ALERT Particle Identification with MLPs

CLAS Collaboration Meeting

June 29th, 2026

Uditha Weerasinghe

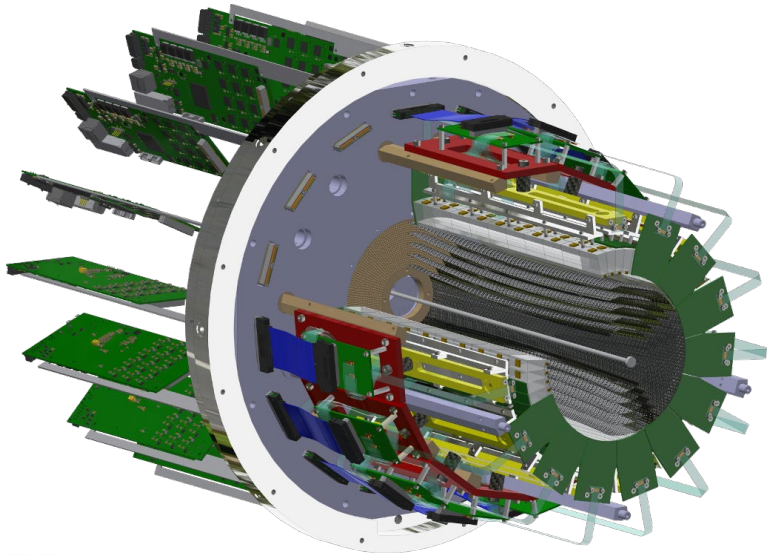
Mississippi State University

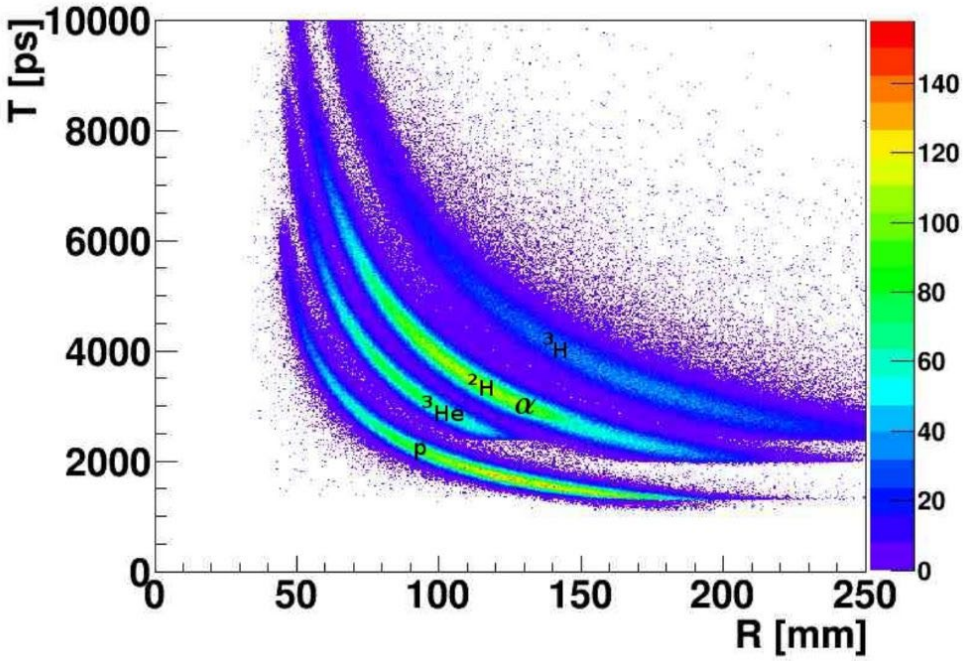
Outline

- Particle Identification in ALERT
- A Two-Stage AI PID Framework
  - Pre-KF PID
  - Post-KF PID
- Integration to coatjava
- Ongoing and Planned Improvements
- Summary and Outlook

Particle Identification in ALERT

- ALERT detects recoil nuclear fragments:
  - ^{1}H, ^{2}H, ^{3}H
  - ^{3}He, ^{4}He
- Proposal assumptions:
  - ALERT Time of Flight (ATOF) resolves most species
  - ALERT Hyperbolic Drift Chamber (AHDC) resolves complementary combinations
- In practice:
  - Significant overlap between H and He isotopes
  - Traditional cut-based PID struggles in key kinematic regions





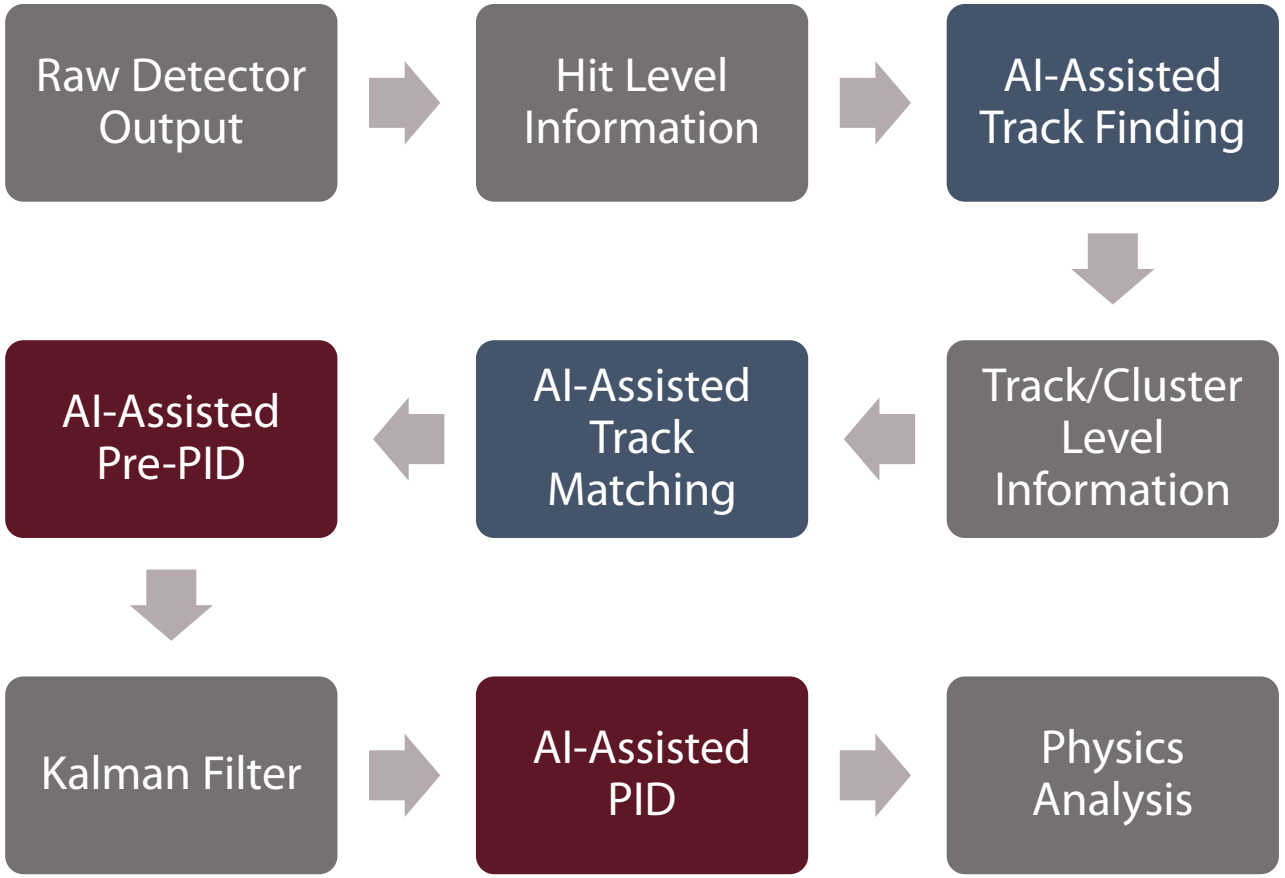
A Two-Stage AI PID Framework

Pre-KF PID

- Uses AHDC track + matched ATOF hit
- Provides fast probabilistic classification
- Intended to assist Kalman Filter (KF)

Post-KF PID

- Uses KF-refined kinematics
- Final physics-level classification
- Improved accuracy



Training Datasets

- A training dataset of 10M single-particle tracks was prepared
- After running through reconstruction ~25% of the tracks were reconstructed, with ~10% containing matched ATOF hits
- For **Pre-KF PID**, all events with a reconstructed track (AHDC::track was present) were selected without considering if an ATOF hit was matched
- For **Post-KF PID**, only events with a Kalman-Filter reconstructed track (AHDC::kftrack was present) and a matching ATOF::hit (ALERT::ai:prediction having a matching pair) were selected
- The two datasets were then individually class-balanced across five particle species[^{1}H, ^{2}H, ^{3}H, ^{3}He, ^{4}He] and split into training, testing, and validation samples

