Lovely Report

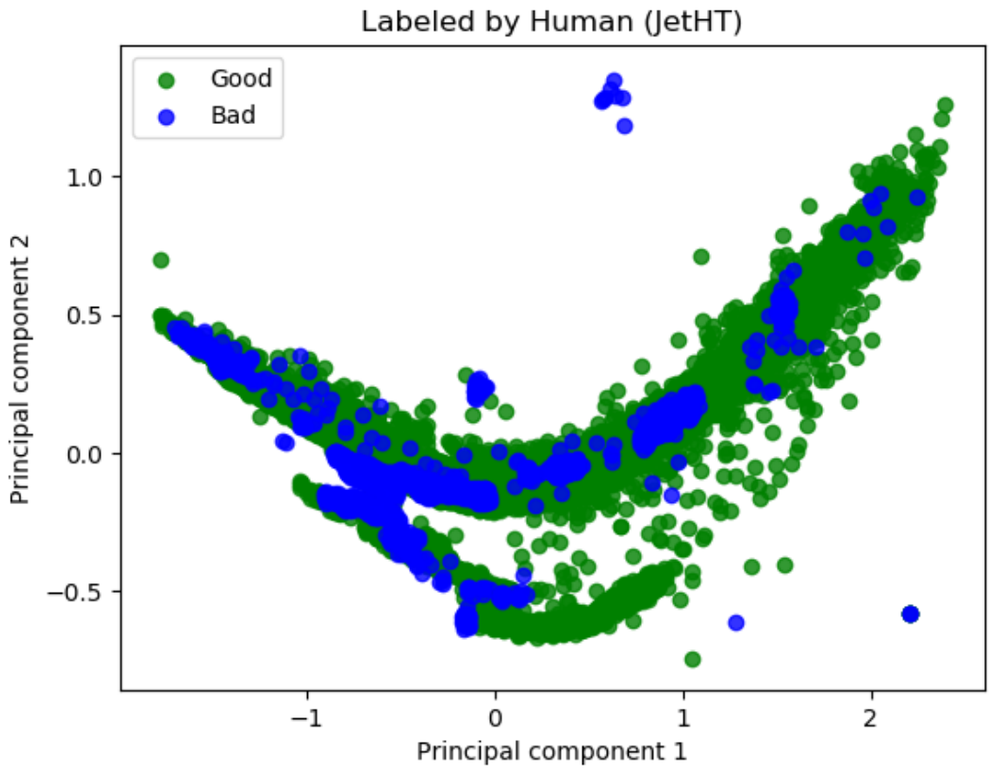
Patomporn (Jab)
CMS-DQM-WHATEVER4DC
15 July 2019

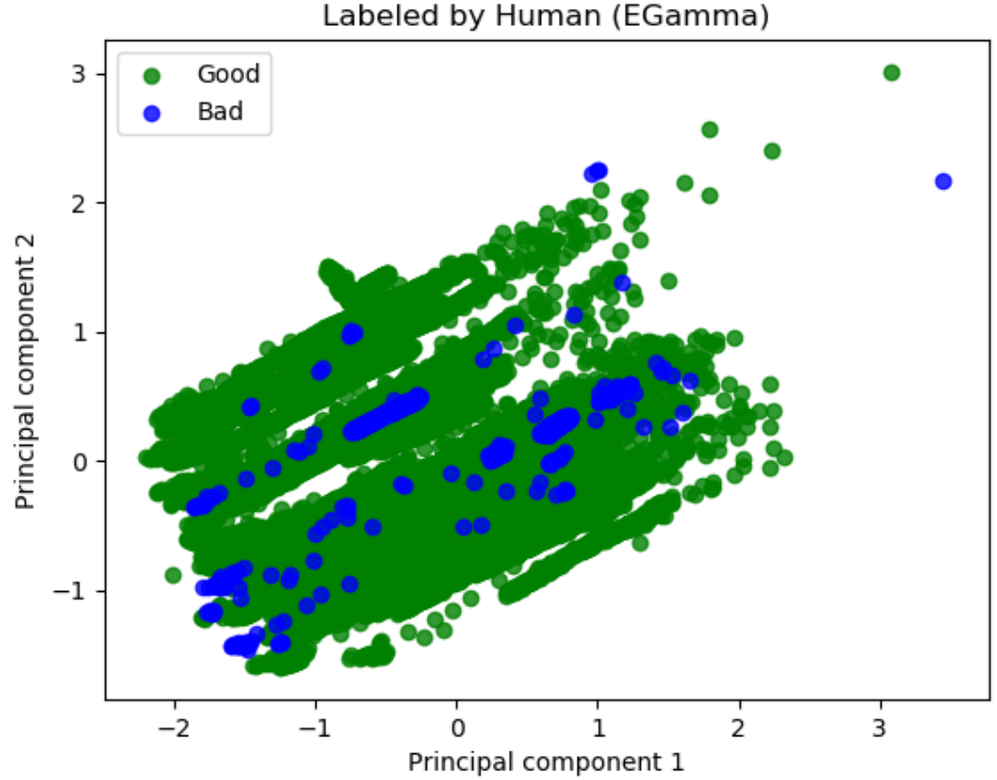
Outline

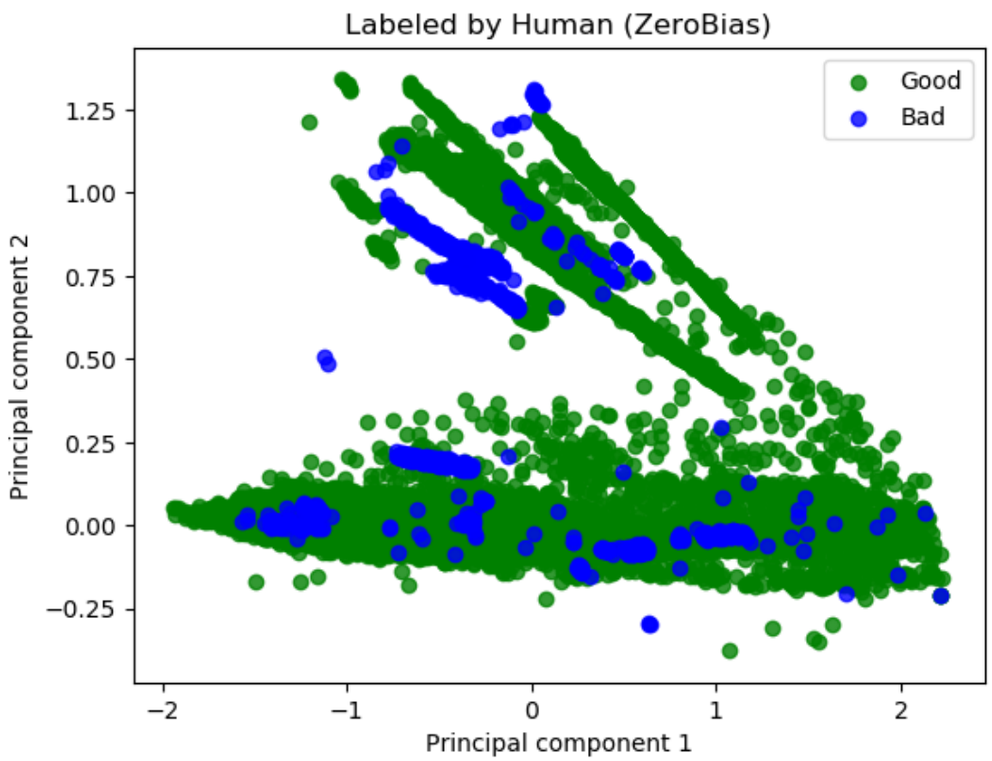
- Preliminary study for 2018
  - Similarity of bad and good LS
  - Results
- Discussion for this work
  - Why our model is too aggressive
  - Fundamental question
  - Solution

