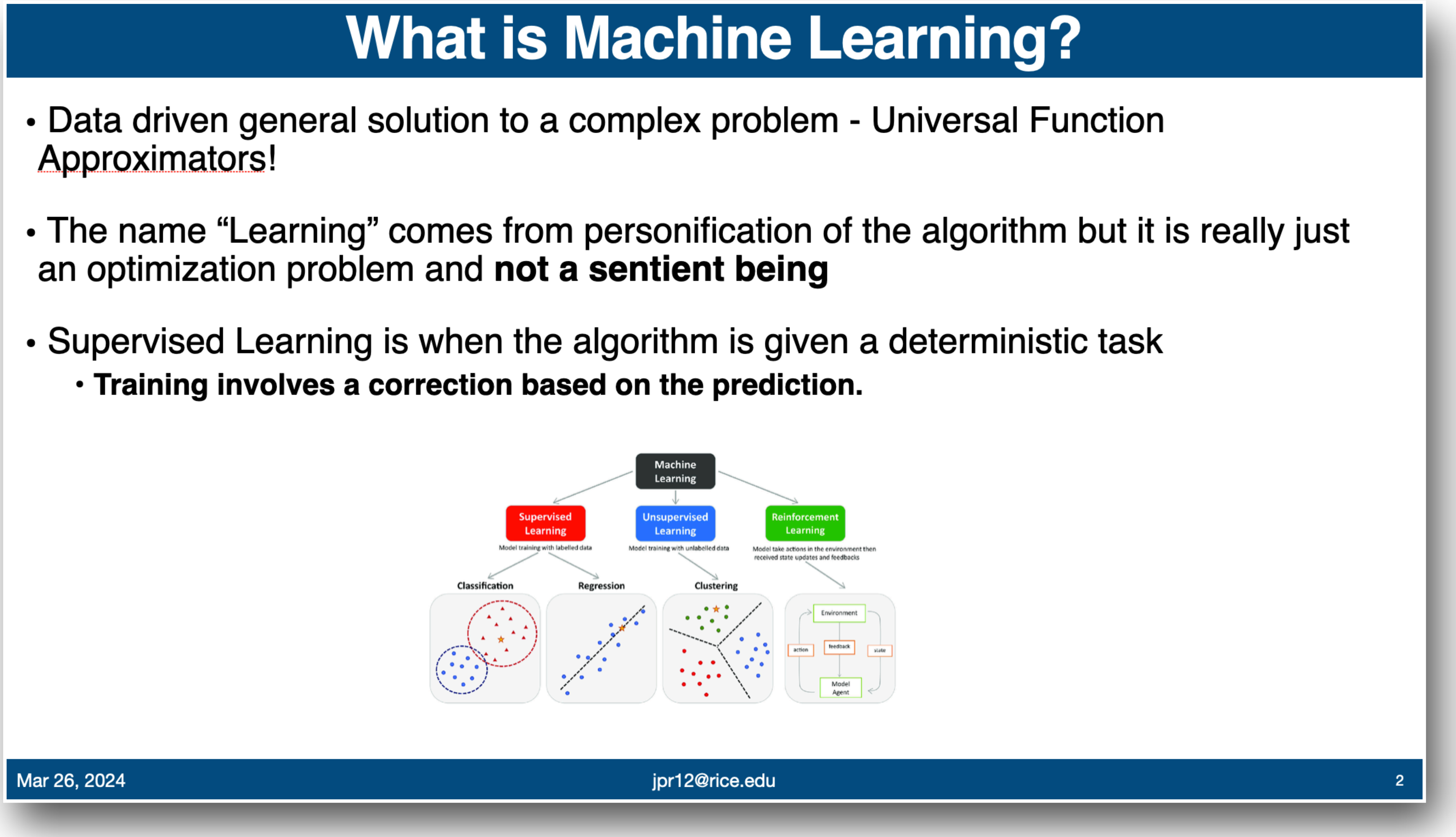
Introduction to Regressor BDT

Offshell Workshop Tutorial

John Rotter (Rice)
Graduate Student

RICE

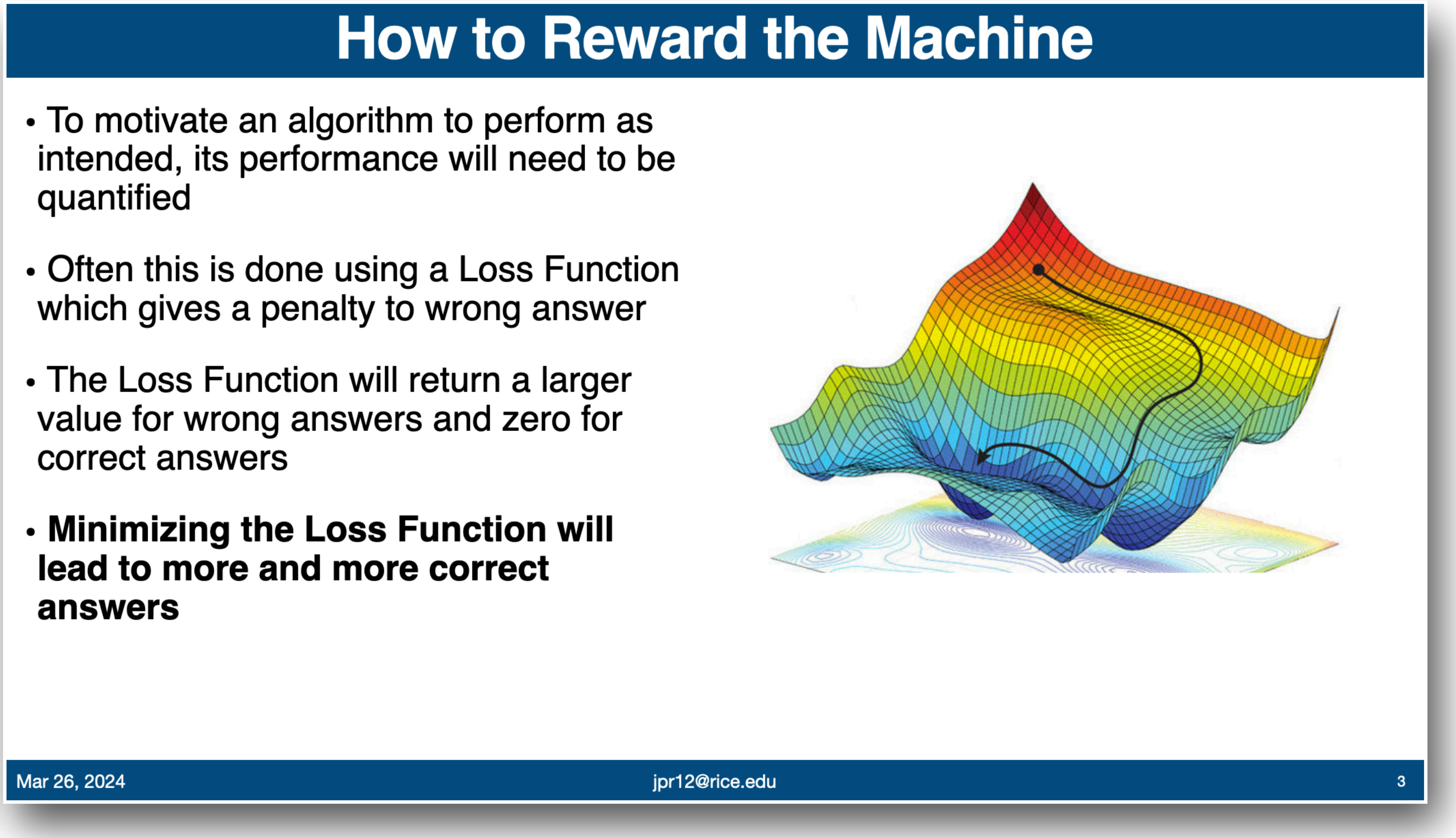
Previous Tutorials



What is Machine Learning?

- Data driven general solution to a complex problem - Universal Function Approximators!
- The name “Learning” comes from personification of the algorithm but it is really just an optimization problem and **not a sentient being**
- Supervised Learning is when the algorithm is given a deterministic task
  - **Training involves a correction based on the prediction.**