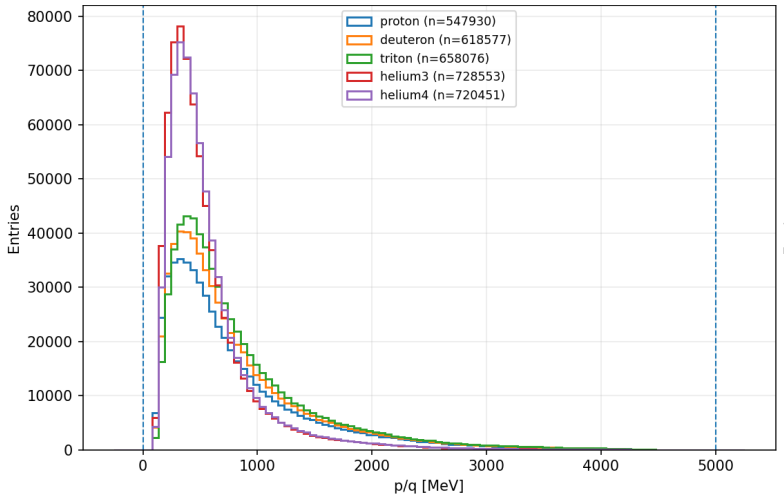
Pre-KF PID: Features

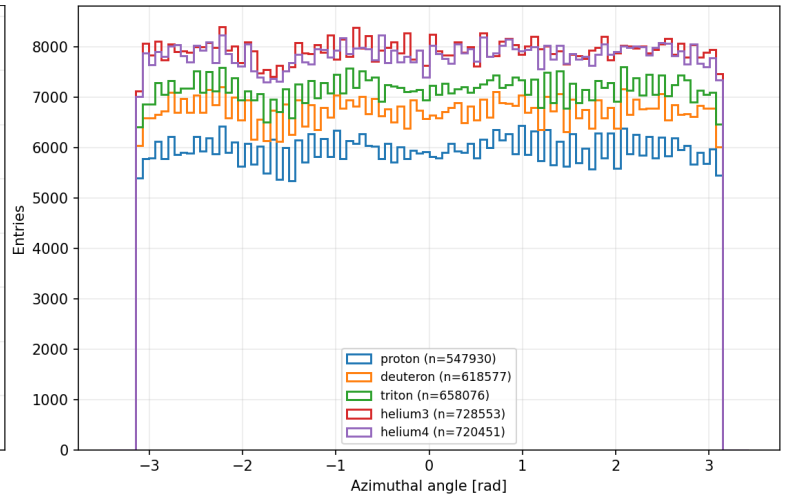
AHDC::track (10 features)

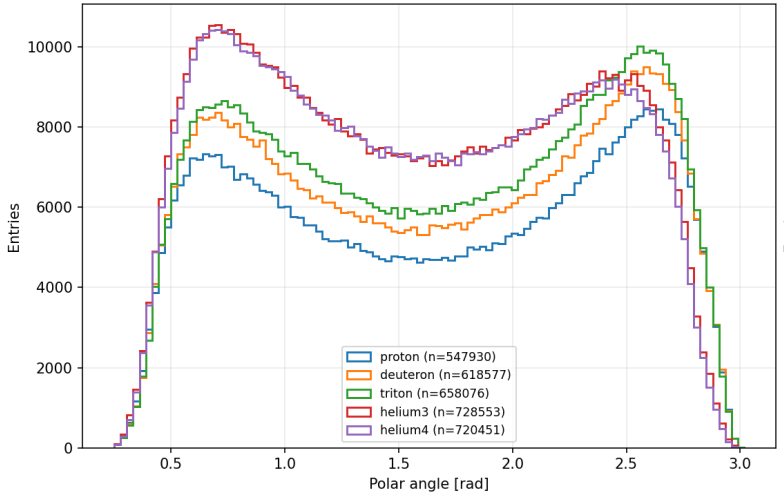
- x, y, z
- p_x, p_y, p_z
- n_{hits}
- sum(adc)
- χ^2
- sum_residuals

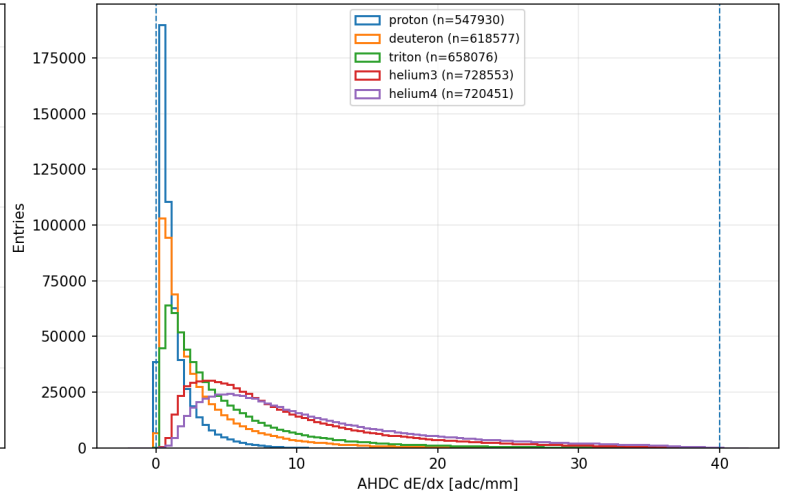
ATOF::hits (5 features) [where available]

- time
- x, y, z
- energy









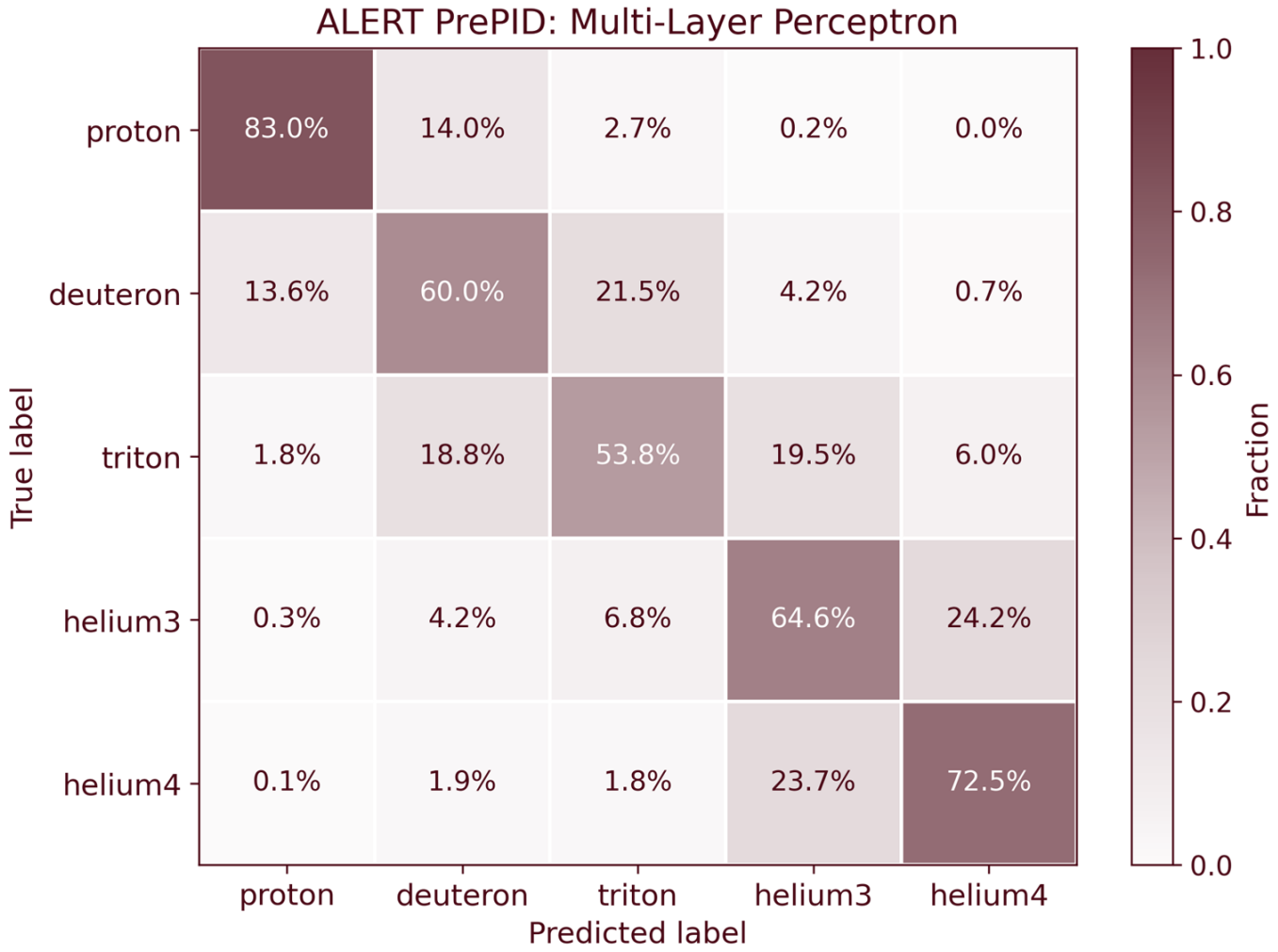
Pre-PID: Candidate Models

- To evaluate the best modeling strategy for this classification task, we trained and optimized two complementary model types:
  - **MLP:** The MLP architecture was already implemented in the workflow, so if it achieved better performance, it would be easier to integrate and continue using. It can also learn complex nonlinear feature interactions through multiple hidden layers.
  - **CatBoost:** CatBoost provides a strong gradient-boosted decision tree baseline, especially for structured/tabular data. It can capture nonlinear feature splits with limited preprocessing and often performs well without requiring a deep neural network architecture.
- Using both models allows us to compare a readily available neural-network-based classifier against a strong tree-ensemble-based classifier.
- Hyperparameter optimization with Optuna was used to tune each model fairly and identify the best-performing configuration for validation accuracy