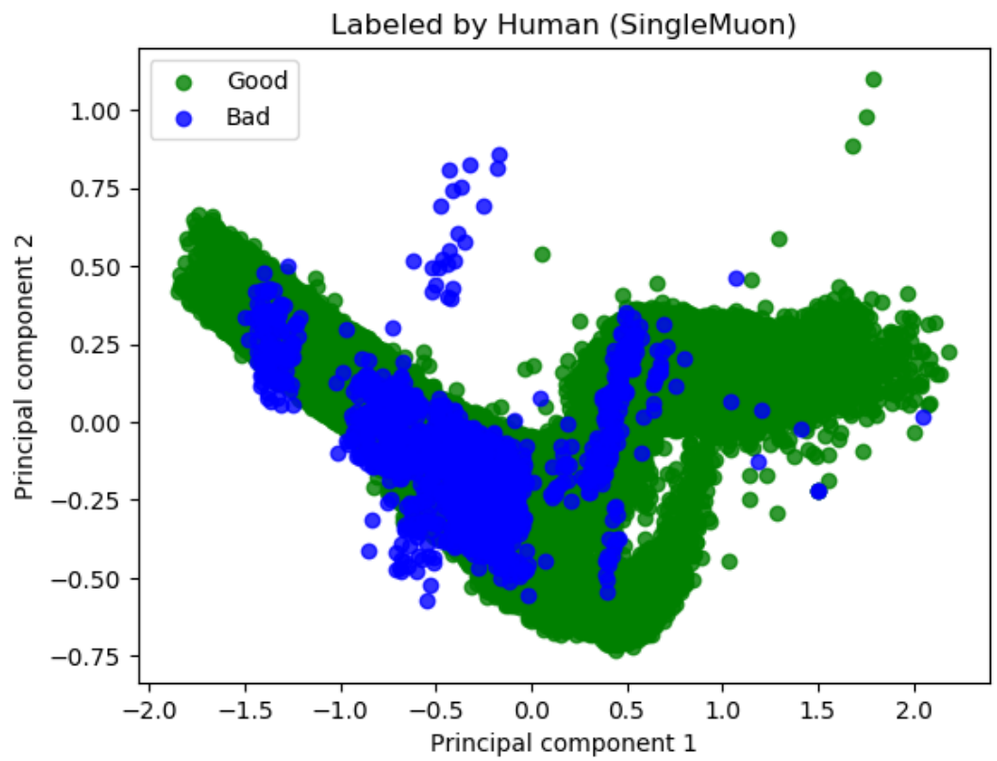
Similarity of bad and good LS in 2018 datasets

