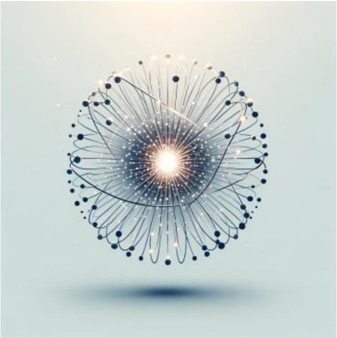
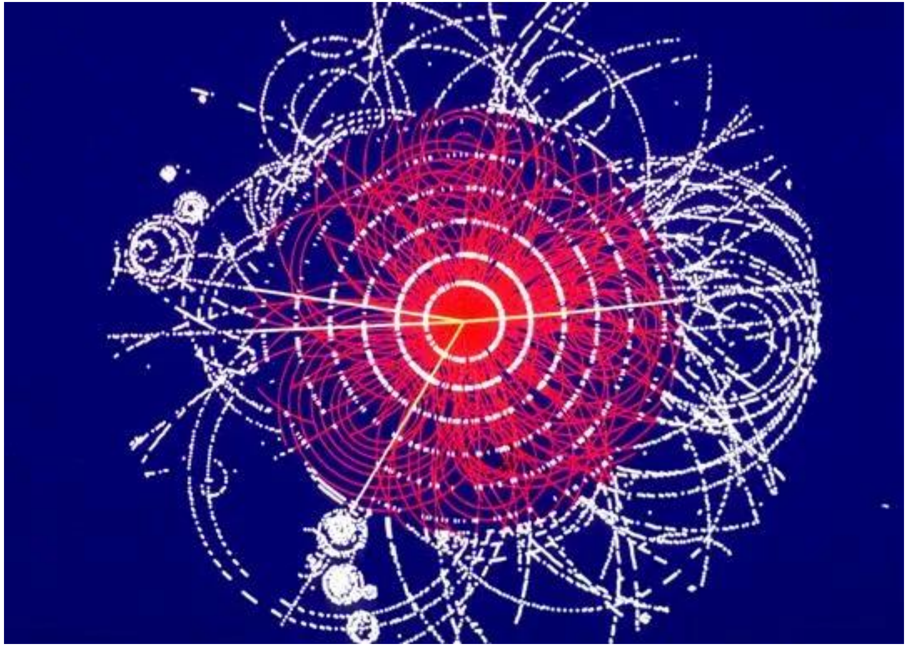


Fair Universe

HiggsML Uncertainty Challenge

Sample Submission

Introduction

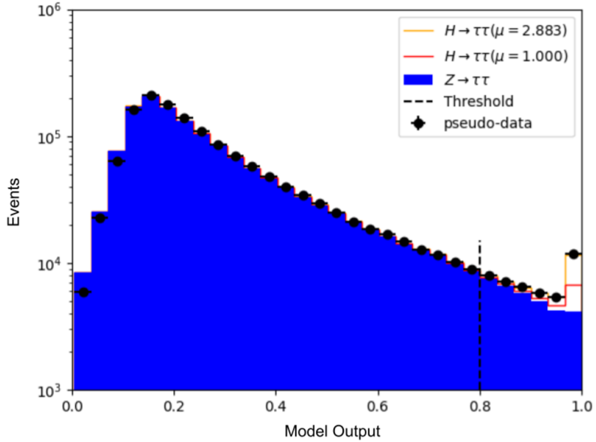


This is a brief summary of the baseline method for the Higgs ML Uncertainty Challenge

The method is a simple ML Classifier with NLL for estimation

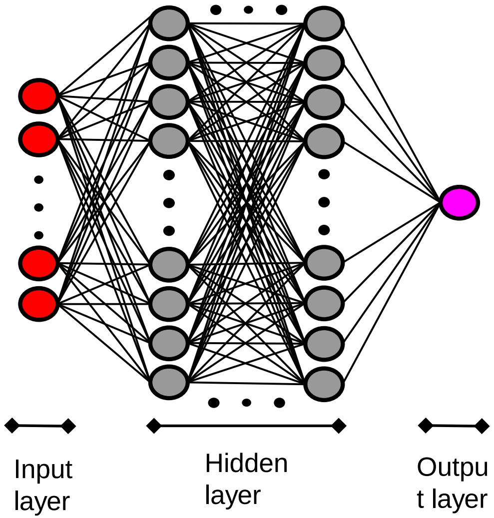
Basic Algorithm

1. Start
2. Divide data into ***train_set*** and ***holdout_set***
3. Use ***train_set*** to Train the ML Classifier
4. Construct for S and B functions from ***holdout_set***
5. Combine Define Negative Log Likelihood function as function of TES and mu
6. For Each pseudo experiment
    a. Predict score for pseudo experiment
    b. Use Minuit to find value of mu, sigma_mu and TES
    c. Returns
        - ***mu***
        - ***p16 = mu - sigma_mu***
        - ***p84 = mu + sigma_mu***
7. End