Pre-PID: MLP Optimization

Best MLP configuration:

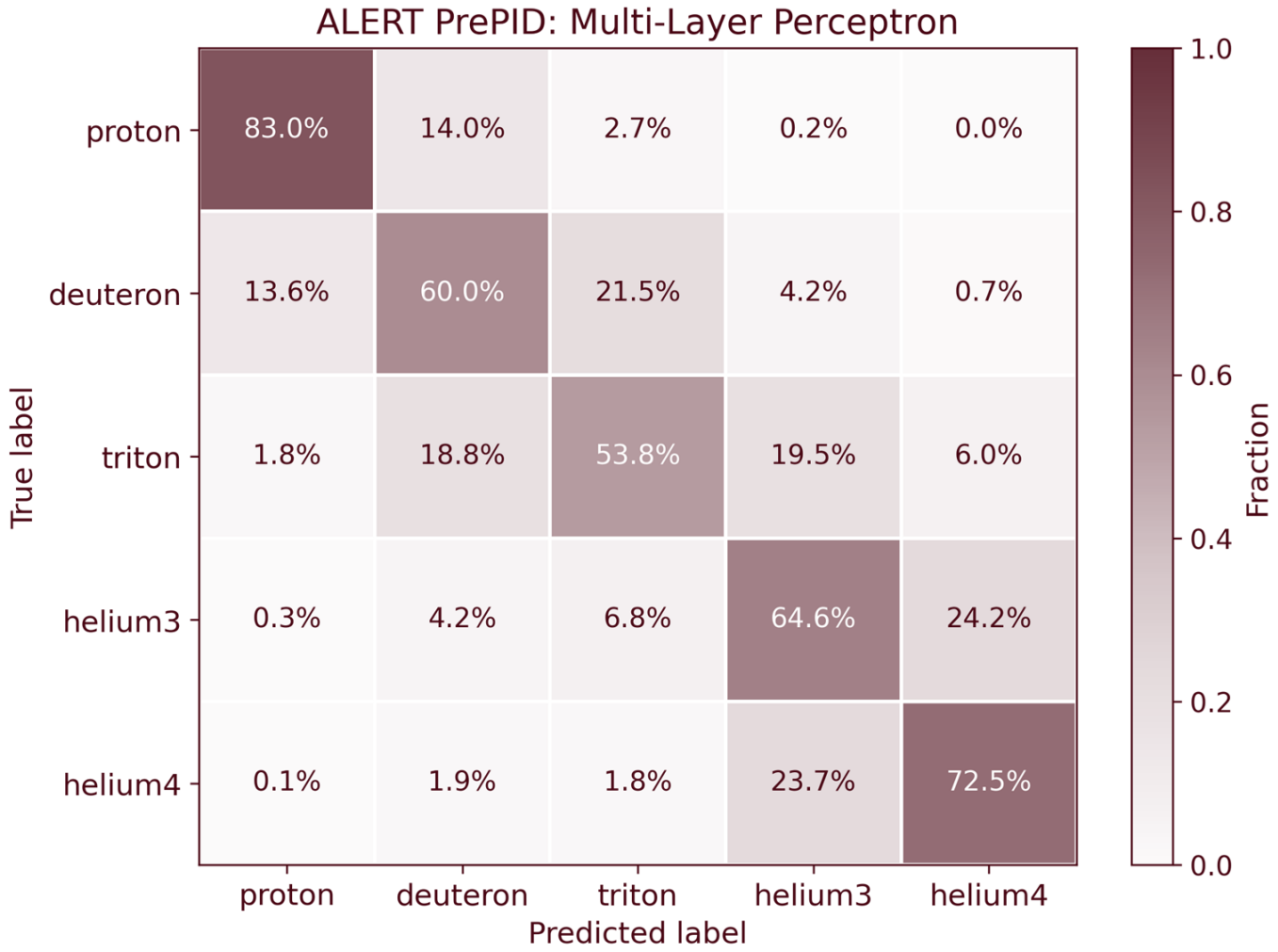
- **Activation:** GELU
- **Batch normalization:** Enabled
- **Batch size:** 1024
- **Hidden dimension:** 768
- **Number of layers:** 6
- **Dropout:** 0.179
- **Learning rate:** 3.02×10^{-4}
- **Weight decay:** 1.47×10^{-4}



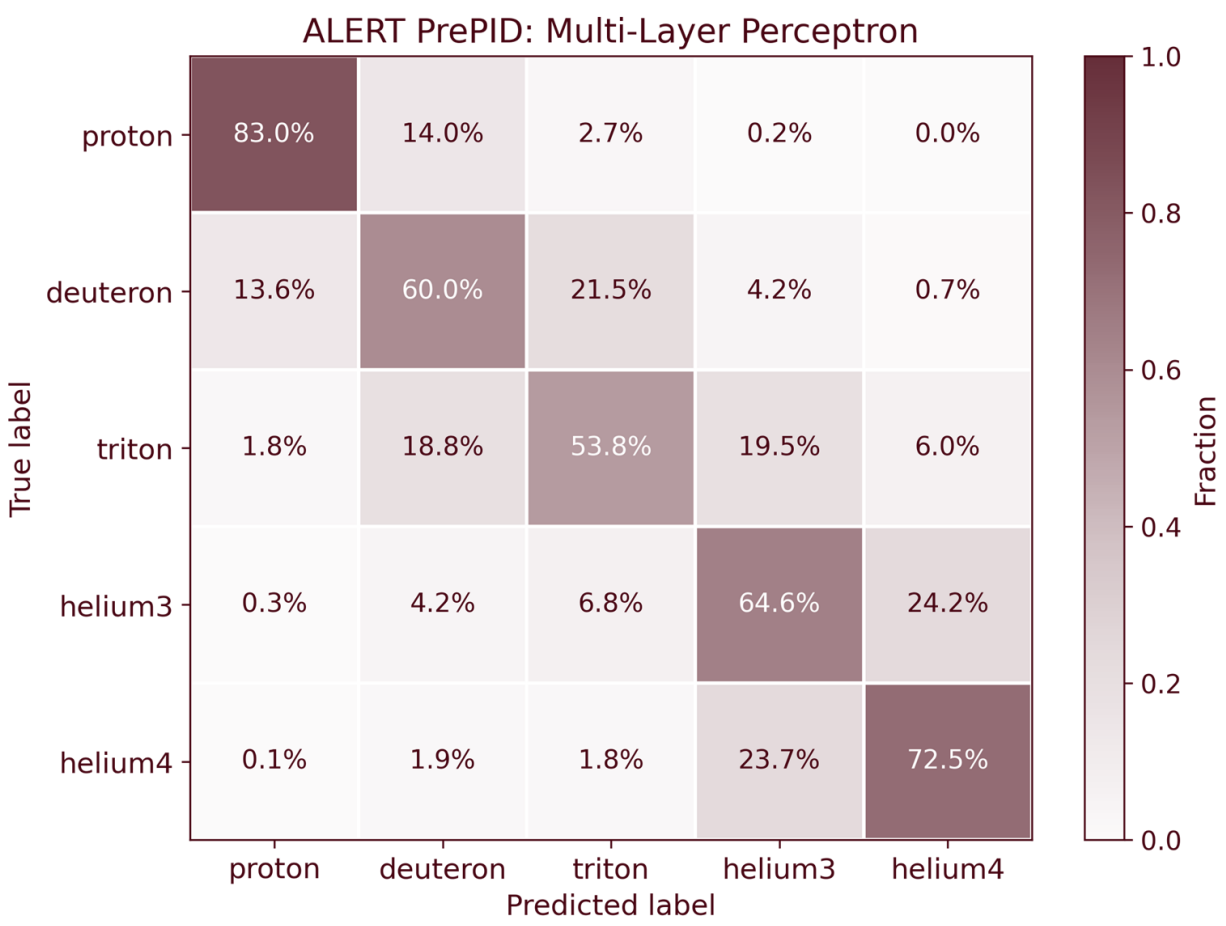
Pre-PID: CatBoost Optimization

Best CatBoost configuration:

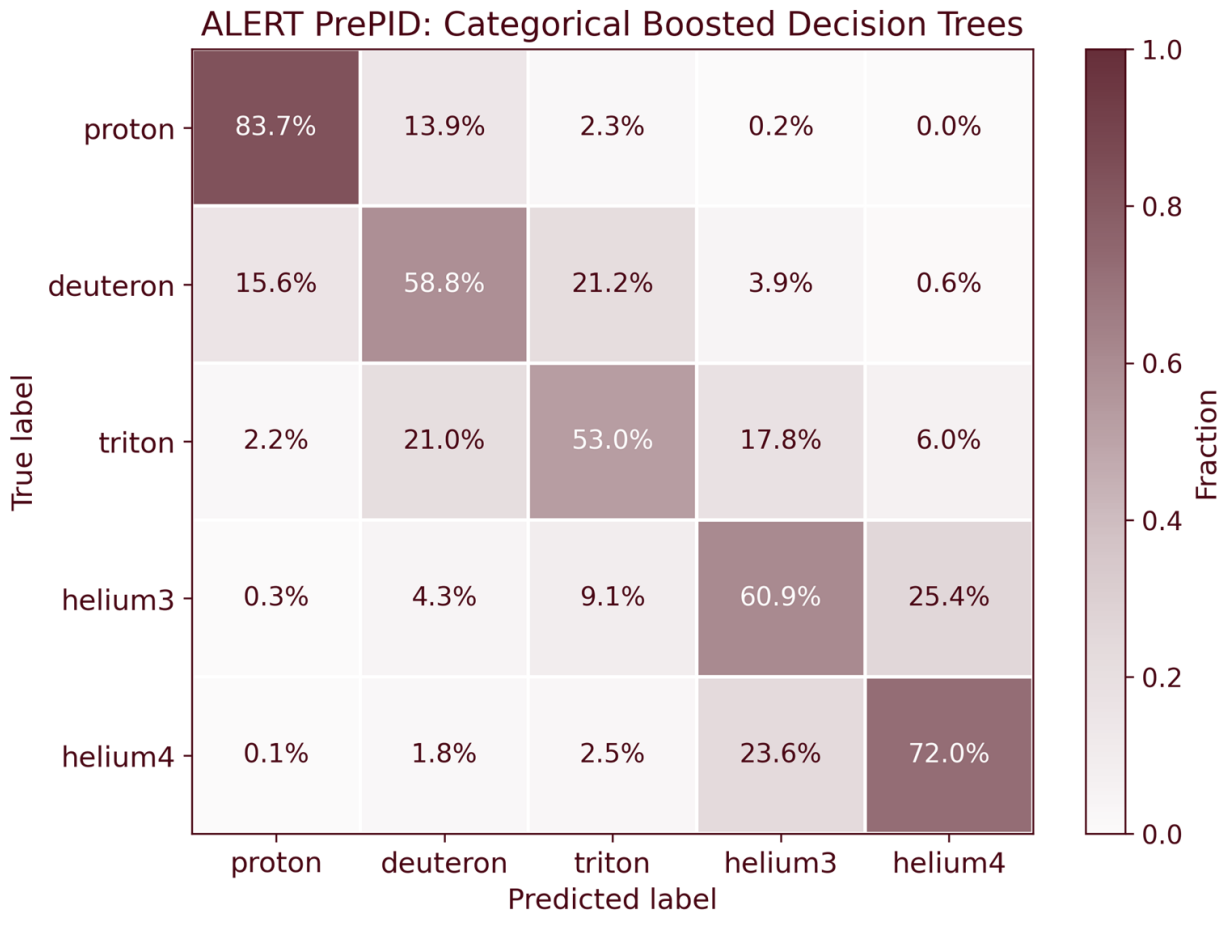
- **Bagging temperature:** 0.500
- **Border count:** 512
- **Tree depth:** 11
- **Iterations:** 4543
- **L2 leaf regularization:** 3.65
- **Learning rate:** 0.0630
- **Random strength:** 0.584