Let’s have a look

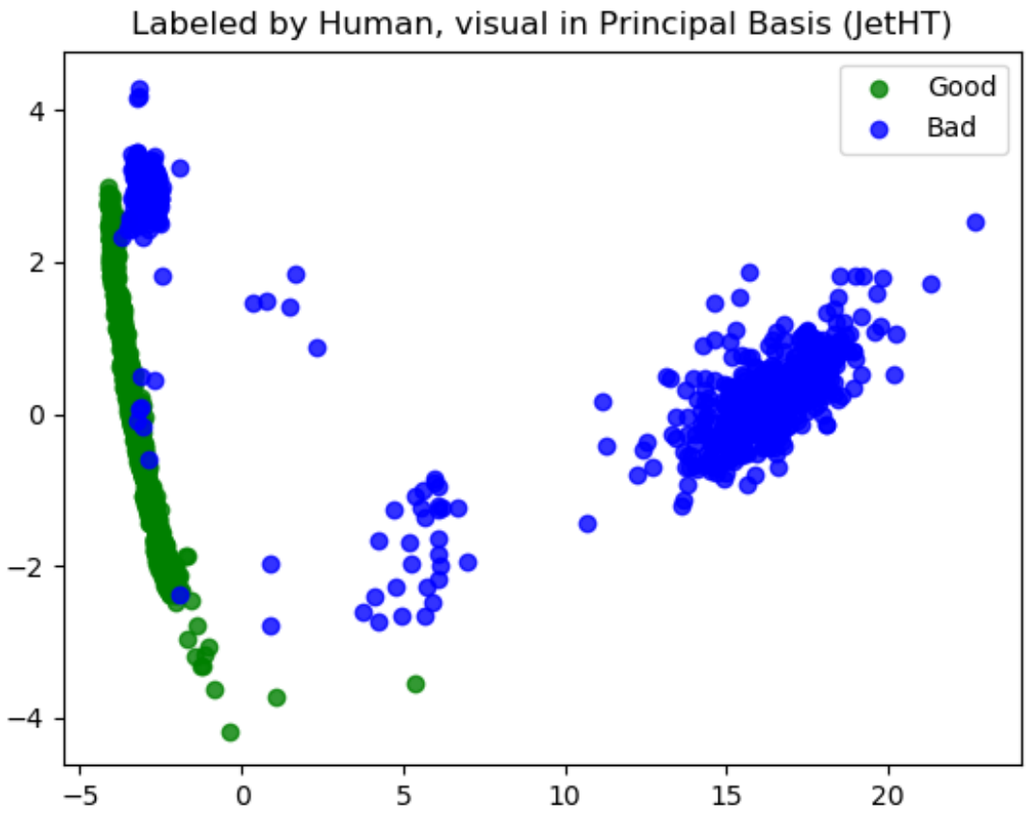




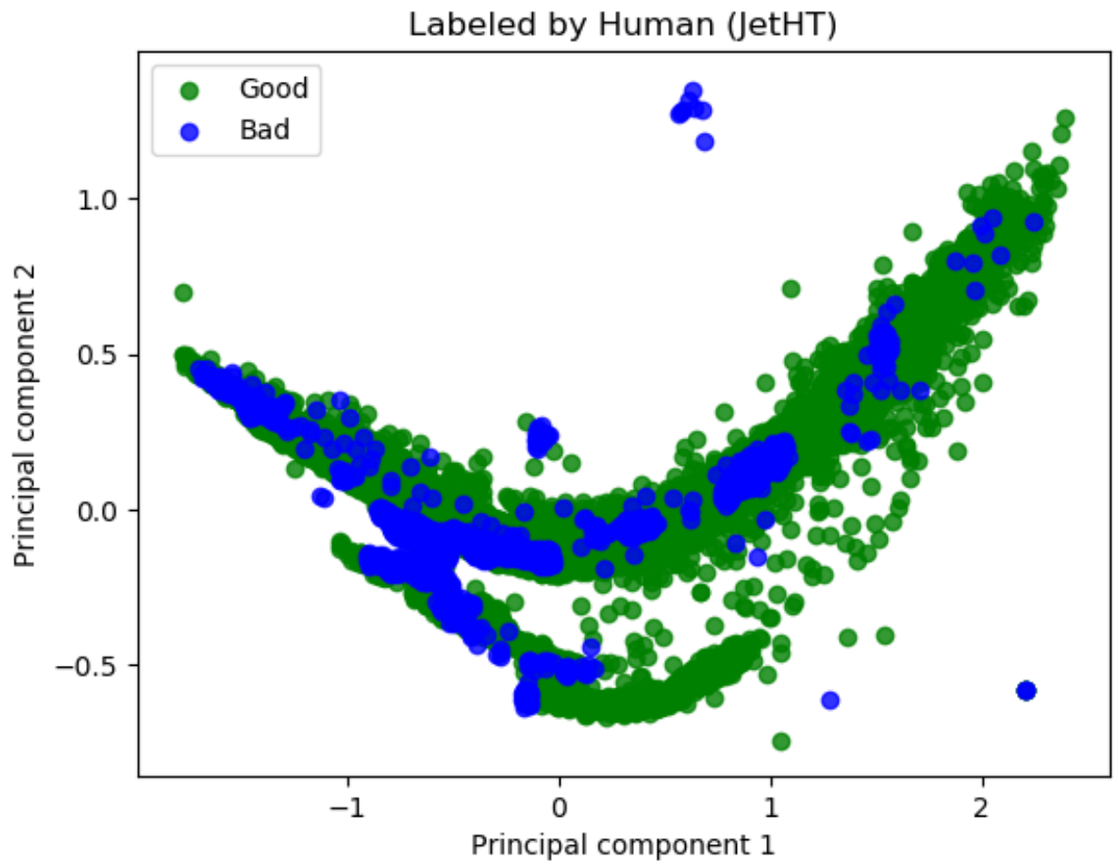




JetHT 2016 VS 2018



2016



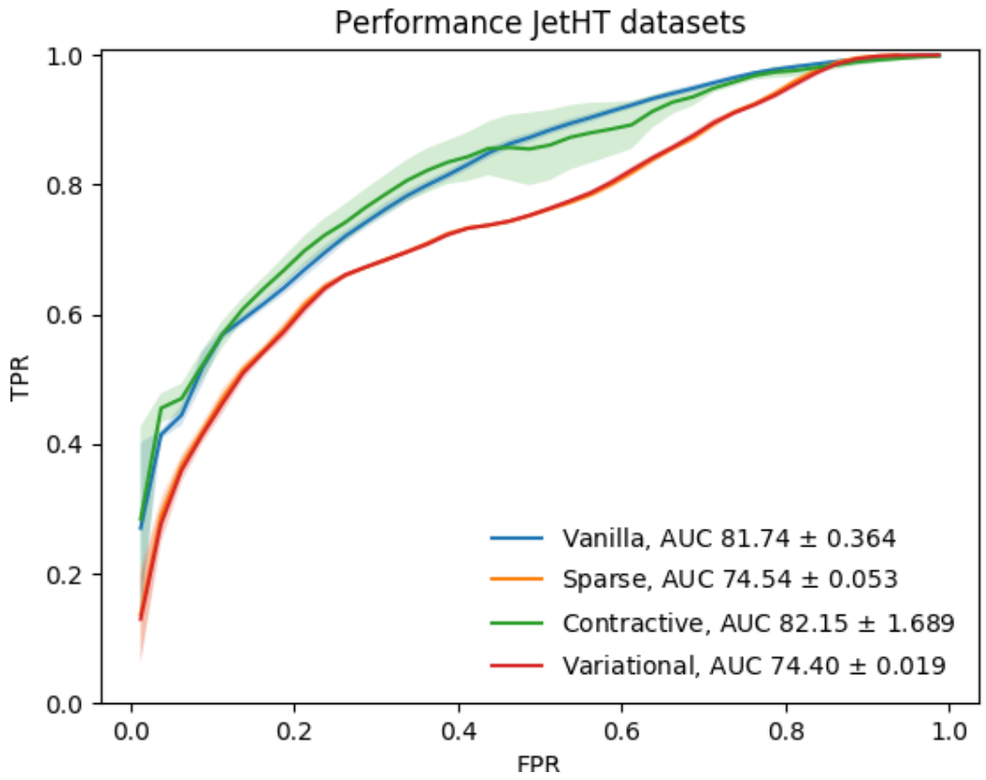
2018

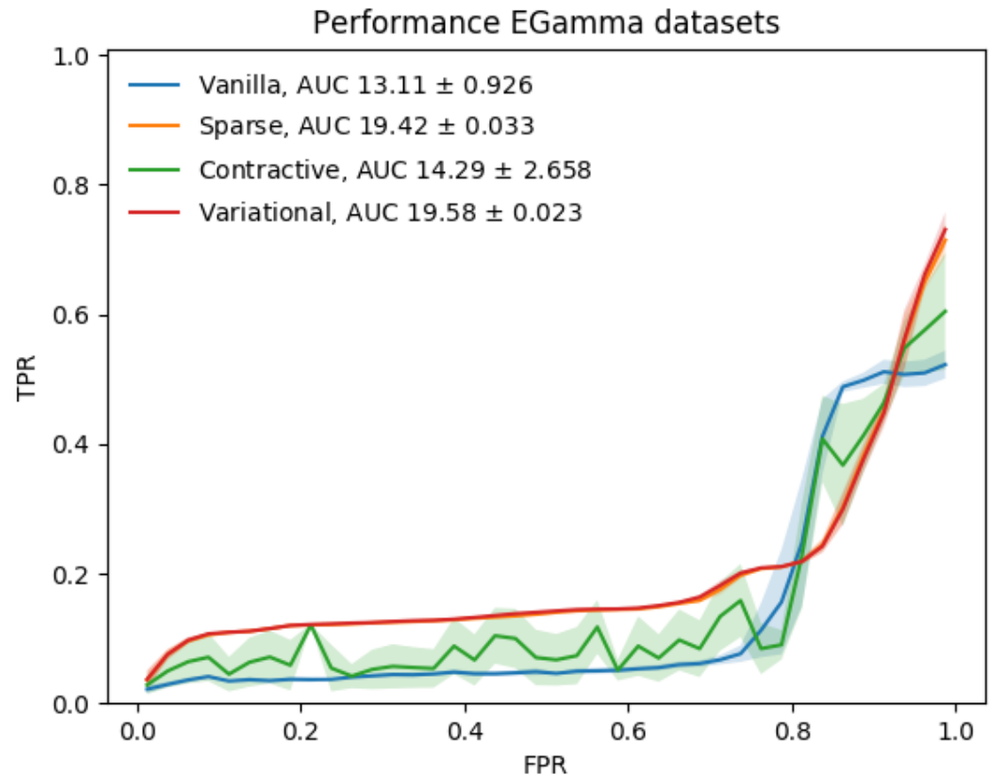
“The problem getting harder when we remove trivial bad LS (Bad Run and DCS bits)”