Mar 26, 2024 jpr12@rice.edu 2



How to Reward the Machine

- To motivate an algorithm to perform as intended, its performance will need to be quantified
- Often this is done using a Loss Function which gives a penalty to wrong answer
- The Loss Function will return a larger value for wrong answers and zero for correct answers
- **Minimizing the Loss Function will lead to more and more correct answers**

Mar 26, 2024 jpr12@rice.edu 3

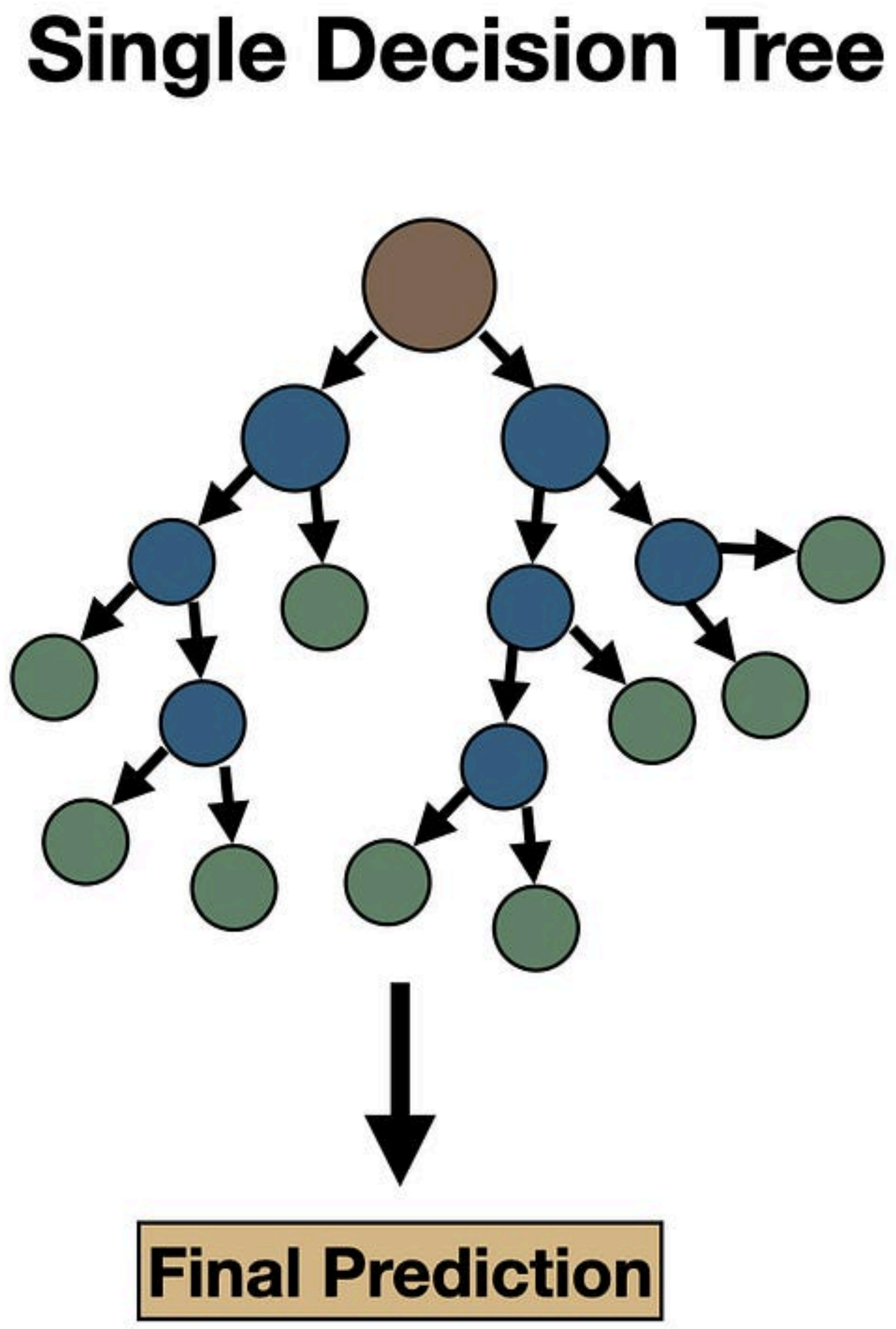
https://indico.cern.ch/event/1375252/timetable/#16-machine-learning