Pre-PID: Model Accuracy After Optimization

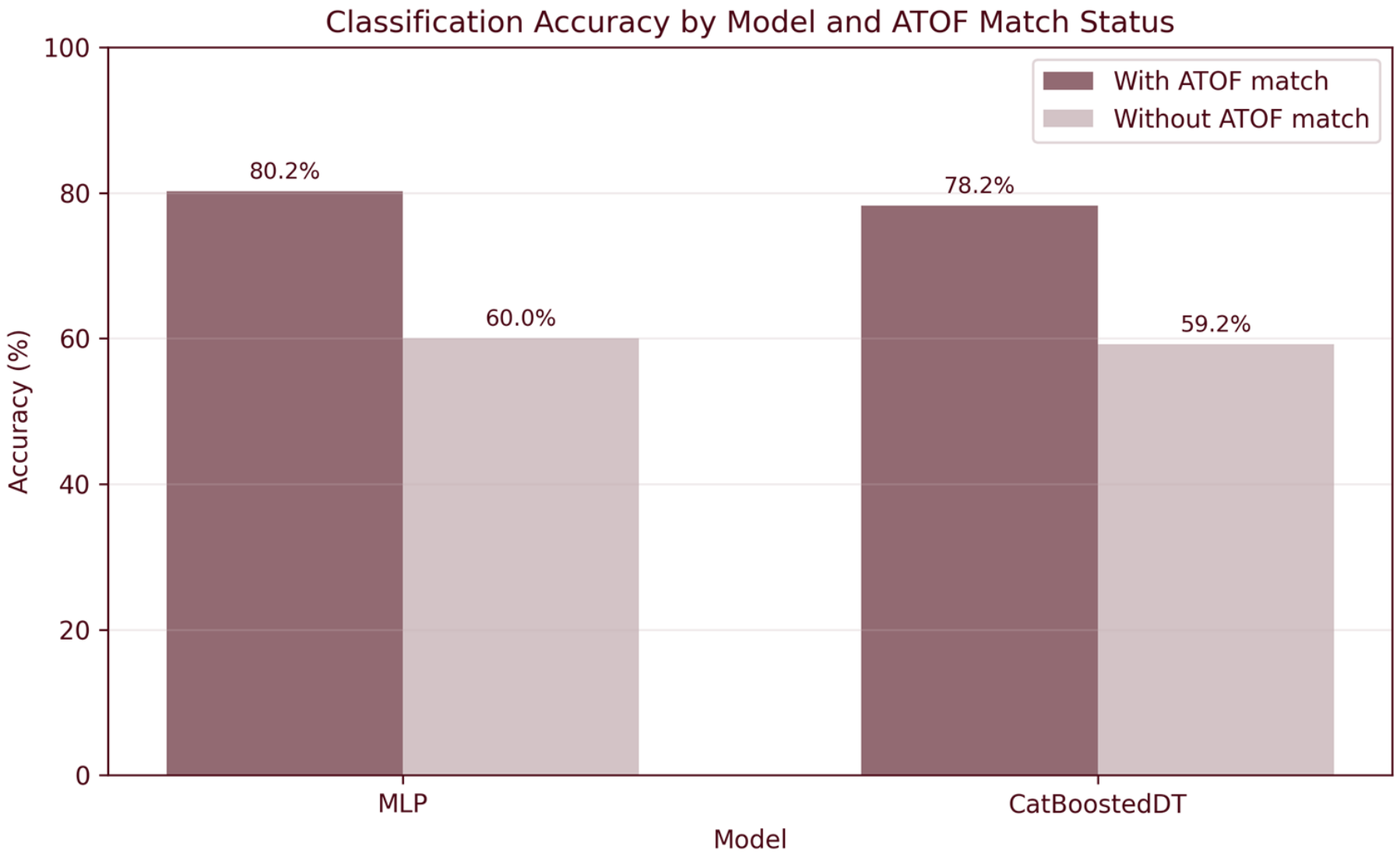


Avg Accuracy: 66.4%



Avg Accuracy: 65.2%

Pre-PID: Model Accuracy vs ATOF Match



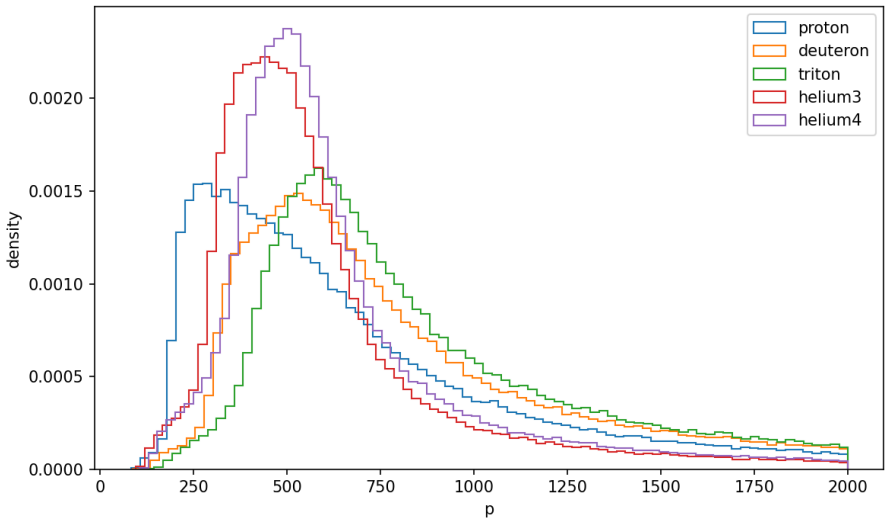
Post-KF PID: Features and Training

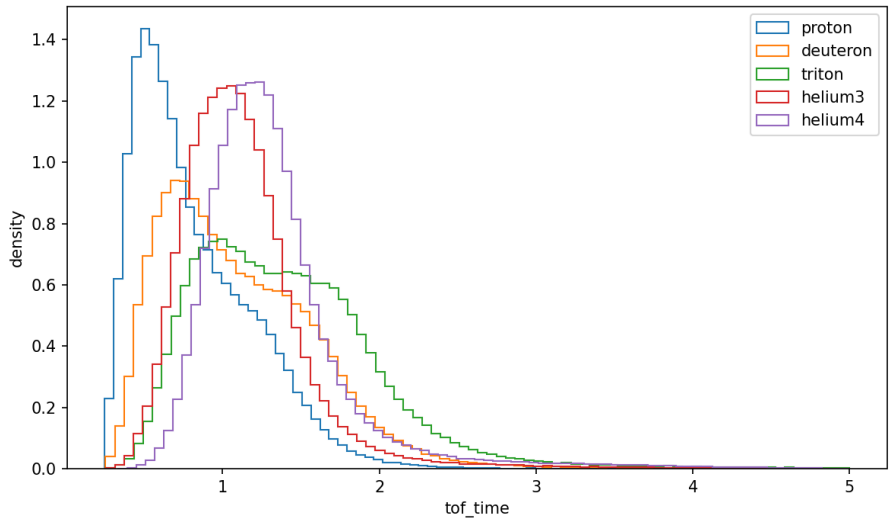
AHDC::kftrack (13 features)

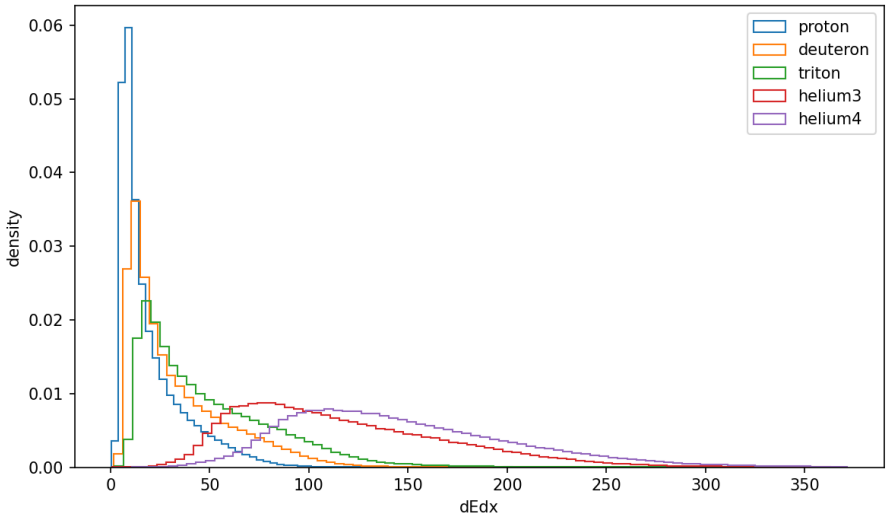
- x, y, z
- p_x, p_y, p_z
- n_{hits}, sum(adc)
- path, dE/dx
- p_{drift}, χ^2
- sum_residuals

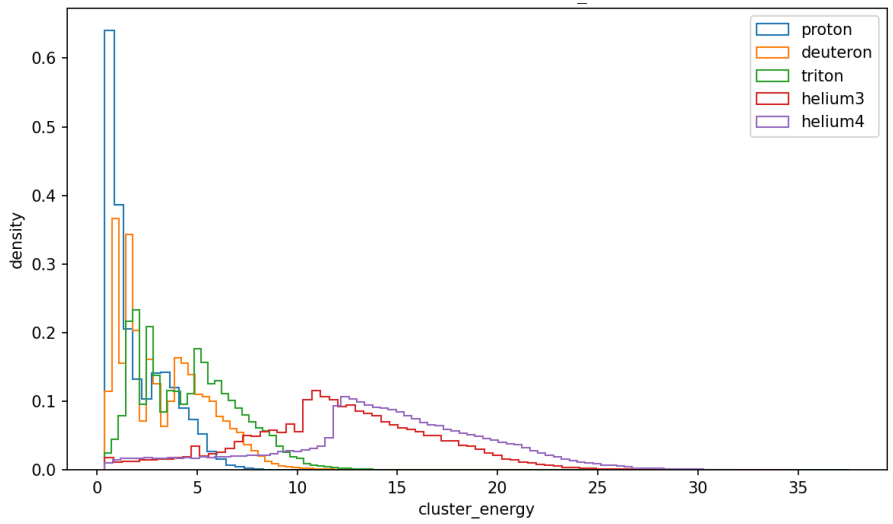
ATOF::clusters (9 features)