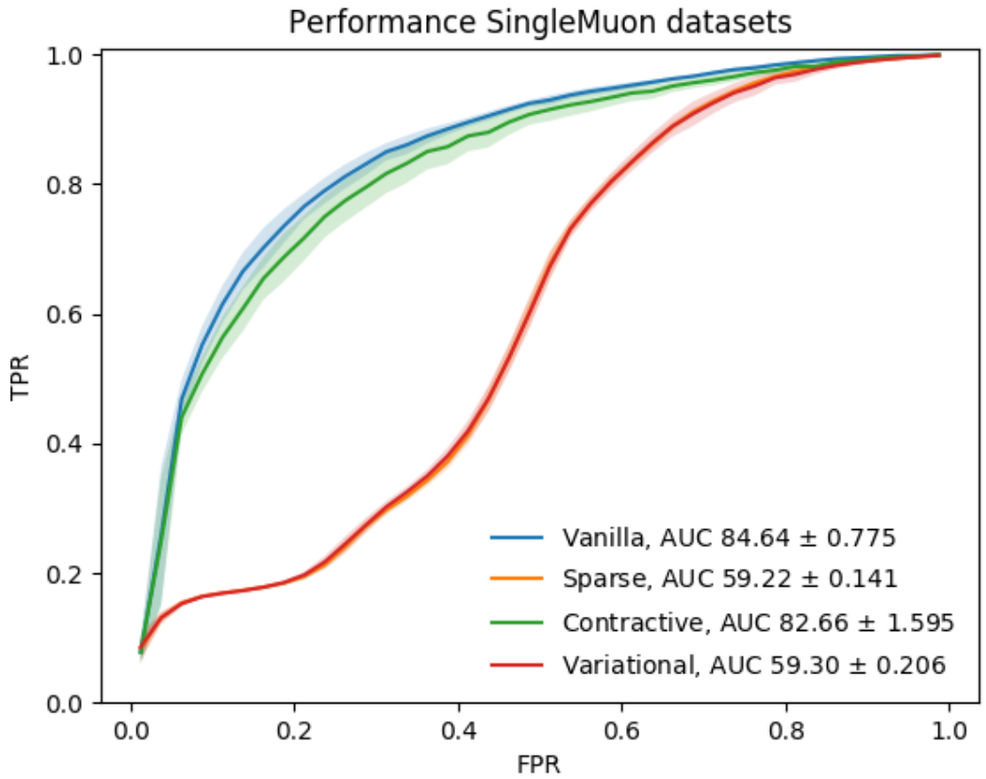
Results

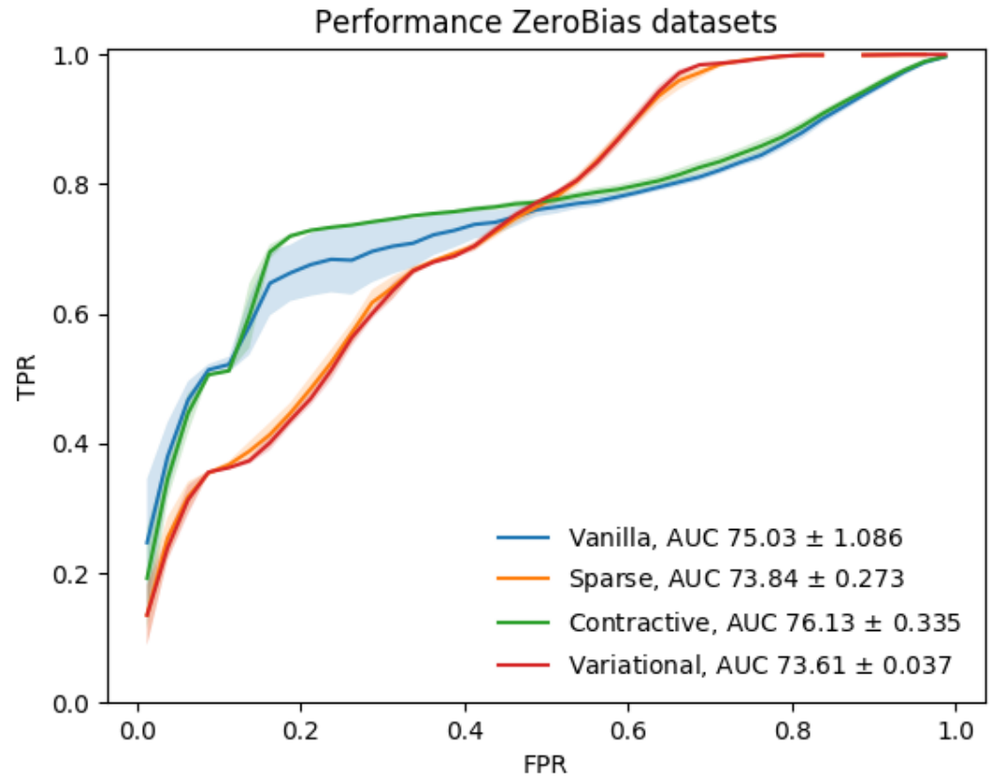
Thanks for IBM GPU Machine :)
160 models has been done in 1 day