Weak Learners

- Weak learners are small networks or trees that perform just barely better than guessing

- The idea is that it is still learning but is not learning the full model

- Cannot be trained to arbitrary level of accuracy

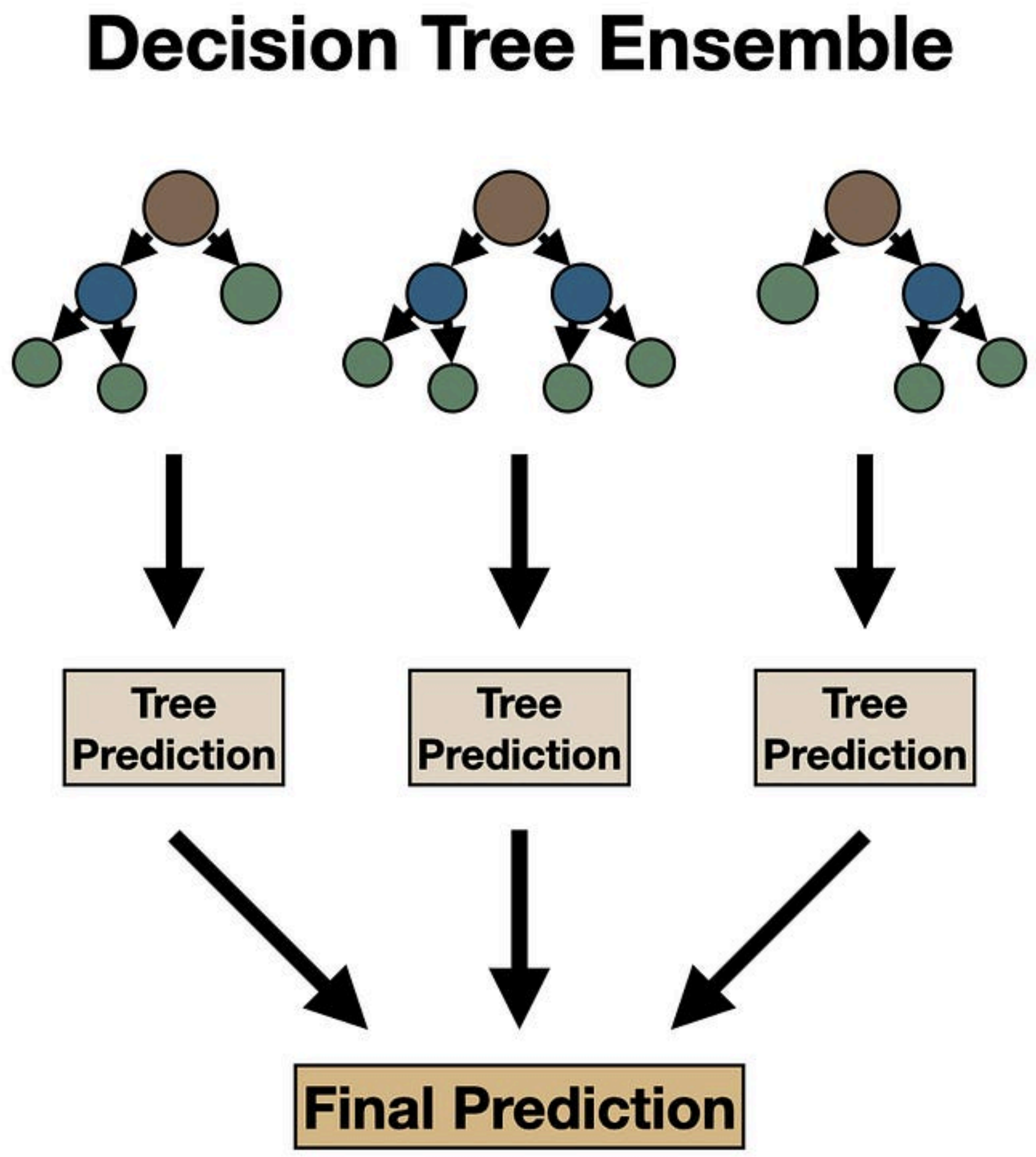


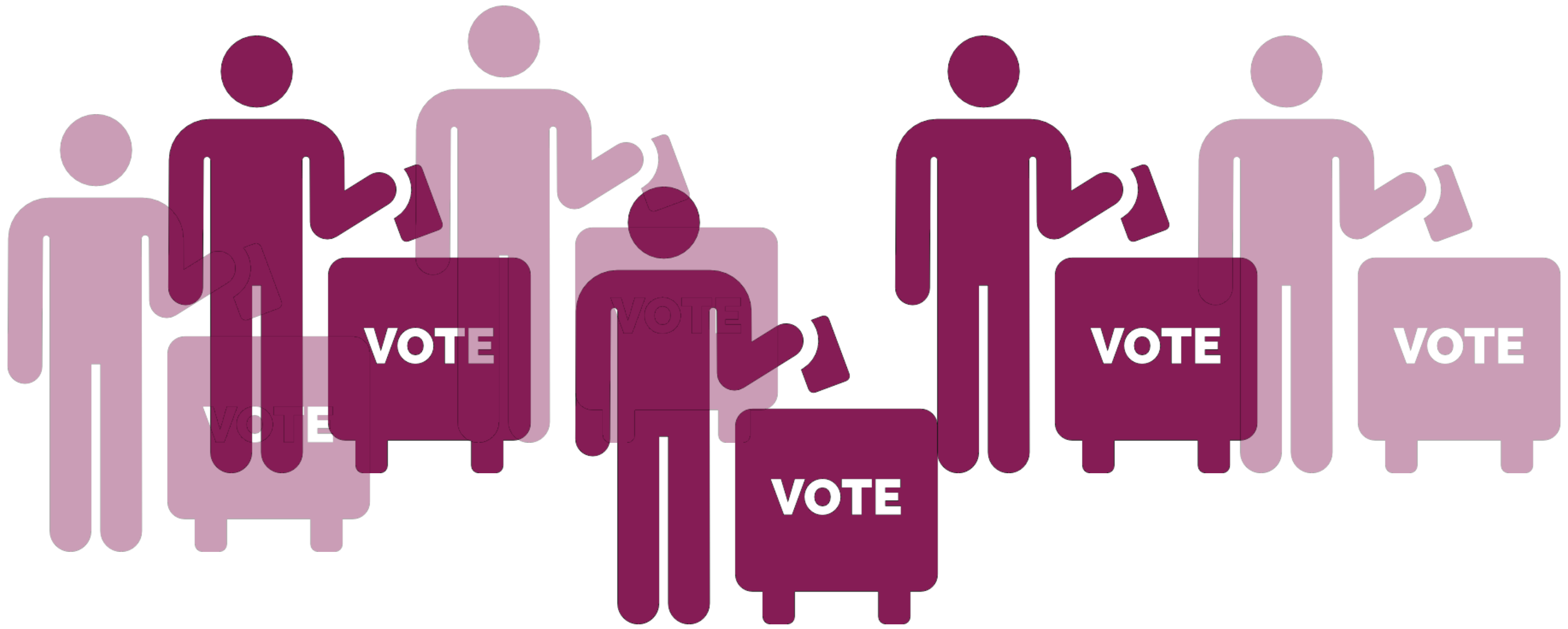
Ensemble of Weak Learners

- Many weak learners together can form a **Strong Learner** when chained together

- The idea is that each weak learner can learn about a subset of the structure of the algorithm

- The strong learner can recursively train more weak learners based on the error of the current ensemble!