- n_bar, n_wedge
- time
- x, y, z
- energy
- pathlength, inpathlength



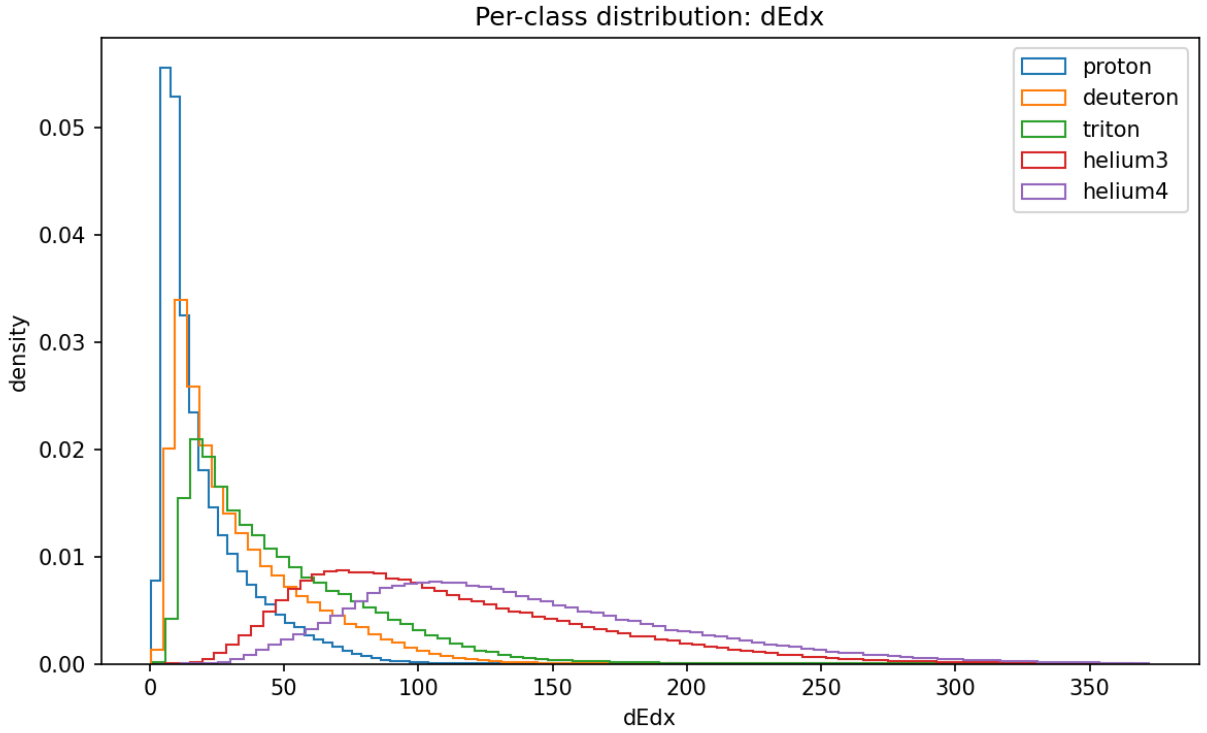


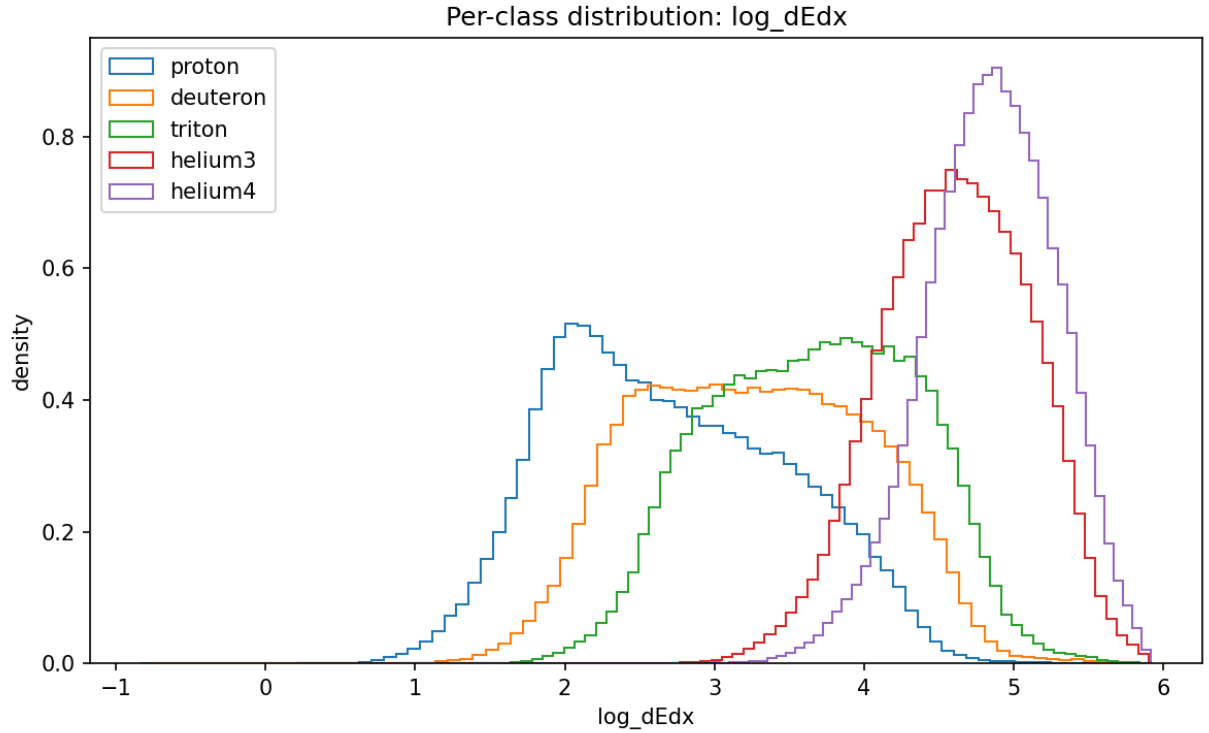




Feature Engineering (an example)

- Some features have long tails, which causes the normalization to be skewed.
- Easiest fix is to take the log scale





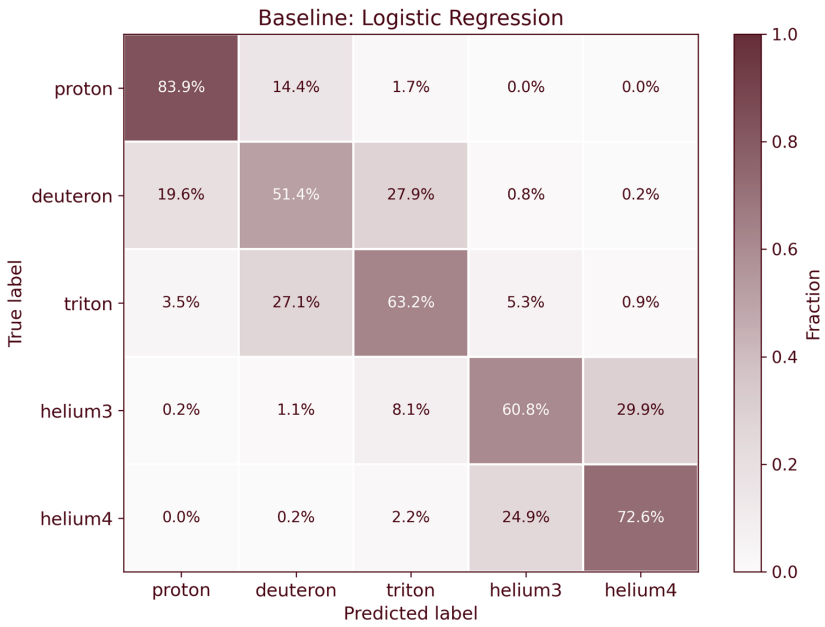
Post-KF PID Models

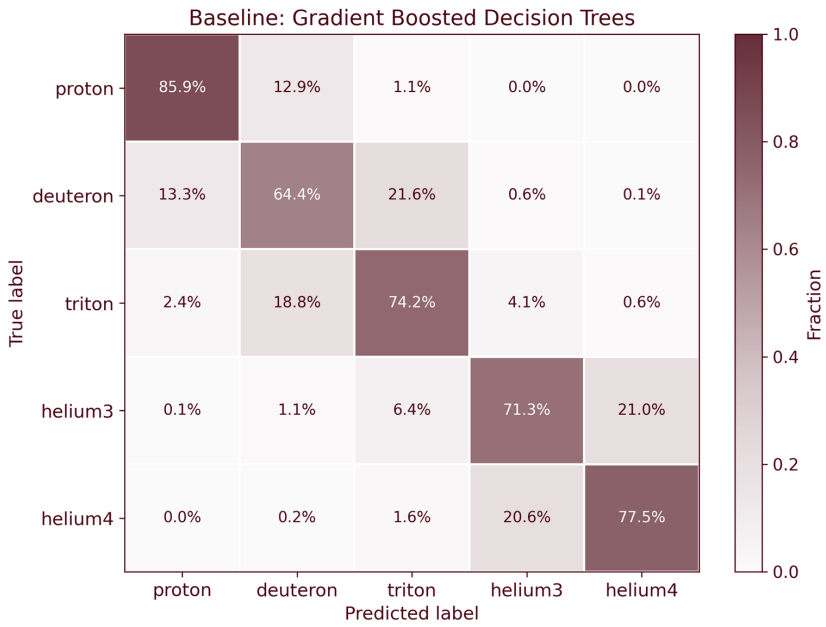
- Attempted to train the post-KF PID model
- Even though the final implementation was decided to be MLP, a few other models were also checked:
  - Logistic Regression
  - Gradient Boosted Decision Trees
  - Multi-Layer Perceptron
- MLP and GBDT show a good improvement over the pre-PID models

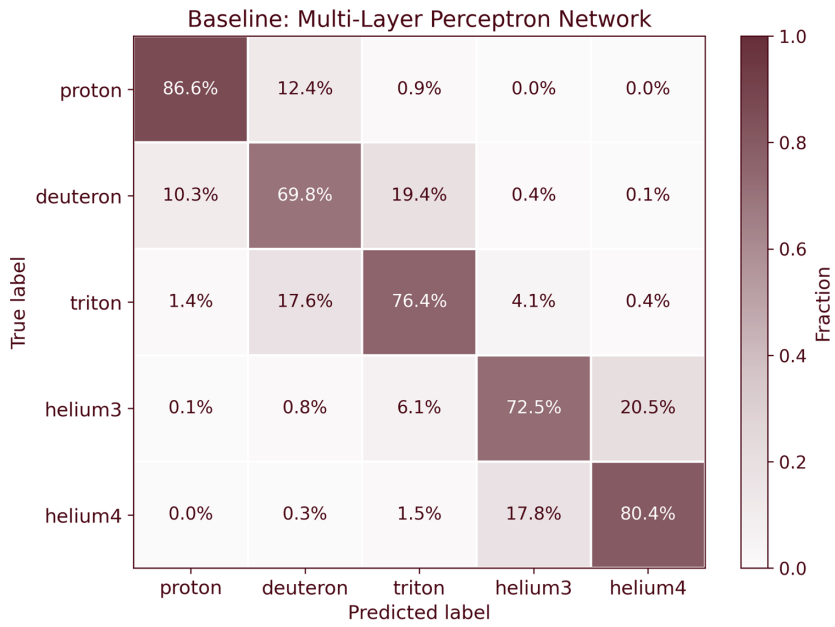
- The MLP and GBDT models were then optimized
- Unless there is a drastic difference in accuracy, the MLP model will be implemented

Post-KF PID Models

- Overall Accuracies:
  - Log-Reg: 65.7%
  - GBDT: 74.2%
  - MLP: 76.7%
- (The GBDT and MLP are ~5% better than the comparable models of pre-PID)