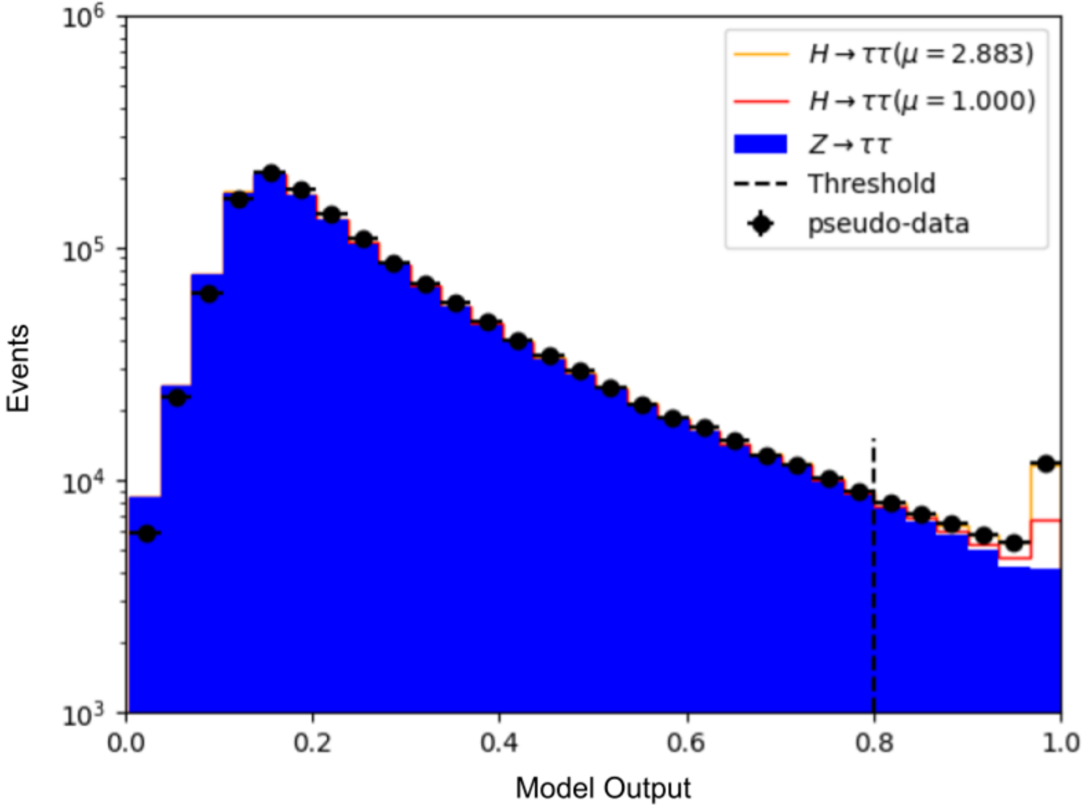


NN with L2 regularization using PyTorch

- PyTorch NN Classifier is Trained to distinguish Signal (Higgs) from Background (Z)
- 32 features,
- Architecture
  - 4 Hidden layers with 200 nodes
  - 1 Output node
  - Sigmoid Activation between layers
  - L2 Regularization during training
- Model return score between 0 (background) and 1(signal),



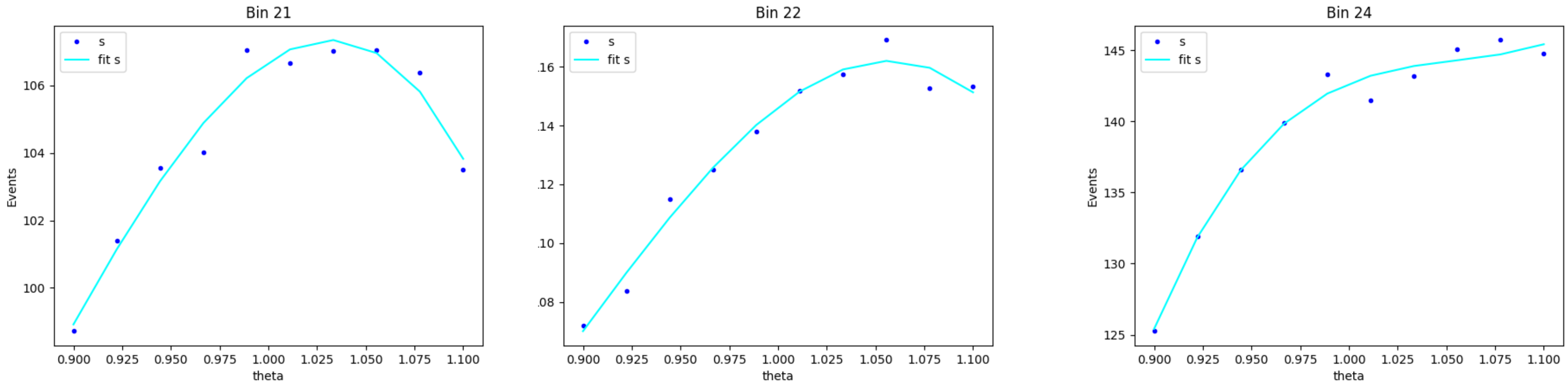
Histograms on Model Score



For each value of TES, The transformed `holdout_set` is evaluated by the model and histogram is build on the model score