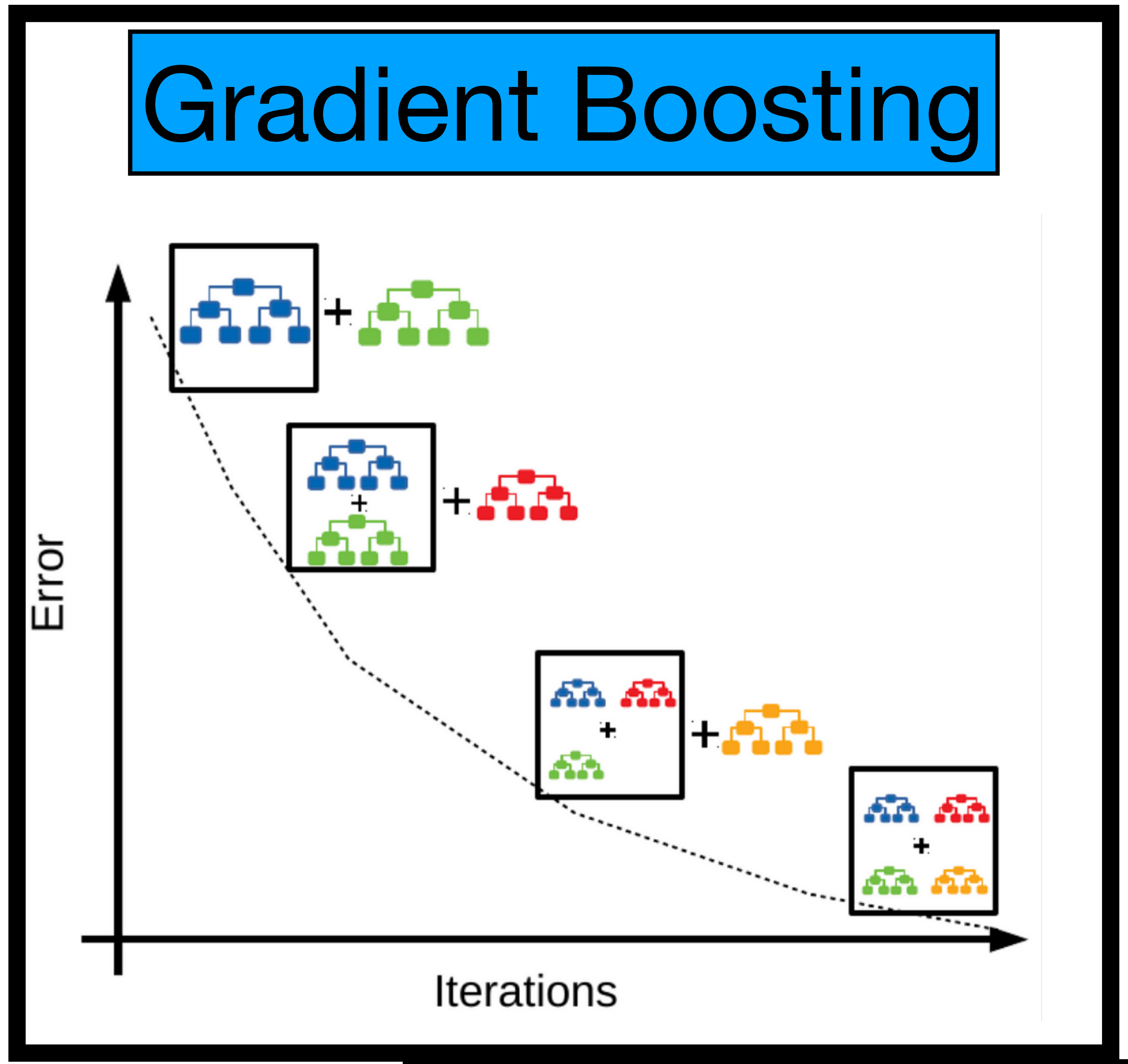


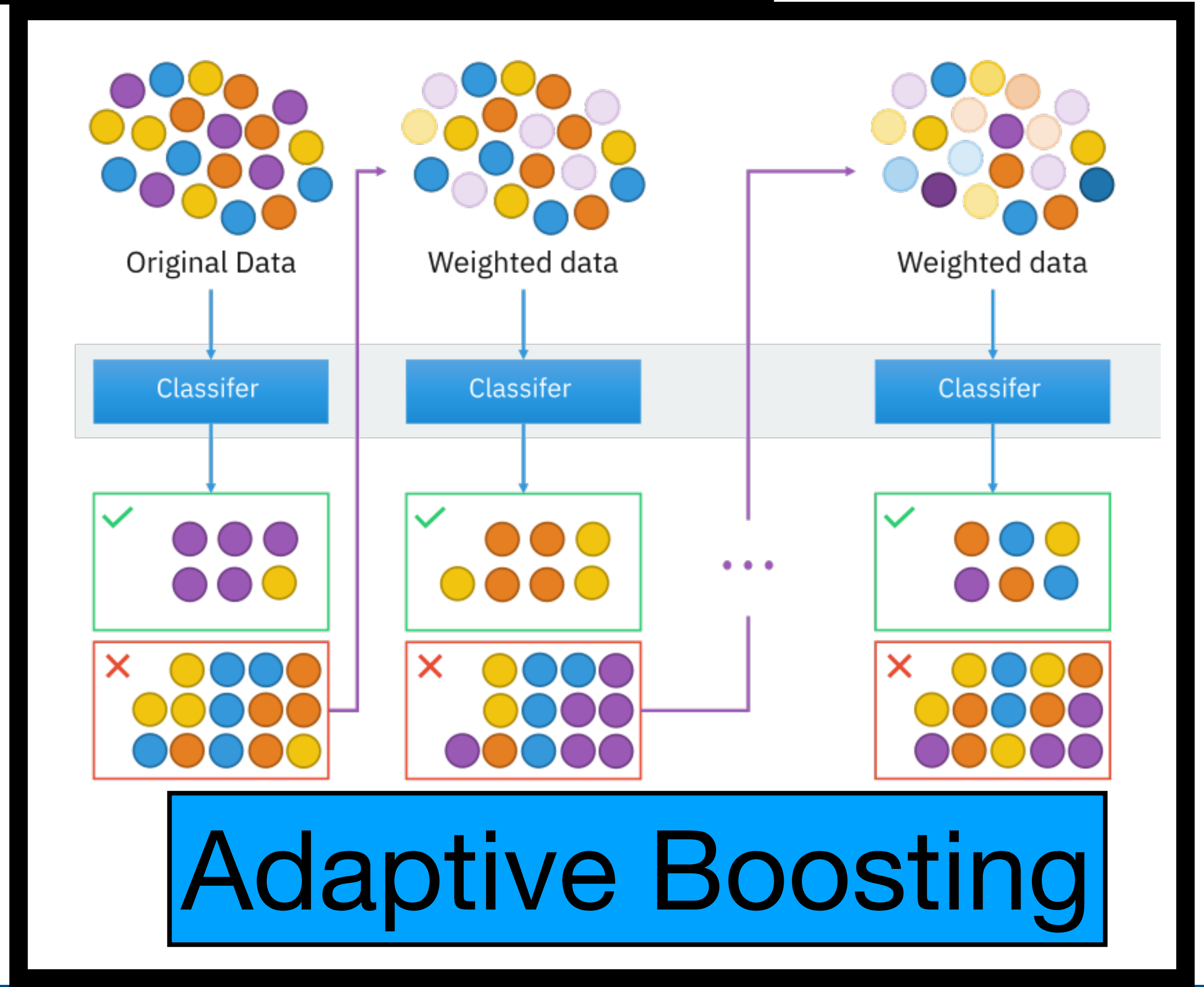
Boosted Decision Trees (BDTs)

- BDTs are a supervised learning technique which **utilizes weak learners in an ensemble to make some prediction**

- They can be used for regression or categorization