Performance

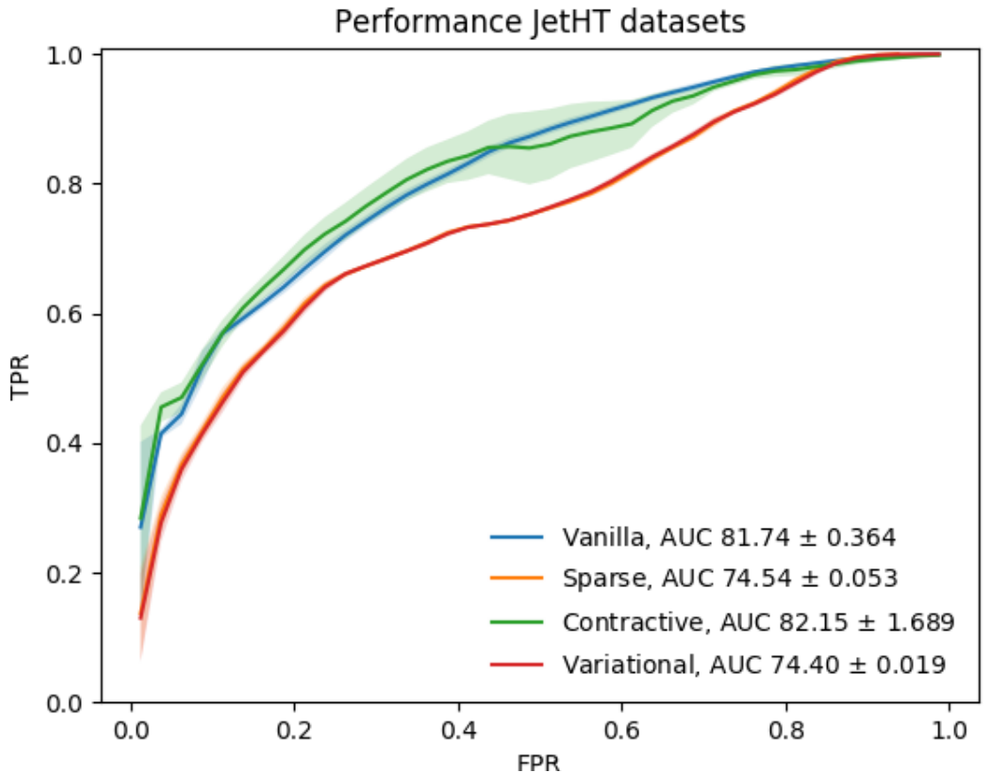


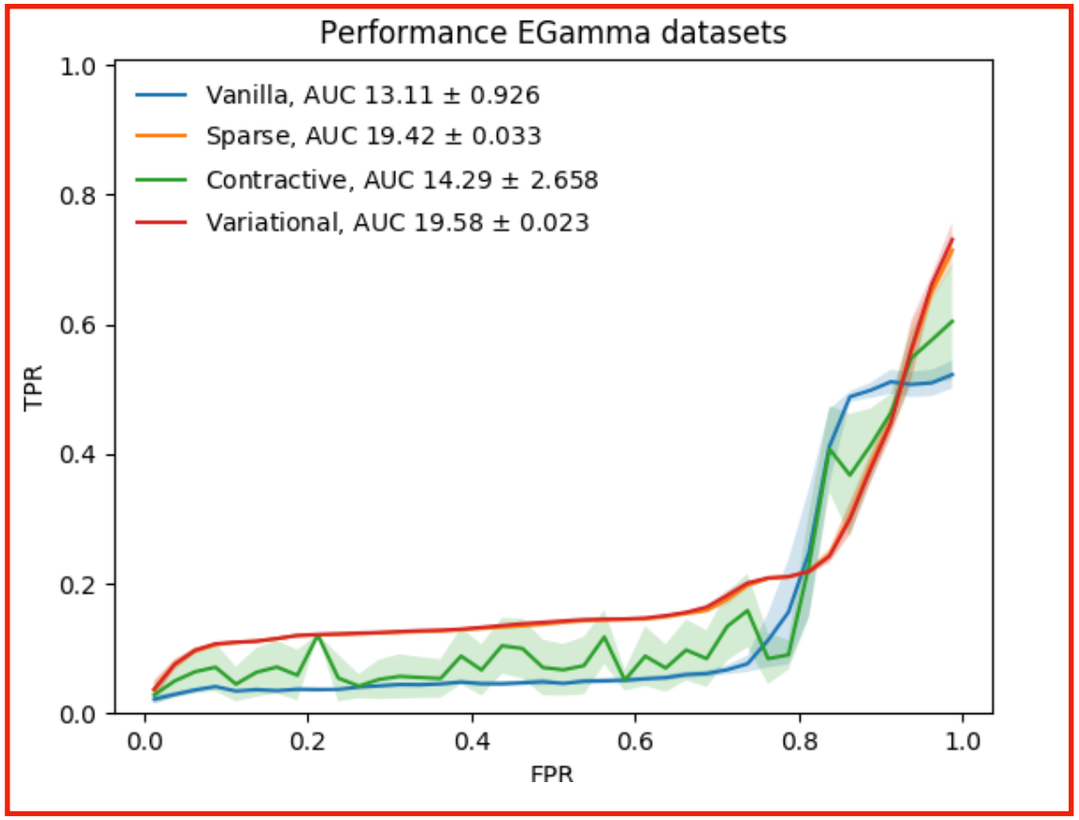


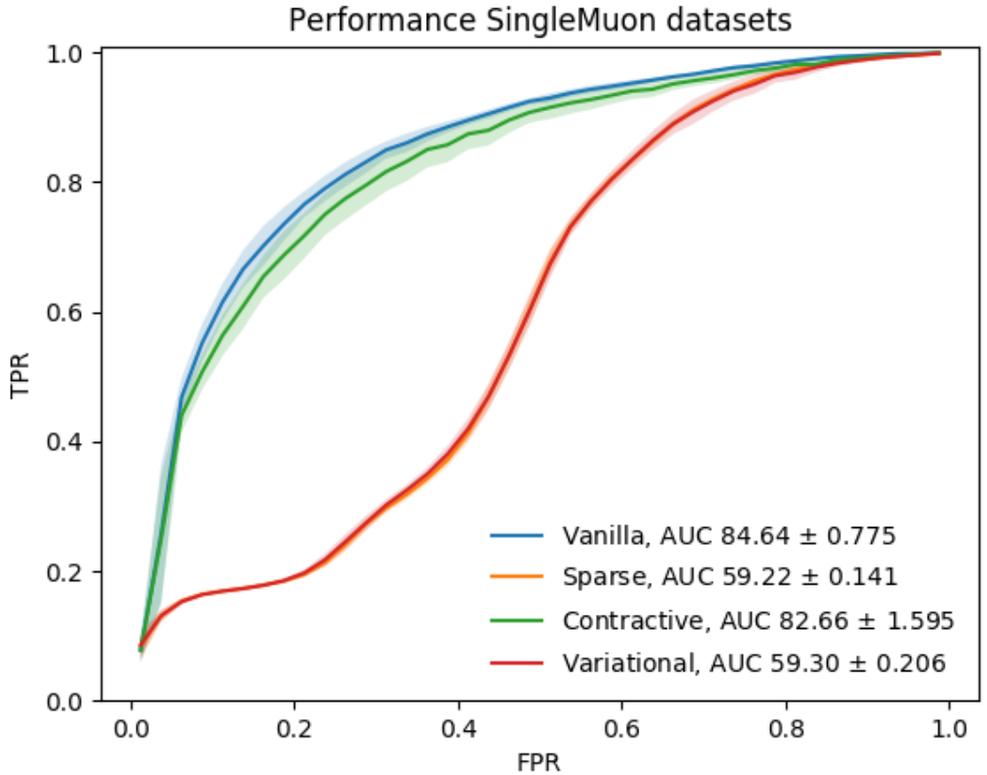


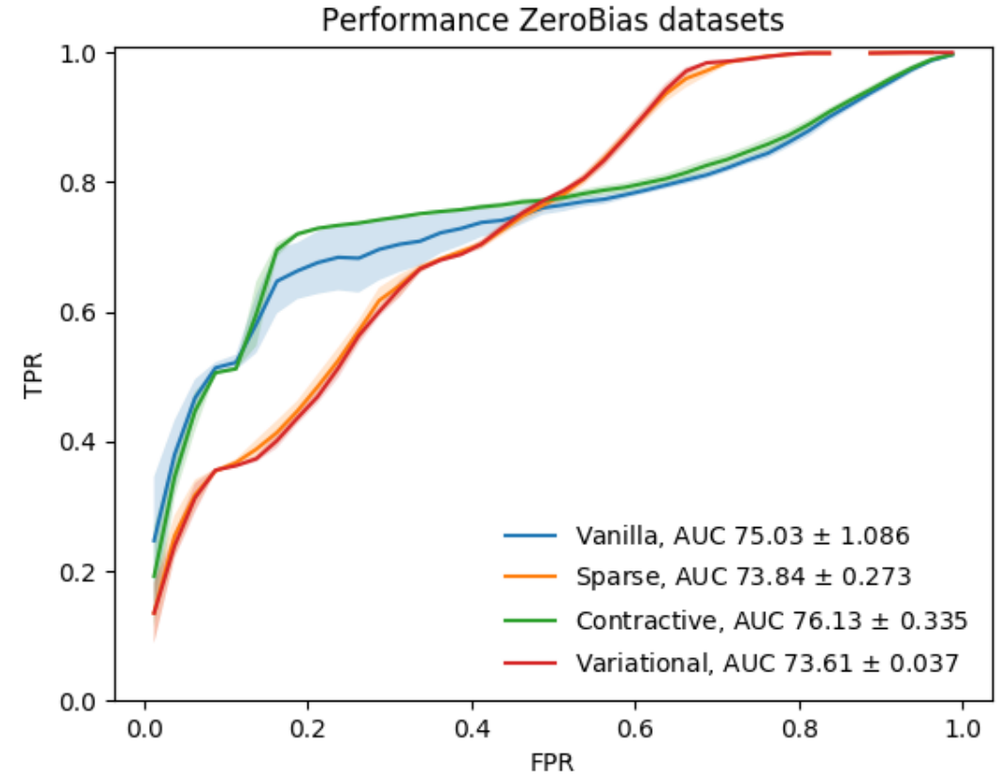


Performance









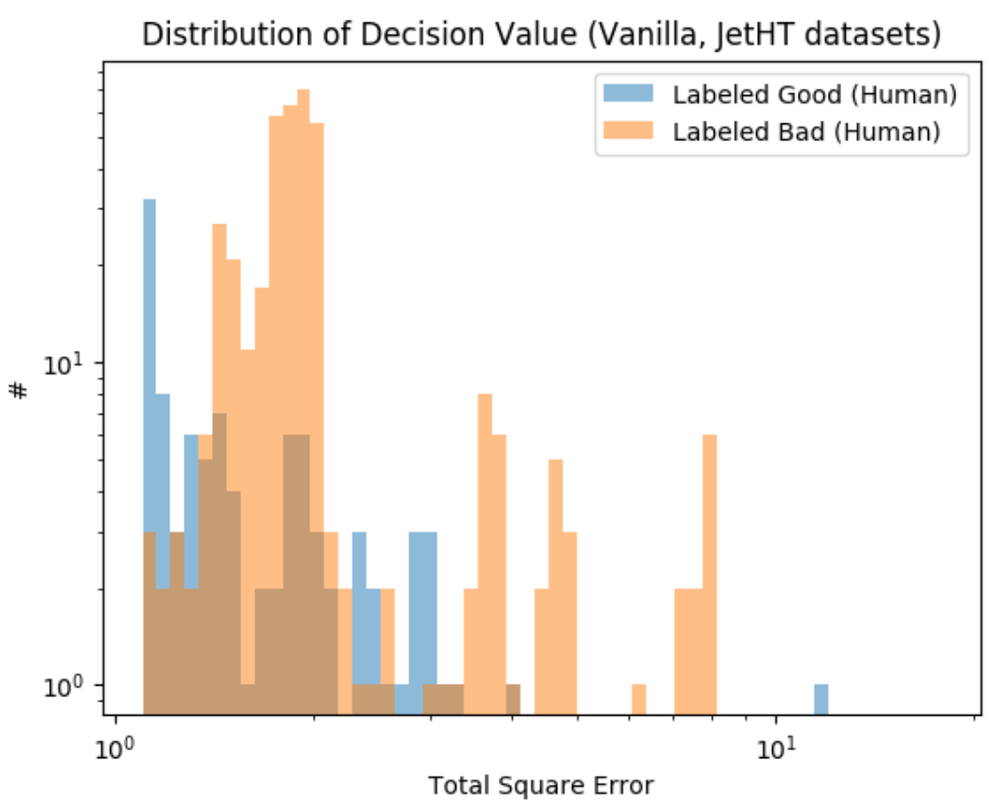
Comparison