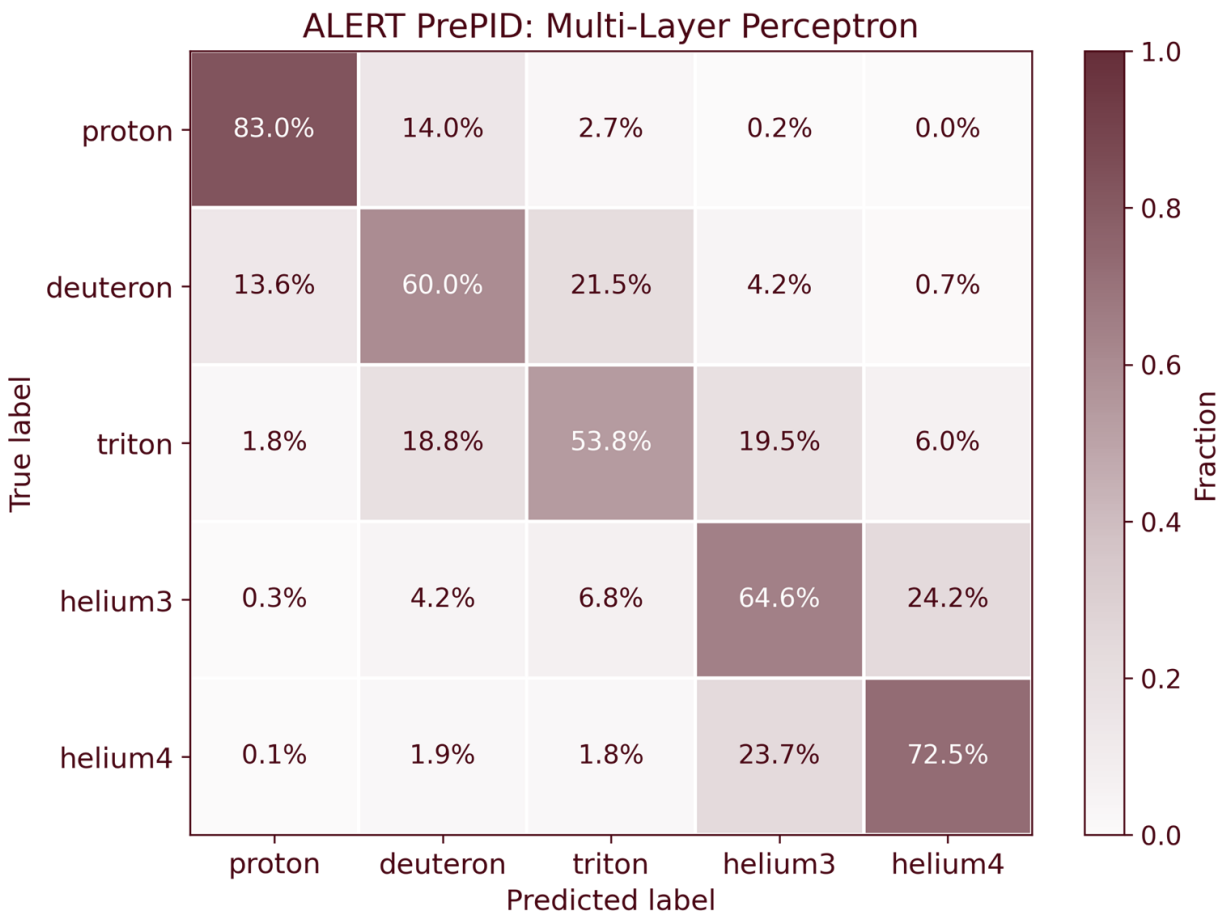


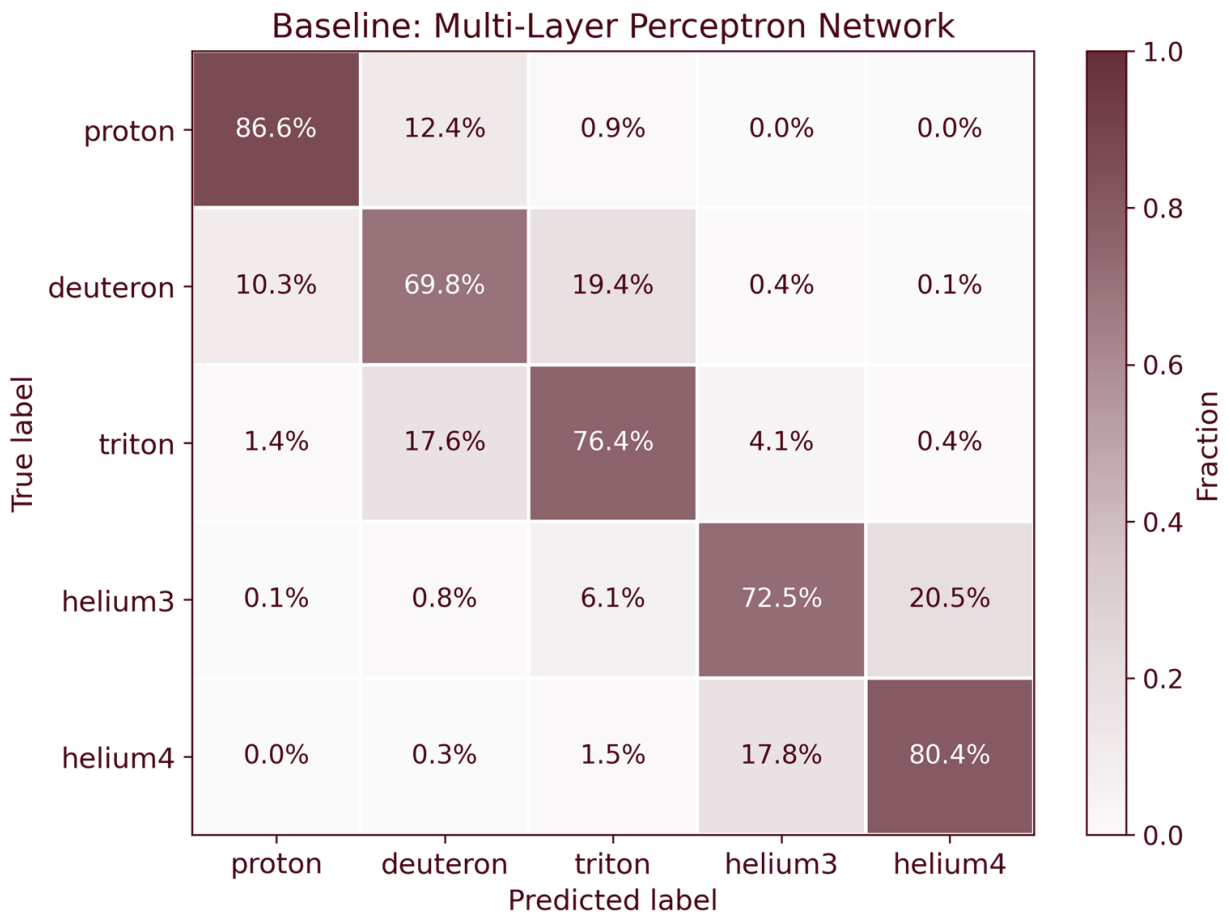




Post-KF PID Models

- MLP comparison with pre-KF PID





Integration to coatjava

PrePID:

- Initially implemented in coatjava (see pull request #1112)
- An update is imminent

Post-KF:

- Architecture prepared
- Will be integrated after a few bug-fixes

Output Format:

- Output is written to `ALERT::ai:prepid` / `ALERT::ai:pid` bank:
  - **`trackid`** – matched AHDC track
  - **`clusterid`** – matched ATOF ~~cluster~~ hit
  - predicted class
  - class probabilities

```
 position for [ALERT::ai:prepid]
* NODE * group =  23000, item =
       trackid :           1
     clusterid :           1
        prepid :          47
         p2212 :      0.0010
           p45 :      0.0290
           p46 :      0.2742
           p47 :      0.4855
           p49 :      0.2103


Choose (n=next,p=previous, q=qu
 position for [MC::Particle] ==
* NODE * group =     40, item =
           pid :          47
            px :    -0.8340
            py :    -0.8274
            pz :     0.3051
            vx :     0.0000
            vy :     0.0000
            vz :   -11.5744
            vt :   124.0000
```

Ongoing and Planned Improvements

- **Different AI models:** Testing and comparing different types of classification models in addition to the currently used MLP and GBT models
- **Hyperparameter optimization:** Systematic tuning of model hyperparameters using Optuna to improve classification performance and training stability.
- **Feature engineering studies:** Continued exploration of physics-motivated feature transformations to improve model learning and separation power.
- **Domain adaptation:** Investigating strategies to improve model robustness when applied to real detector data, reducing simulation–data discrepancies.
- **Systematic stability tests:** Evaluating model performance across different kinematic regions and detector conditions to ensure stable behavior.
- **Extended inference capability:** Modifying the Pre-PID model to produce particle predictions even when no matched ATOF hit is present, allowing the model to assist track reconstruction in more cases.

Ongoing and Planned Improvements

- ~~**Different AI models:** Testing and comparing different types of classification models in addition to the currently used MLP and GBT models~~
- ~~**Hyperparameter optimization:** Systematic tuning of model hyperparameters using Optuna to improve classification performance and training stability.~~
- ~~**Feature engineering studies:** Continued exploration of physics-motivated feature transformations to improve model learning and separation power.~~
- **Domain adaptation:** Investigating strategies to improve model robustness when applied to real detector data, reducing simulation–data discrepancies.
- **Systematic stability tests:** Evaluating model performance across different kinematic regions and detector conditions to ensure stable behavior.
- ~~**Extended inference capability:** Modifying the Pre-PID model to produce particle predictions even when no matched ATOF hit is present, allowing the model to assist track reconstruction in more cases.~~