- Boosting refers to how the trees "grow" in the forest
  - Gradient Boosting (GBoost)
  - Adaptive Boosting (AdaBoost)
  - Extreme Gradient Boosting (XGBoost) - Parallelized Gradient Boosting





BDT Leaves and Predictions

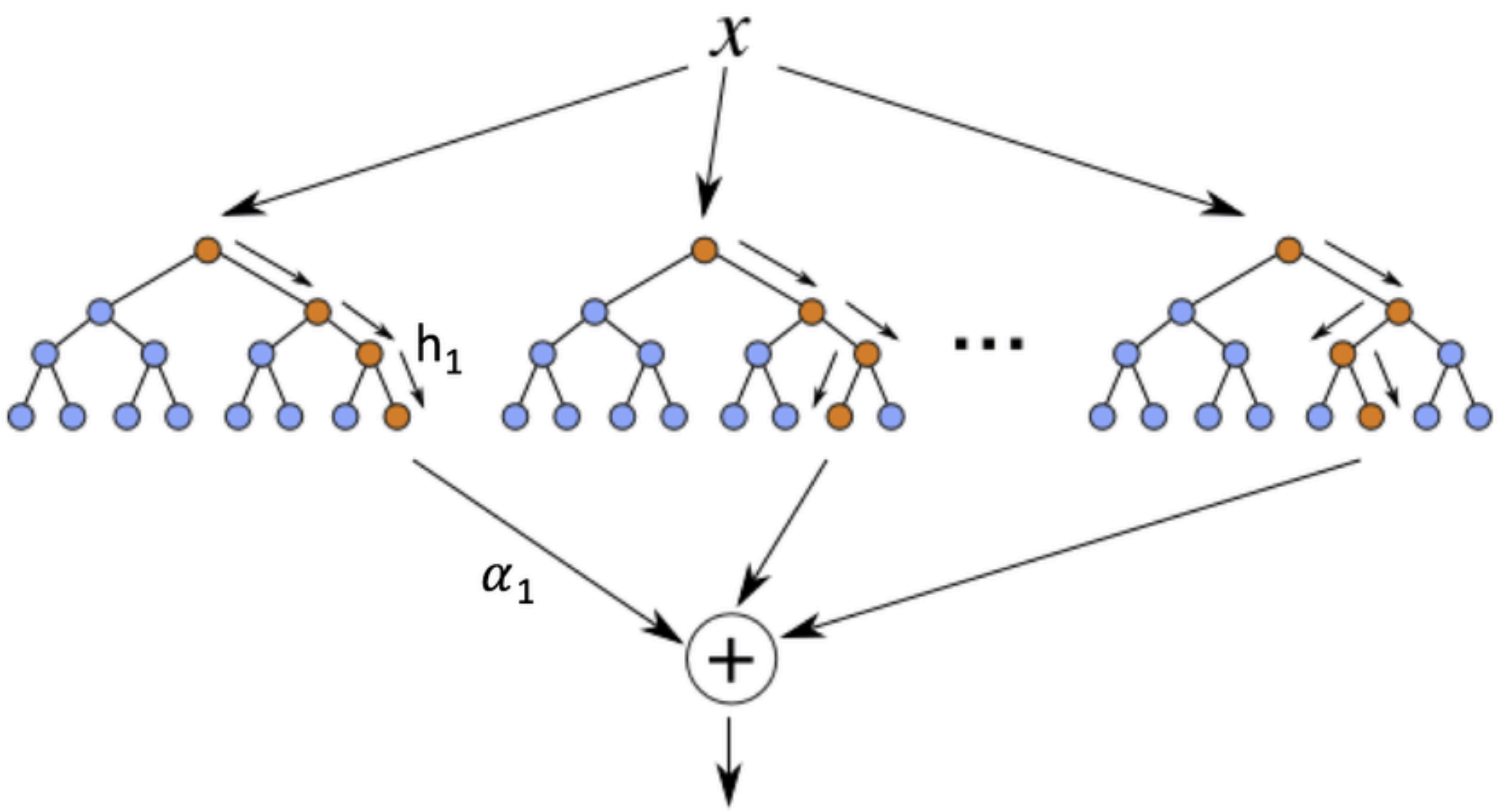
- Based on the BDT structure chosen, the prediction method may vary