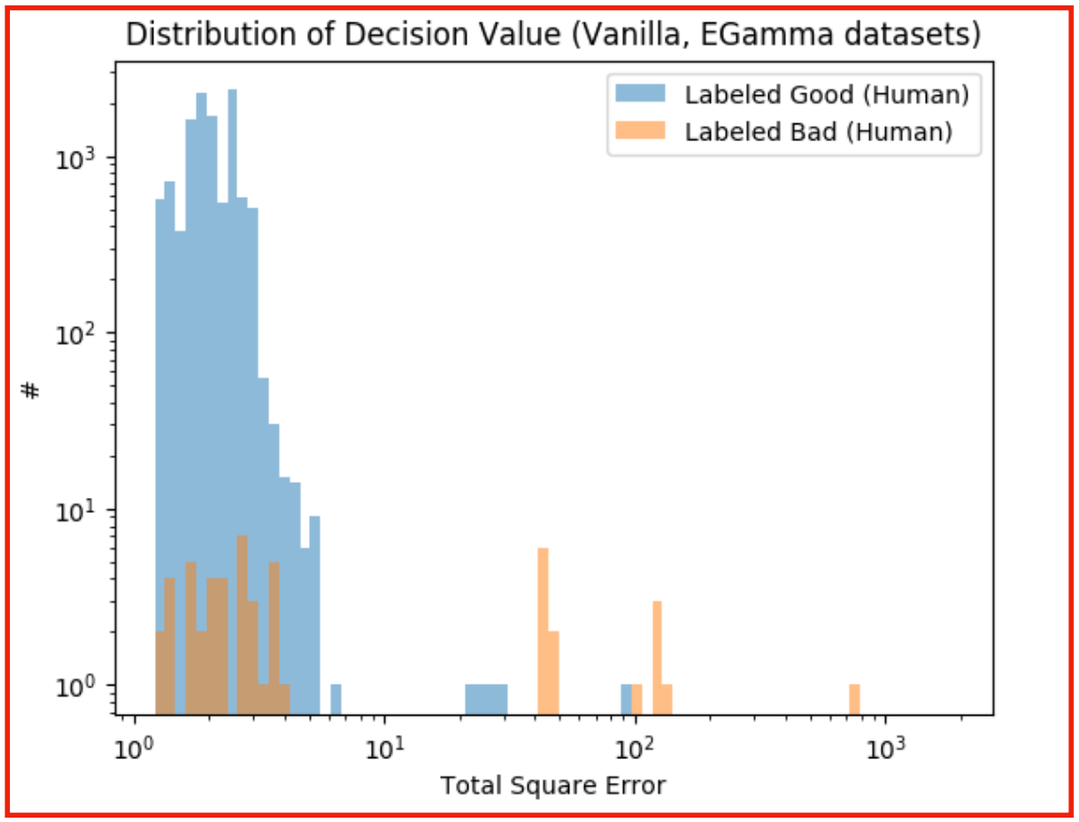
Model\Channel	ZeroBias	JetHT	EGamma	SingleMuon
One-Class SVM	74.2	74.8 (99.1)	16.4	52.9
Contractive AE	74	82 (99.1)	14	83
Vanilla AE	75	81 (99.2)	13	85
Sparse AE	74	74 (98.9)	19	59
Variational AE	74	74 (96.8)	19	59

