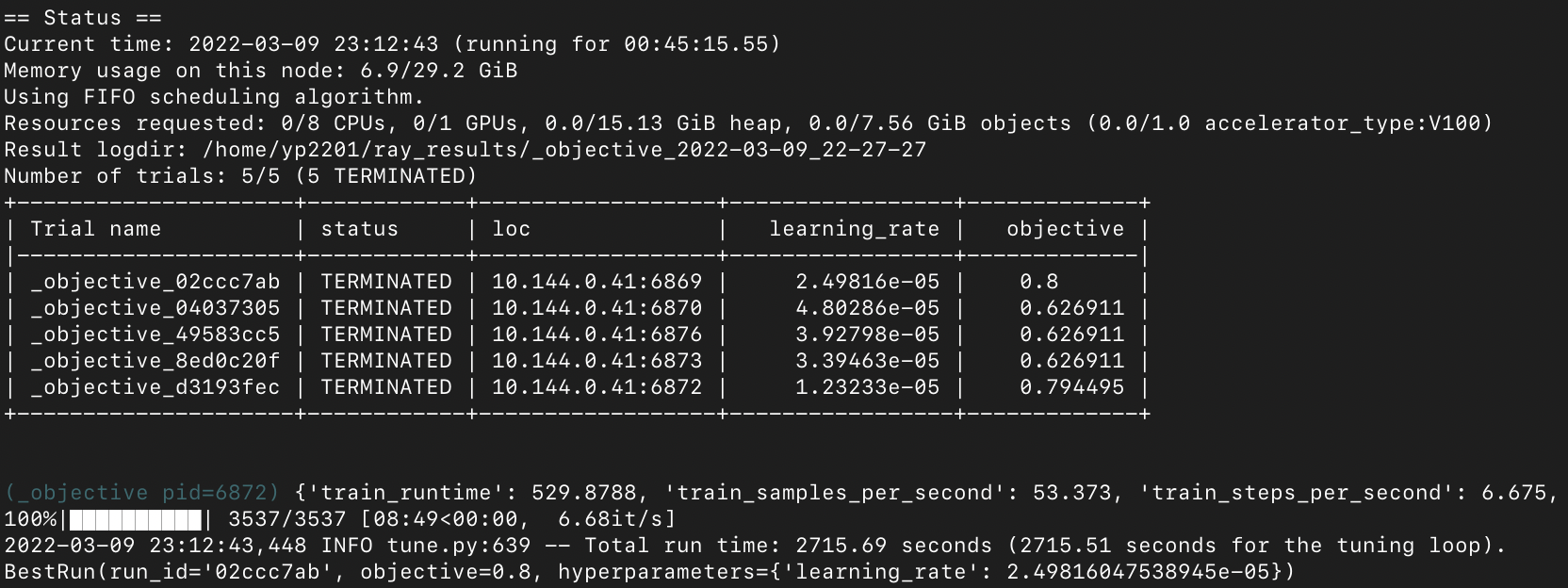
**Reporting results**

Best Run showed as below:

BestRun(run\_id='02ccc7ab', objective=0.8, hyperparameters={'learning\_rate': 2.49816047538945e-05}) 

There were some same results among models, but some were different(Better objective). Objective value didn’t vary a lot, but some were different.

For each models, I didn’t found relationship that are showing higher loss with better evaluation accuracy of f1 score. Detailed result for each models are as below:

1) trial\_id: d3193fec,

eval\_loss: 0.6119393110275269,

eval\_accuracy: 0.7944954128440367

eval\_f1: 0.8386167146974064

2) trial\_id: 8ed0c20f,

eval\_loss: 0.6606119871139526,

eval\_accuracy: 0.6269113149847095

eval\_f1: 0.7706766917293233

3) trial\_id: 49583cc5,

eval\_loss: 0.6606153249740601,

eval\_accuracy: 0.6269113149847095

eval\_f1: 0.7706766917293233

4) trial\_id: 04037305,

eval\_loss: 0.6604805588722229,

eval\_accuracy: 0.6269113149847095,

eval\_f1: 0.7706766917293233

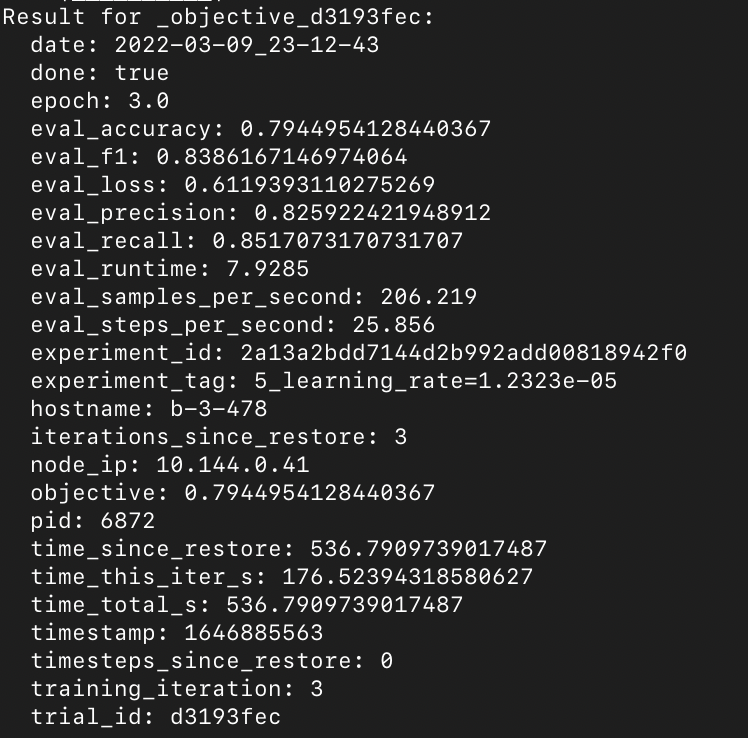
5) trial\_id: 02ccc7ab,

eval\_loss: 0.6439625024795532,

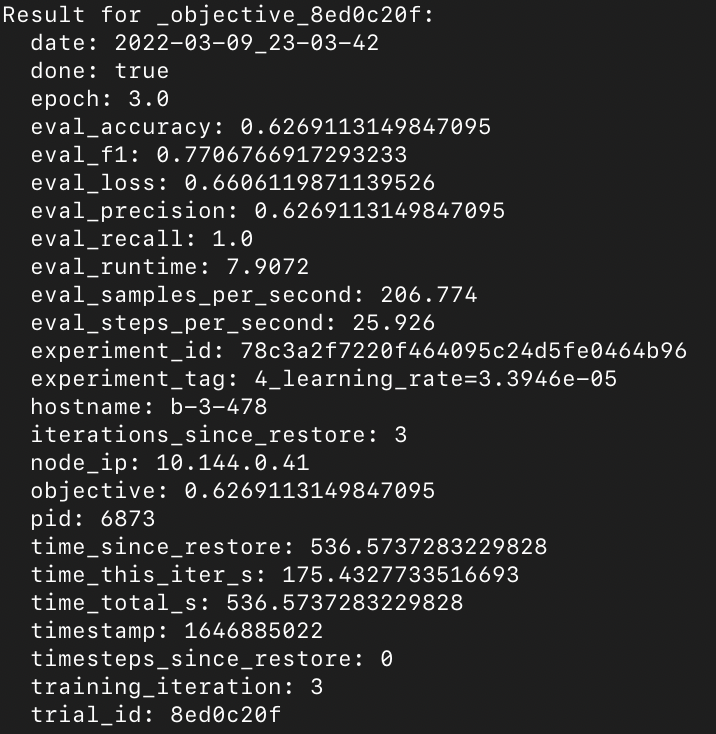
eval\_accuracy: 0.8

eval\_f1: 0.8439140811455847

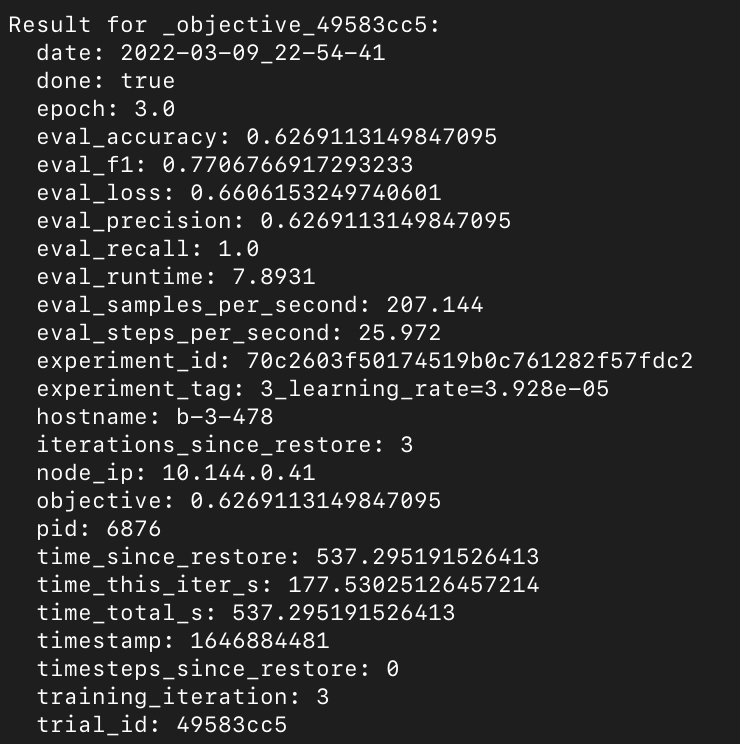
1)

