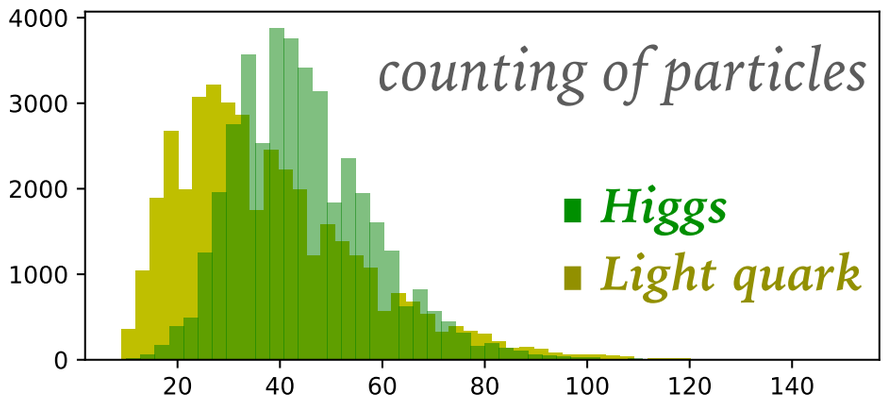
↑ process the testing sample and store the resulting predictions as csv file.

```
id,class
0,0
1,0
2,1
3,1
. . . . . .
```

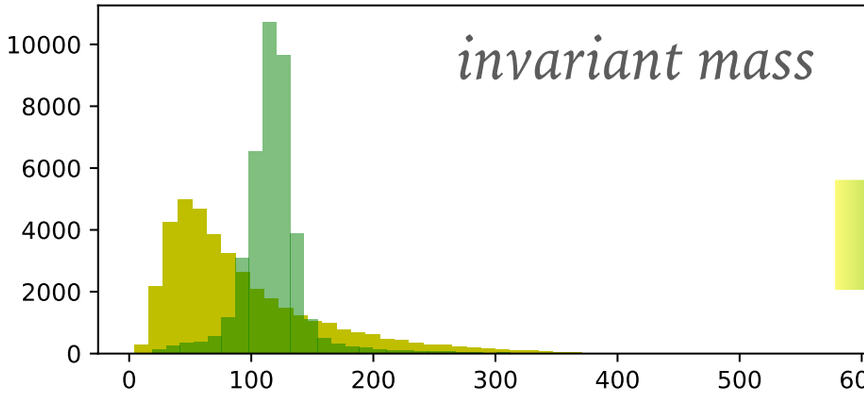



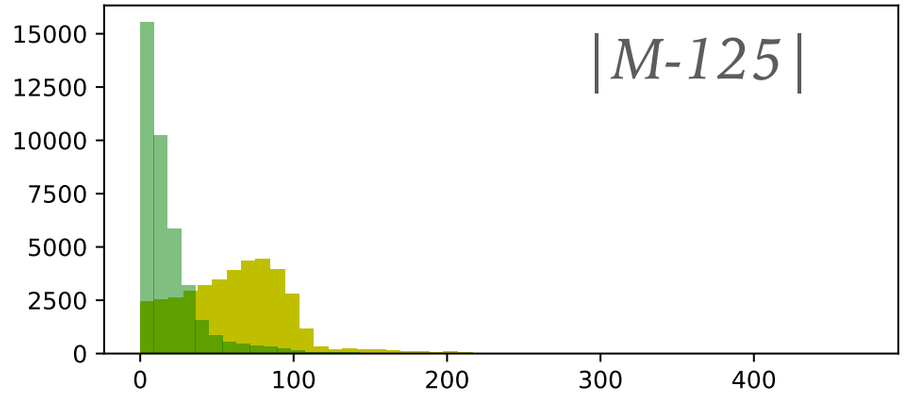
A DEMO EXAMPLE: LDA W/ 2 FEATURES (III)

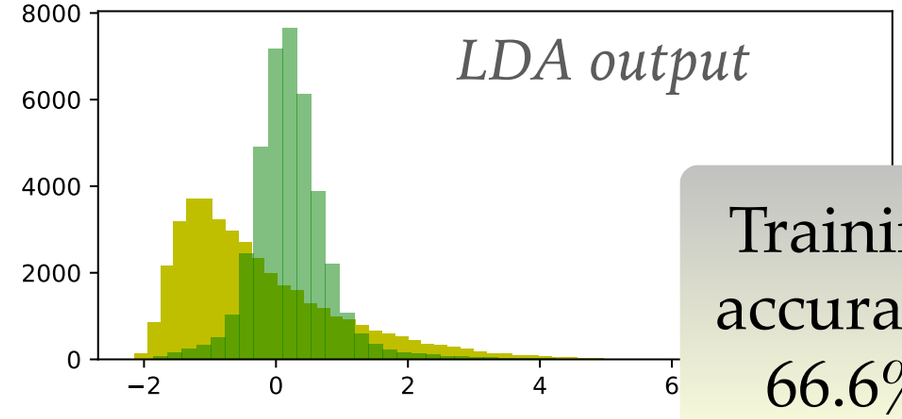
- Output distributions & performance:



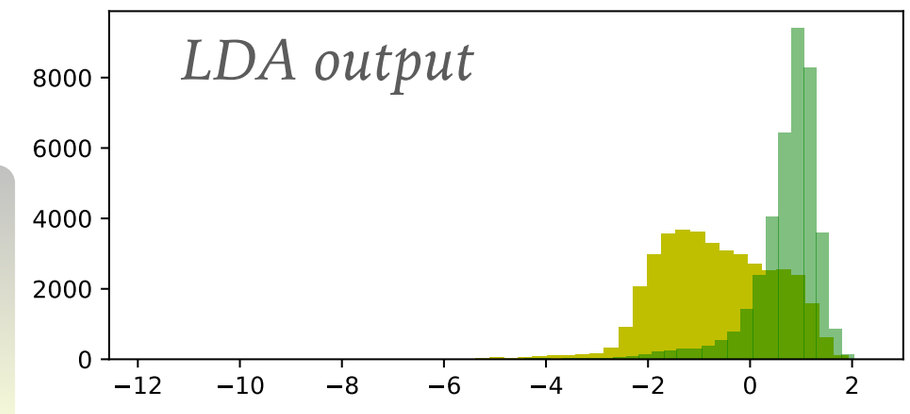
Since LDA can only handle linear combinations, it is better to rearrange the mass variable as given below. This is not required for non-linear models, such as NN (see the ann_demo.py).







Training accuracy: 66.6%

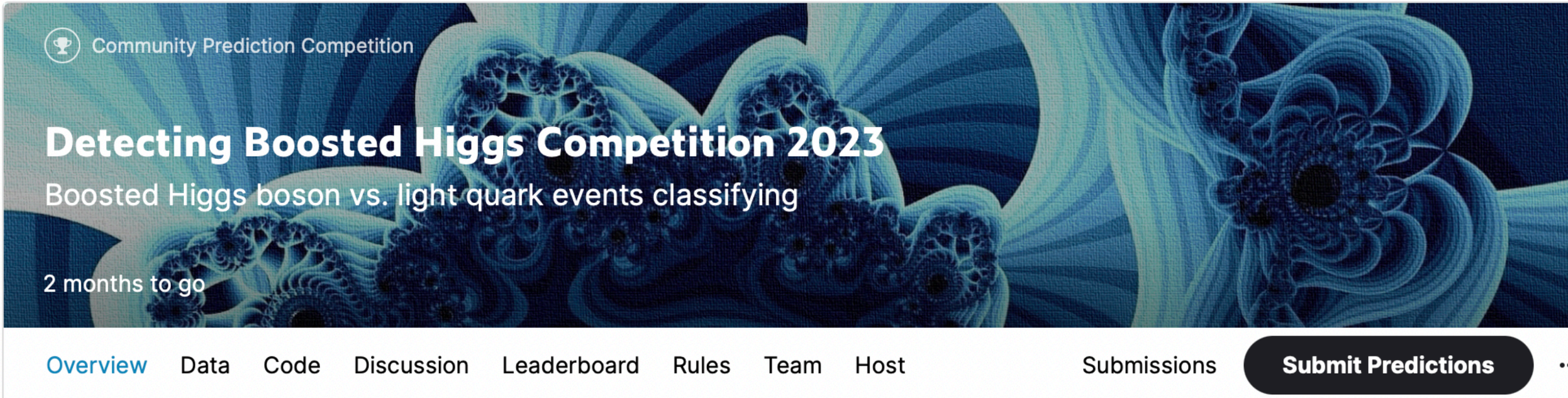


Training accuracy: 79.4%

HOW TO PARTICIPANT

- We are going to use **kaggle**, which is a very popular web for ML competitions (*you can also find many other interesting competitions and data there!*).
- Private link to join the competition:

  https://www.kaggle.com/t/af656b96d99343f88a00f293e618f87f



HOW TO PARTICIPANT – HAVE FUN!

- Please submit your own predictions to the Kaggle website before the deadline **June/16**:
  - Your team name on Kaggle must be start with your student ID and followed by your name, e.g.

    b01234567_XYZ

  - If your accuracy beats half of the participants, you will win a **trophy** (*note, this trophy is the same one for another game competition, even if you win both it only counts once*)!
  - However, if you find your algorithm is so powerful and want to show more, you are encouraged to wrap up as a contribution to the final poster (*this counts*)!