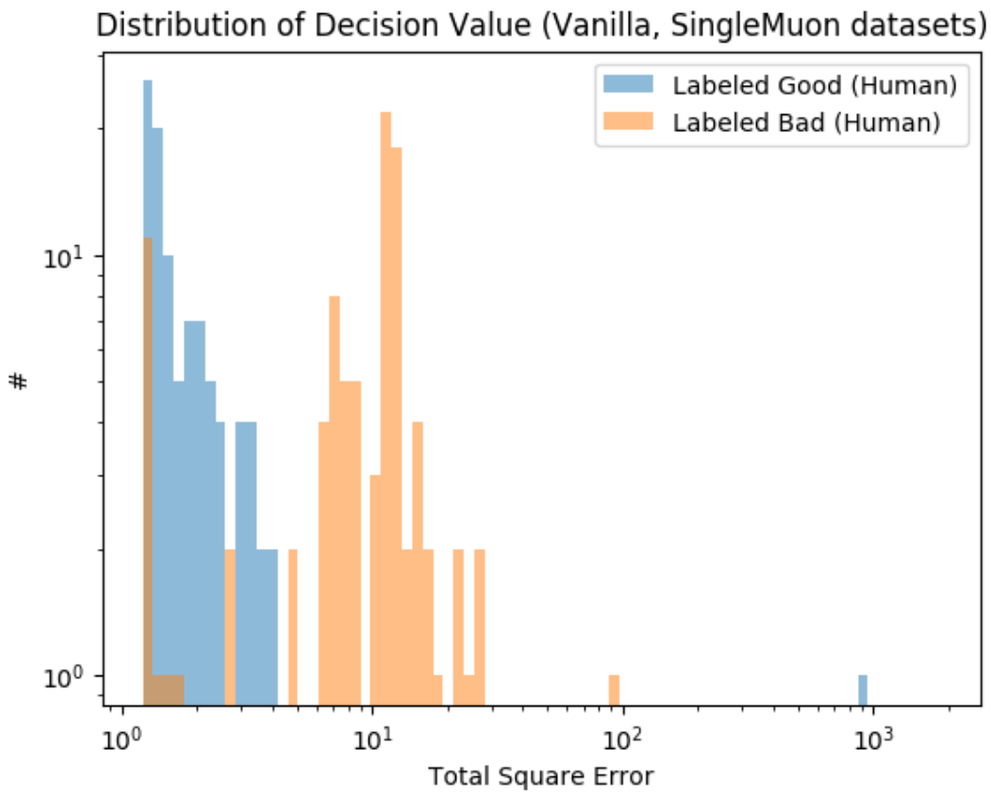
Table of AUC (for convenient you could consider it as accuracy)

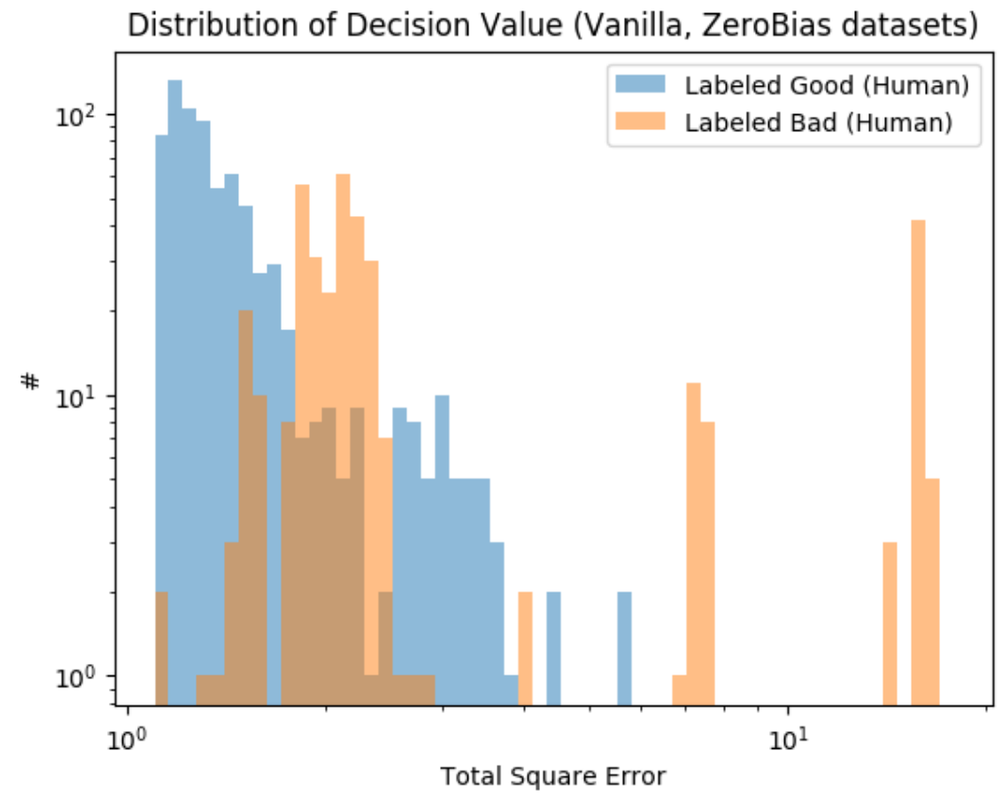
- **2018 datasets** , **2016 datasets**
- Exact same algorithm and hyper parameters

SE Distribution







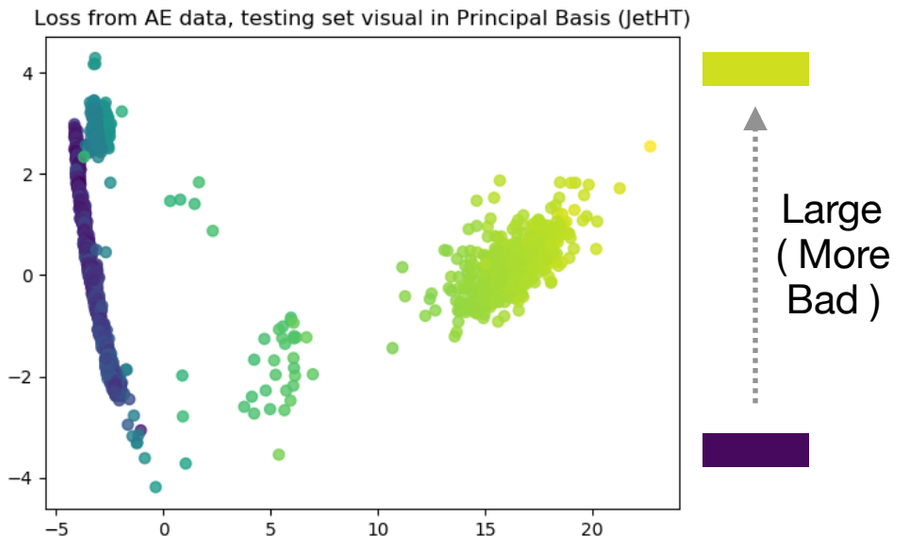


Why it’s aggressive

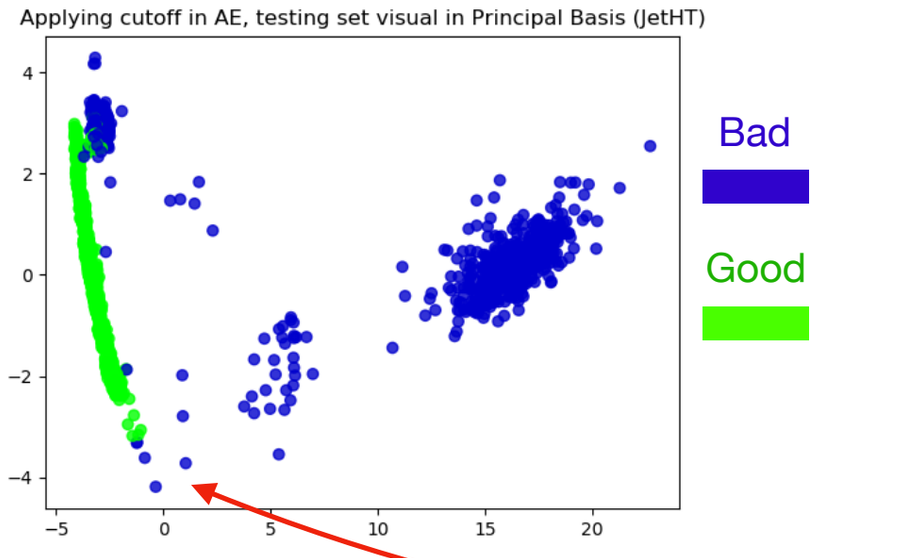
Semi-supervised model is aggressively marking bad LS