****

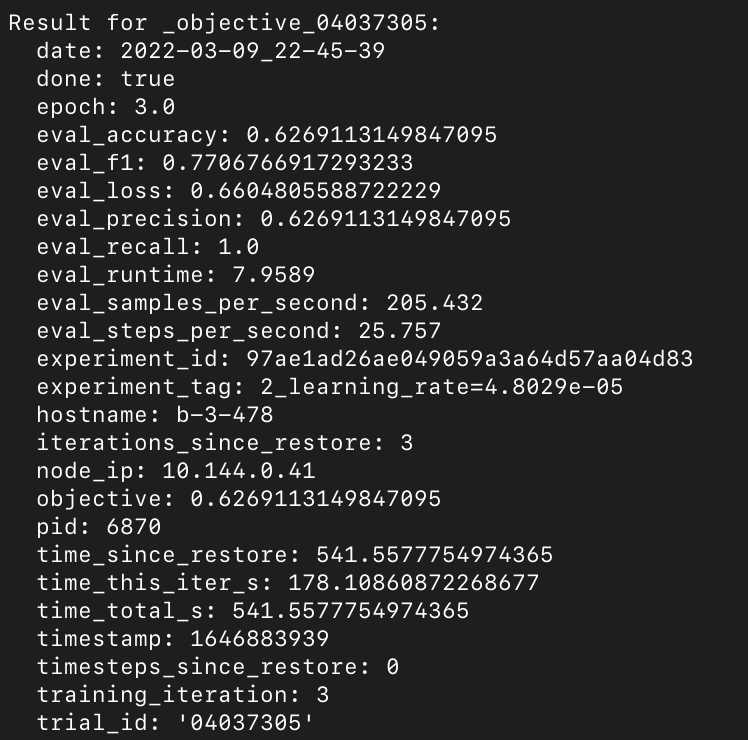
2)

****

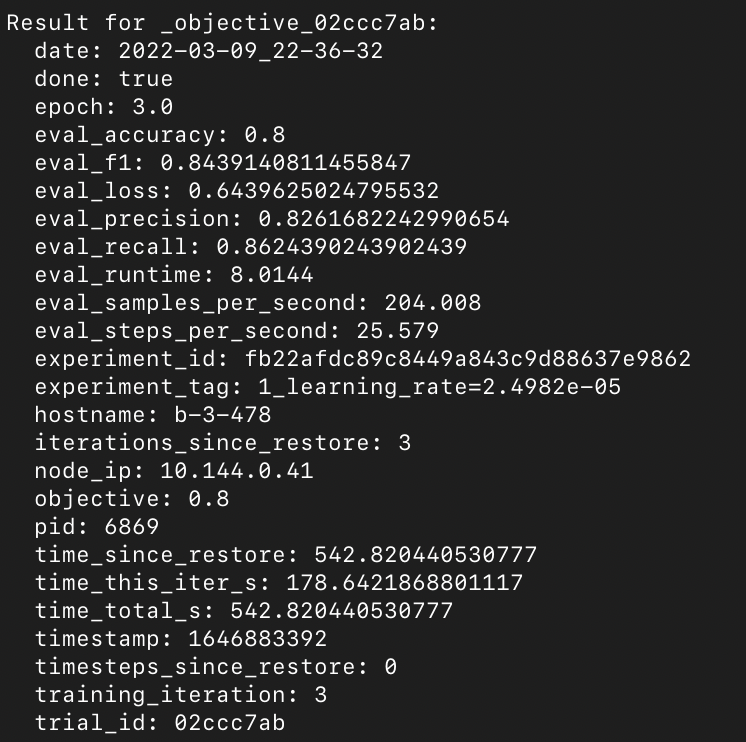
3)



4)

****

5)

****

**\* Extra Credit:**

- I’ve tried (random)grid search, and results were worse than Bayesian optimization.

(Code is included as extra\_credit\_run\_hyperparameter\_search.py)

- Grid Search

Advantage: It is well-known tuning method, and public packages are easy to use

Disadvantage: It tries combination of model in every data without any probability(or likelihood) and this should be inefficient.

- Bayesian optimization:

Advantage: Bayesian optimization should be more close to optimal, as it follows the probability when trying different models

Disadvantage: Public packages can be difficult to use.