Parameterisation of S(α)

With the help of the `holdout_set` for we get values of S and B for each TES in each bin.
A polynomial function is used to fit them. This function is later used in the NLL formalism



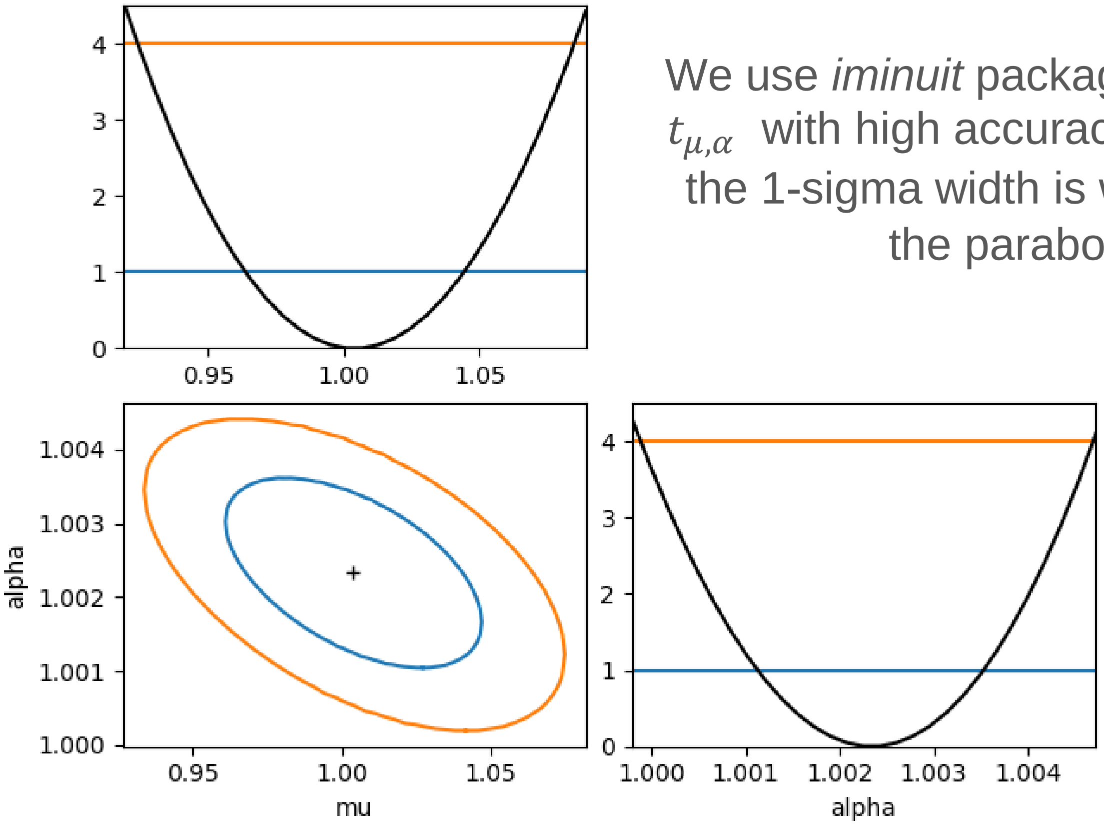
Profile μ and α simultaneously

$$L(\mu, \alpha|\mathcal{D}) = \prod_{i=1}^{N_{\text{bins}}} \frac{(\mu S_i(\alpha) + B_i(\alpha))^{n_i} e^{-(\mu S_i(\alpha)+B_i(\alpha))}}{n_i!}$$

$$\begin{aligned} t_{\mu,\alpha} &= -2\log\left(L(\mu, \alpha \mid \mathcal{D})\right) \\ &= -2\sum_{i}^{N_{\text{bins}}} n_i \log(\mu S_i(\alpha) + B_i(\alpha)) + (\mu S_i(\alpha) + B_i(\alpha)) \end{aligned}$$