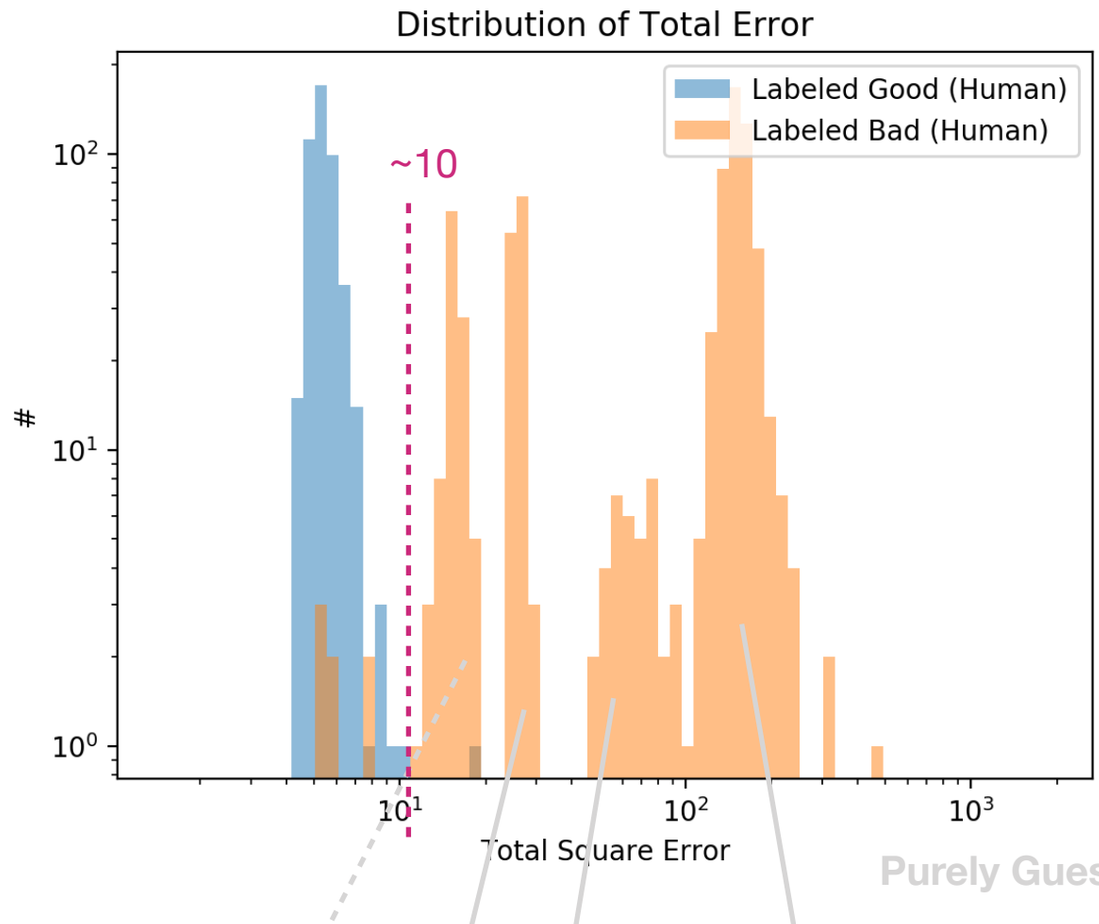
Loss value from Vanilla AE

Loss as color shading

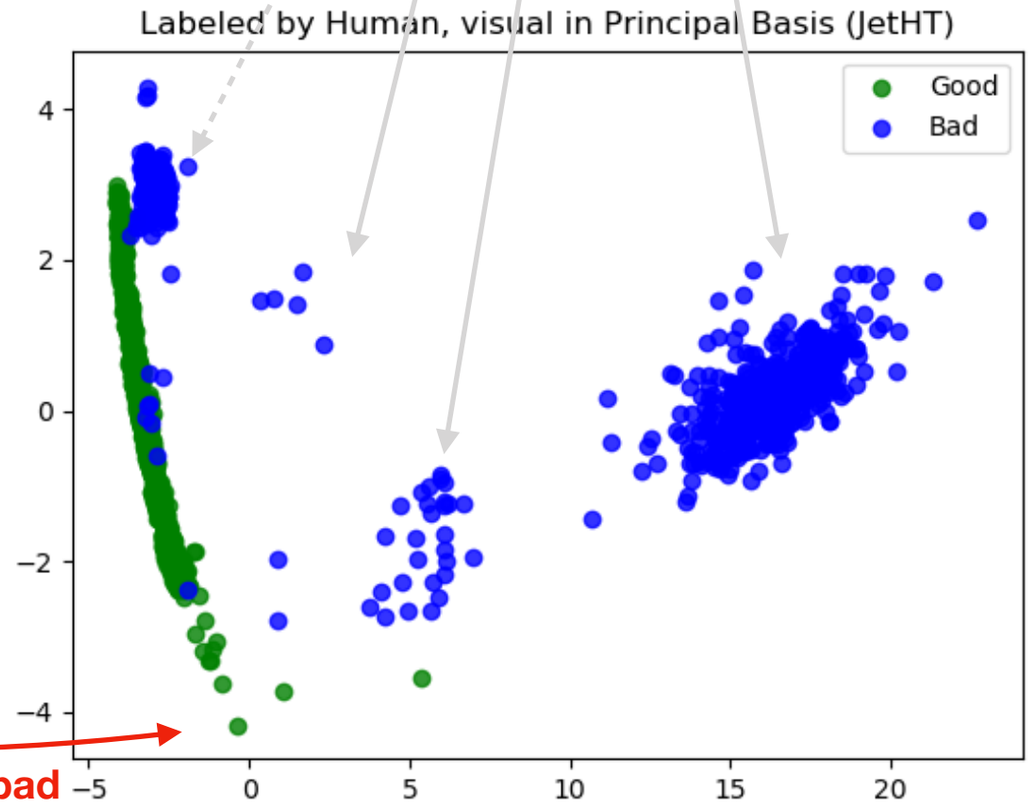


Applying cutoff (MSE > 10.0 is bad)





Purely Guess



Anything that rarely happen, the model would mark as bad

Consequence

- Our ML model determine something that it’s not familiar with by marking as bad LS
- As the time evolving, configuration of CMS also evolve then it’s quite dangerous

Fundamental question

But wait

- If we treat DCS bits as a “ground truth” to train the model:
  - why we need Autoencoder
- else:
  - **Require datasets from simulation** of known bad scenario that DCS bits couldn’t do as well as “normal condition” of detector
  - If we could provide simulated data of bad scenario and normal condition:
    - When it’s disagree with DCS bits when we implemented, how could we trace back to prove correctness

Solution ?

Malfunction Spotter

- Objective
    - Inspect detector malfunction in lumisection granularity which shifter does inspect in run granularity
- Supervised
- Input (x):
    - Physics object
    - Occupancy of sub-detector
- Label (y): status of detector which we could get from RR

Malfunction Spotter

- Model Candidates
    - Decision Tree
    - Random Forest
    - Neural network with **probabilistic output** such as sigmoid to tell “level of confidence” for malfunction in each sub-detector
- If model disagree with DCS bits:
    - call the expert as shifter in RUN granularity does ?
- Future work:
    - Remove physics objects inputs and train only occupancy for implement to reduce online shifter work