- For BDTs in XGBoost, trees can be traversed and will terminate at a “leaf”

- These “leaves” will all contribute some small weight, α_i

- Summing these weights yields the prediction

- Sometimes this summation can also have a function when used in classification such as a sigmoid or softmax



Training a Regressor BDT using XGBoost

- When training any algorithm, a testing, training, and validation are randomly split to identify performance of the training

- Training data includes events whose output is already known

- The input data is fed into the BDT for each event and trained on a small batch of events

- Each time the BDT is evaluated, the residual is calculated and a new tree is appended to minimize the residual

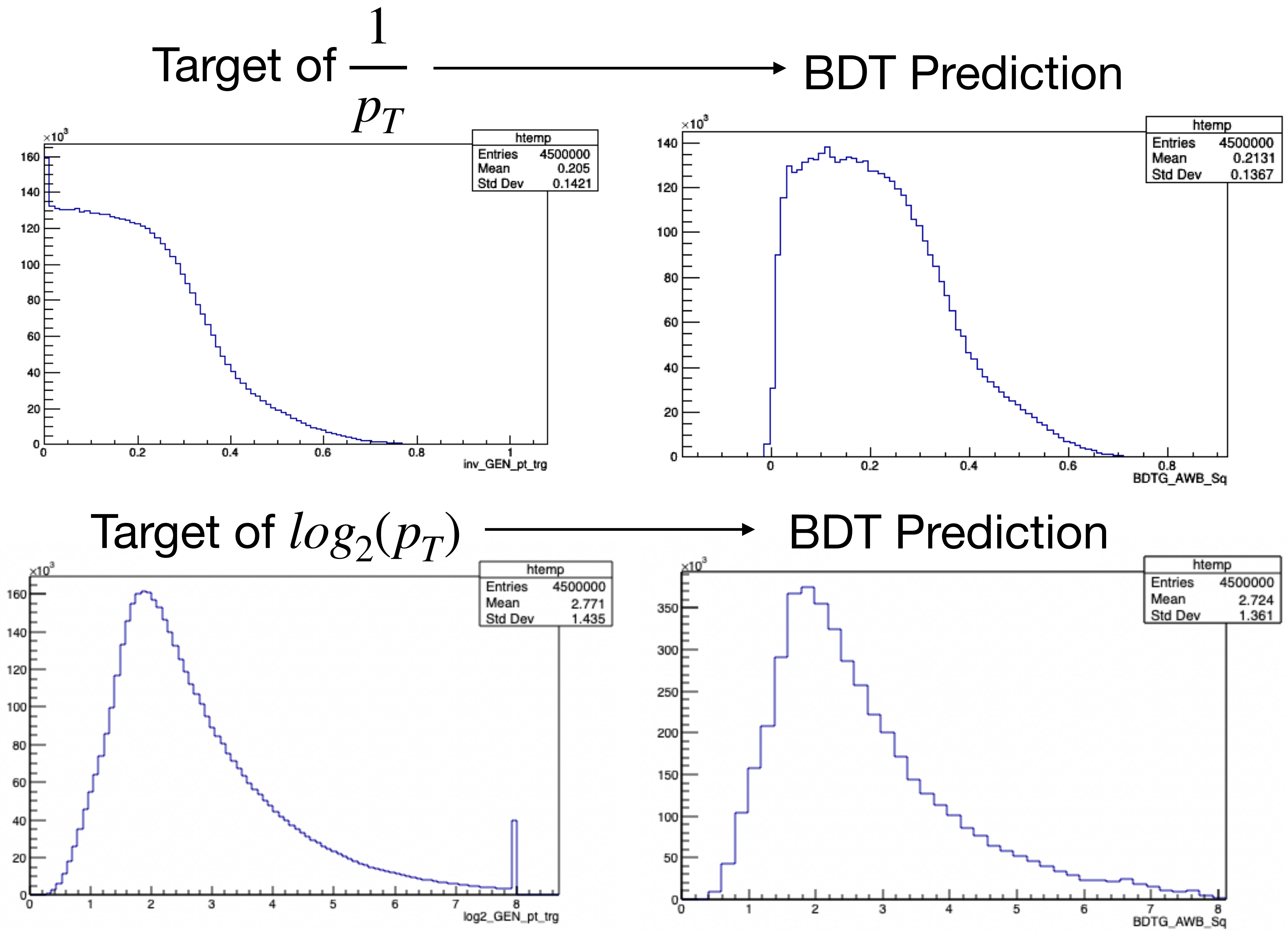
- This is done until the forest size reaches the limit we set in the hyperparameters

Target Variables

- With machine learning algorithms, the regression generally performs better with smooth output spaces

- The algorithm will tend to smooth over sharp peaks

- Playing with the target variable to choose the smoothest output space will improve performance!