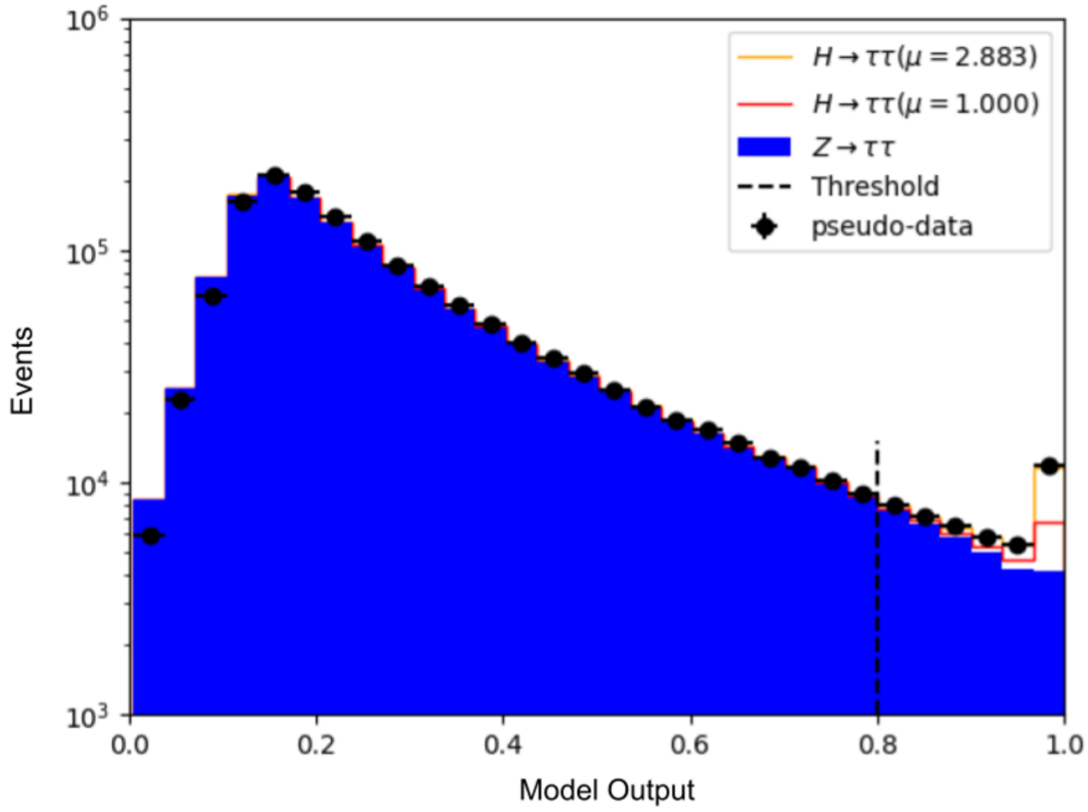
L here is the likelihood estimator which depends on μ and α, thus the μ at which L is maximum or $t_{\mu,\alpha}$ is minimum is the predicted $\hat{\mu}$,

NLL ($t_{\mu,\alpha}$) curve and contour

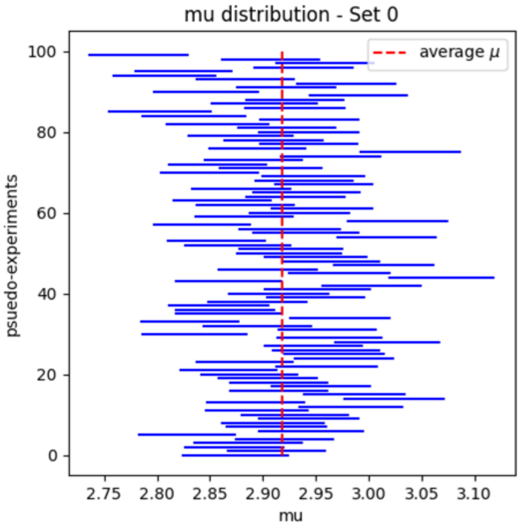


We use *iminuit* package to find the minimum of $t_{\mu,\alpha}$ with high accuracy and the 1-sigma width, the 1-sigma width is width between points on the parabola for $t_{\mu,\alpha} = 1$

Fit on one pseudo Experiments



Signal Strength and Coverage - Pytorch



====================
Overall Score
====================
[*] --- RMSE: 0.054
[*] --- MAE: 0.098
[*] --- Interval: 0.084
[*] --- Coverage: 0.46
[*] --- Quantiles score: -1.651

Coverage plot for NN pytorch (syst) [100 PX]

Other Remarks

- The model is designed to train for 100 epochs with a early stop
- It’s recommended to be trained outside codabench.
- The model has a method to detect trained model, to avoid unnecessary re-training.
- Please comment it out or delete the trained model from the sample_code_submission, if you want to retrain.
- The starting kit has option to run on *sample_data* or `public_data` The `sample_data` is ~ 1% of the *public_data* so be cautious about it while training.
- All models should be serializable to be compatible with ingestion.

Back-up

Parameterisation of B(alpha)