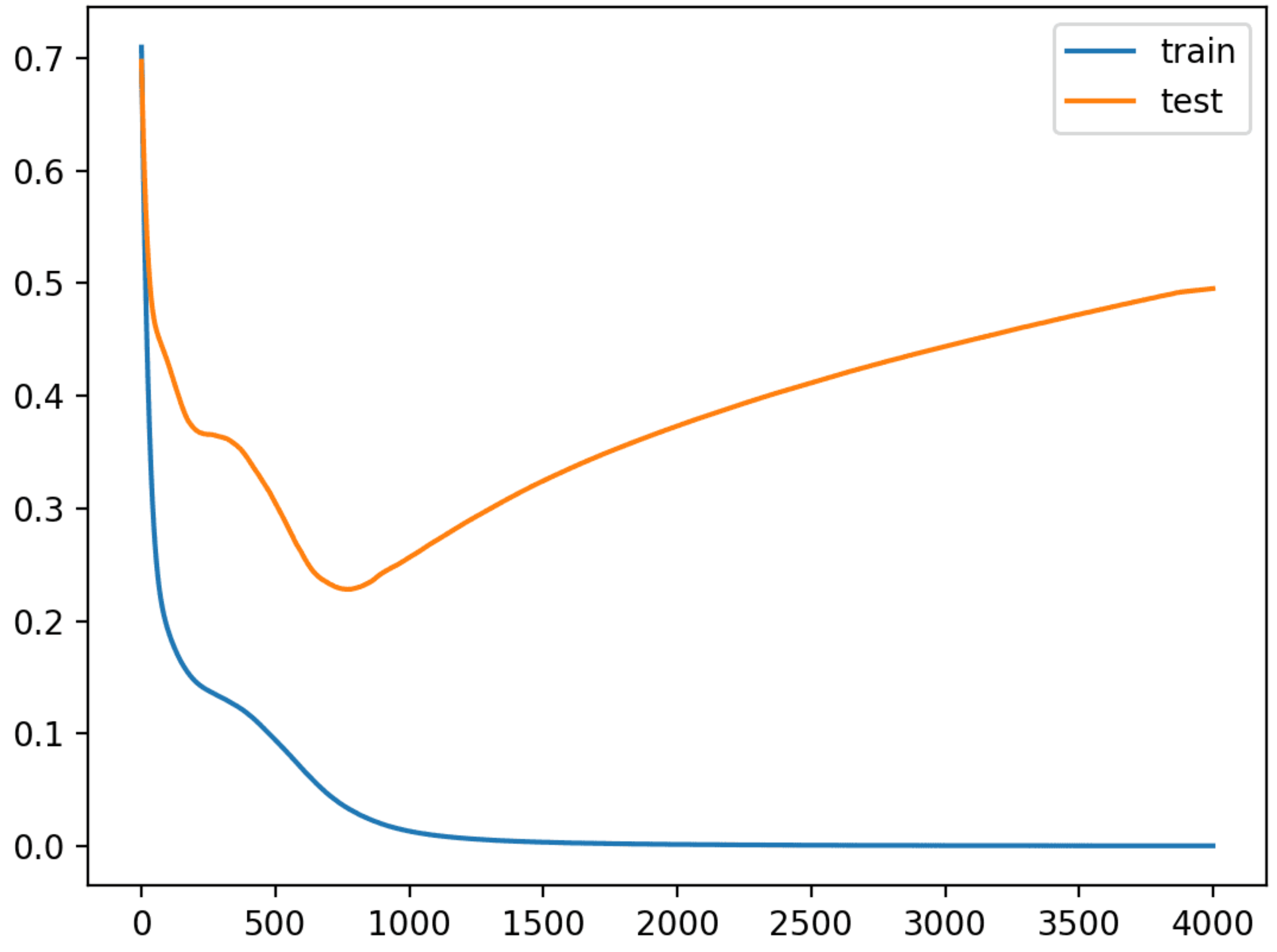


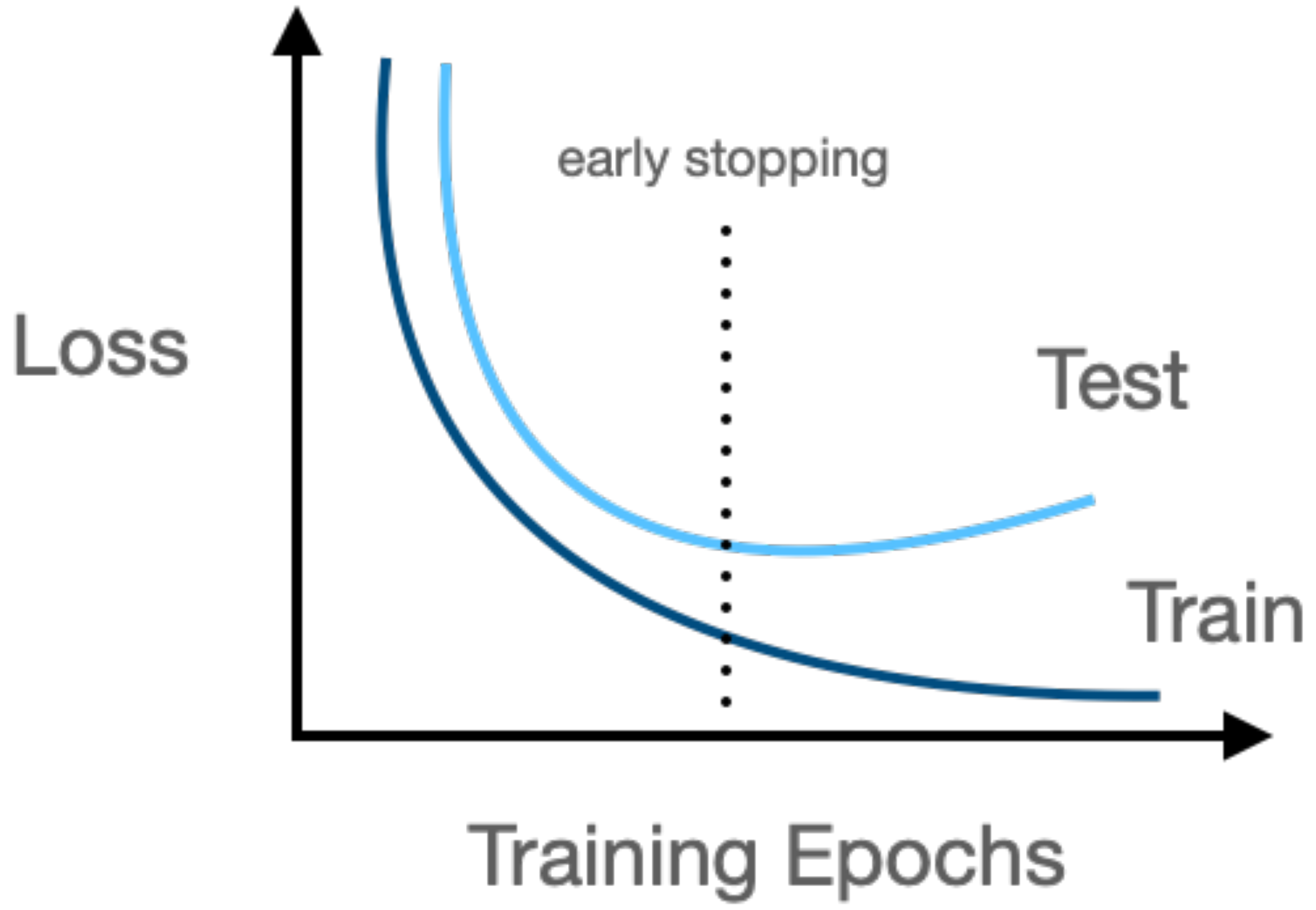
Target of $\frac{1}{p_T}$ → BDT Prediction

Target of $log_2(p_T)$ → BDT Prediction

Overtraining or Overfitting

- **Overtraining** is still an issue with BDTs
- This mainly results from using incorrect tree structure or too large of forests for the input datasets





Input Variables

- One of the largest setbacks of BDTs compared to NN is that the input variables must be well defined

- Unlike NN, BDTs will produce binary cuts based on the input variables and will optimize the cuts and leaf weights **based on a given input variable** as opposed to connections **between input variables.**

- As a result, it is important to construct input variables that will be useful to the BDT rather than give default values

- For Example: ϕ_A and ϕ_B are not useful on their own, but $\Delta\phi_{AB}$ is useful for the BDT! A NN may be able to create a similar variable over many layers but a BDT will try to create logic from ϕ_A and ϕ_B in separate weak learners.

Hyperparameters

- Regressor BDTs have tons of hyperparameters to mess with

- To name a few:
    - Batch Size
    - Number of Trees
    - Tree Depth
    - Learning Rate
    - Event weights
    - Target Variable
    - Loss function
    - Boosting Algorithm (And optimizer)

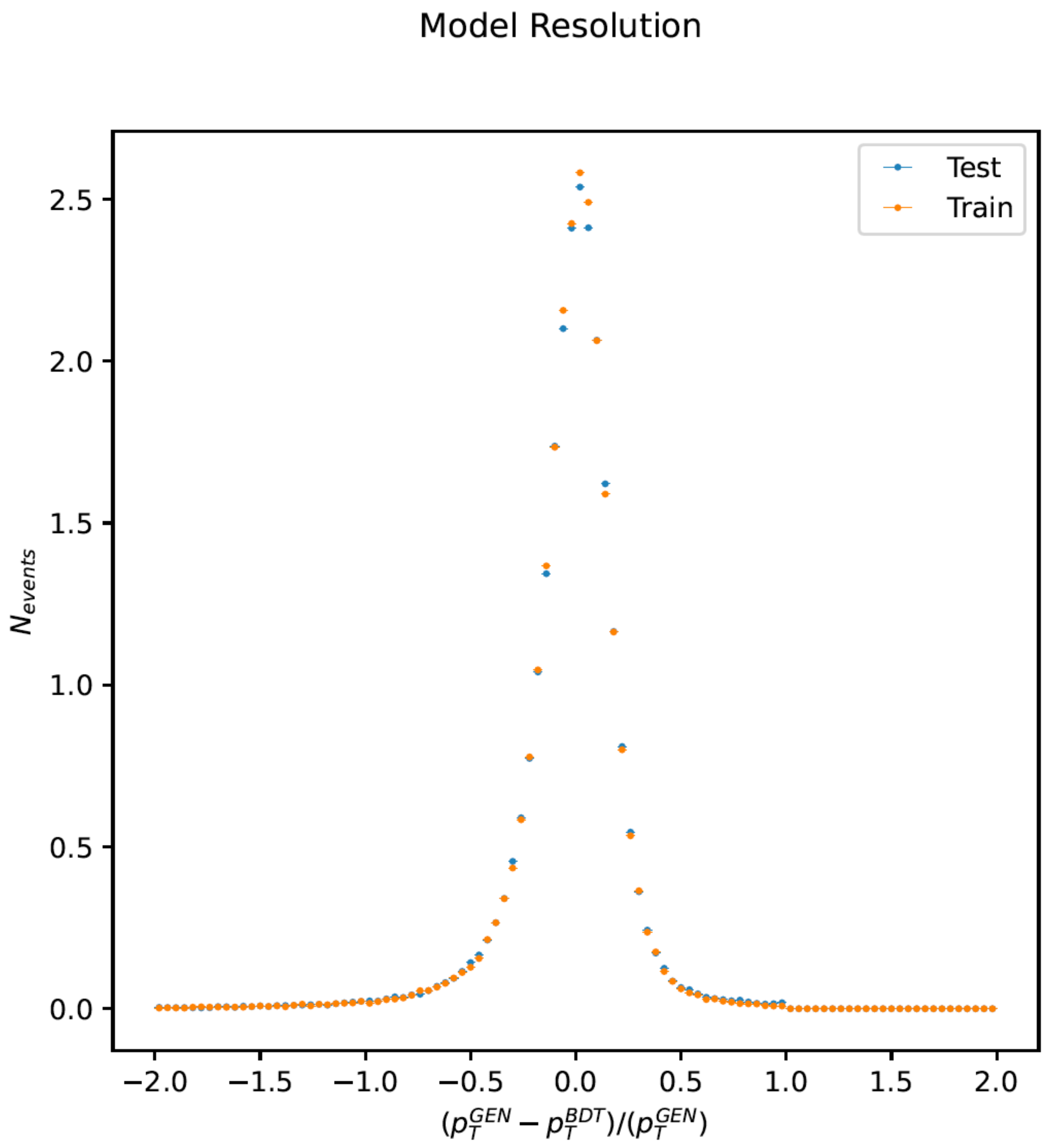
The Tutorial

- We will be training a regressor BDT to learn to regress a muons momentum in CMS based on hits in the muon chambers!

- We will be looking at simulated events that contain many variables given from the CSCs in the endcap

- The main metric we will be plotting will be the resolution of our regressor!

https://github.com/Offshell-Workshop-LPC/Offshell-Workshop-ML-Tutorials/tree/main/Tutorial_3


Model Resolution
Test
Train
N_{events}
$(p_T^{GEN} - p_T^{BDT})/(p_T^{GEN})$