Summary and Outlook

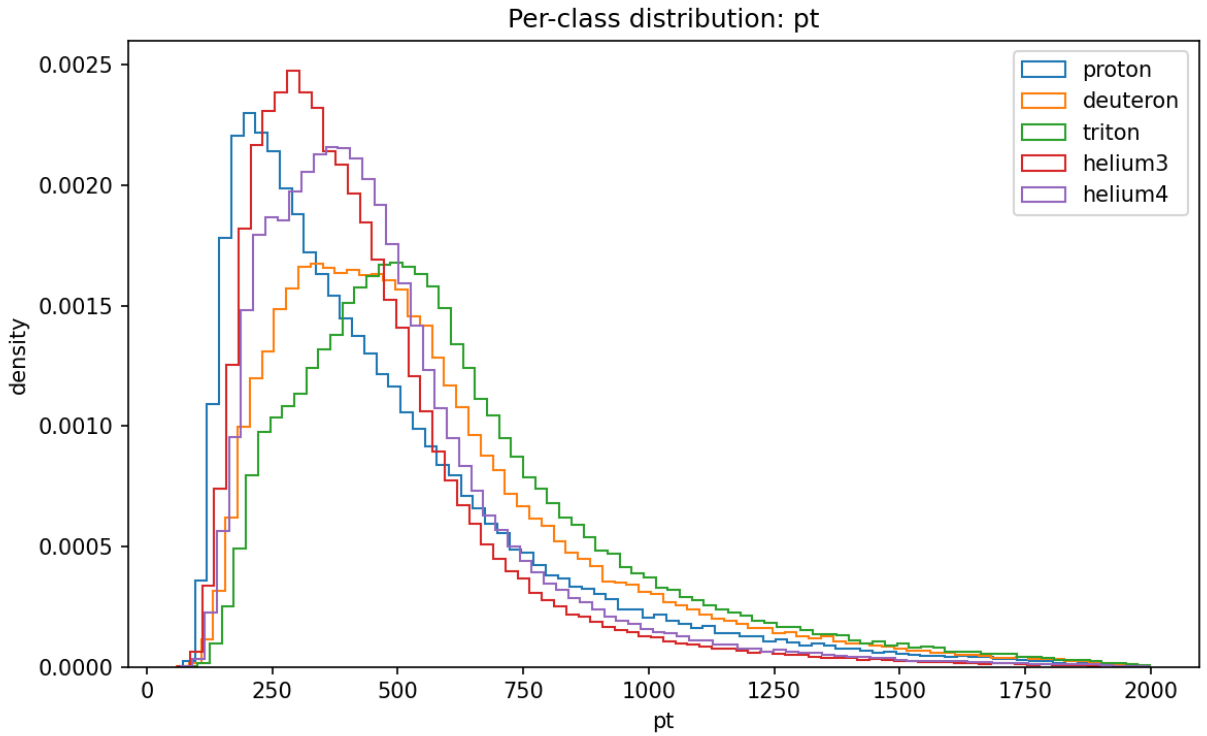
- Two-stage AI PID framework was developed to identify nuclear recoil fragments in ALERT
- PrePID model is currently implemented in coatjava, but an update is forthcoming
- Post-KF PID demonstrates above 80% accuracy for simulated data and will be implemented in coatjava soon
- Real-data performance is under active study
- Future improvements are in development

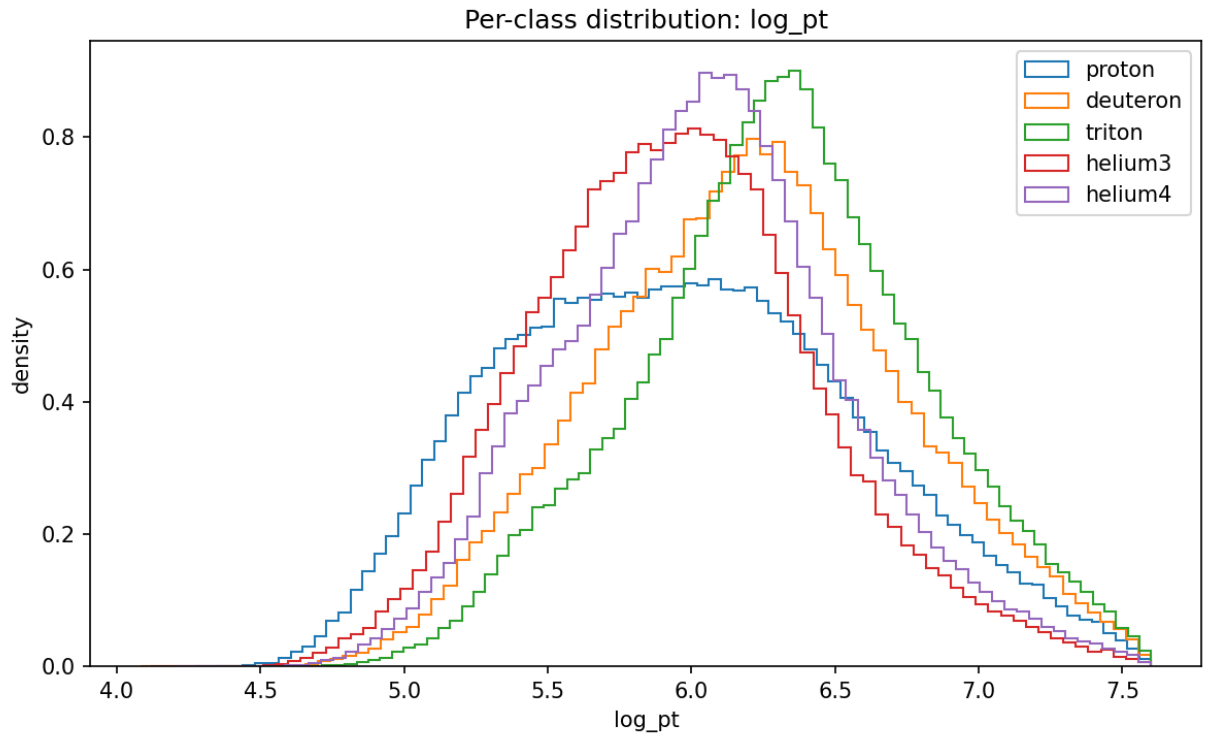
Thank You!

Backup Slides

Feature Engineering (another example)

- Some features have long tails, which causes the normalization to be skewed.
- Easiest fix is to take the log scale

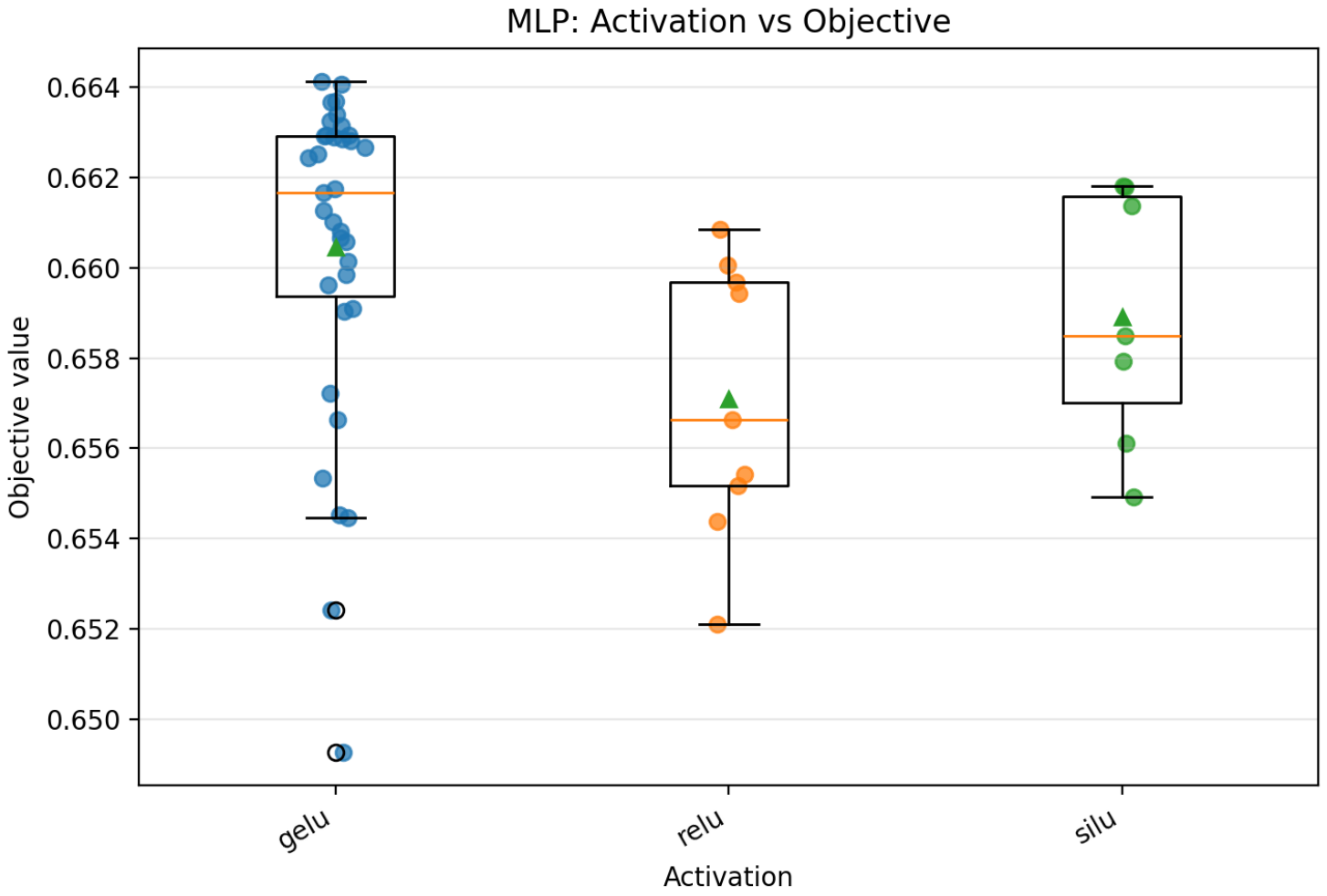




Pre-PID: MLP Optimization

Activation functions tested

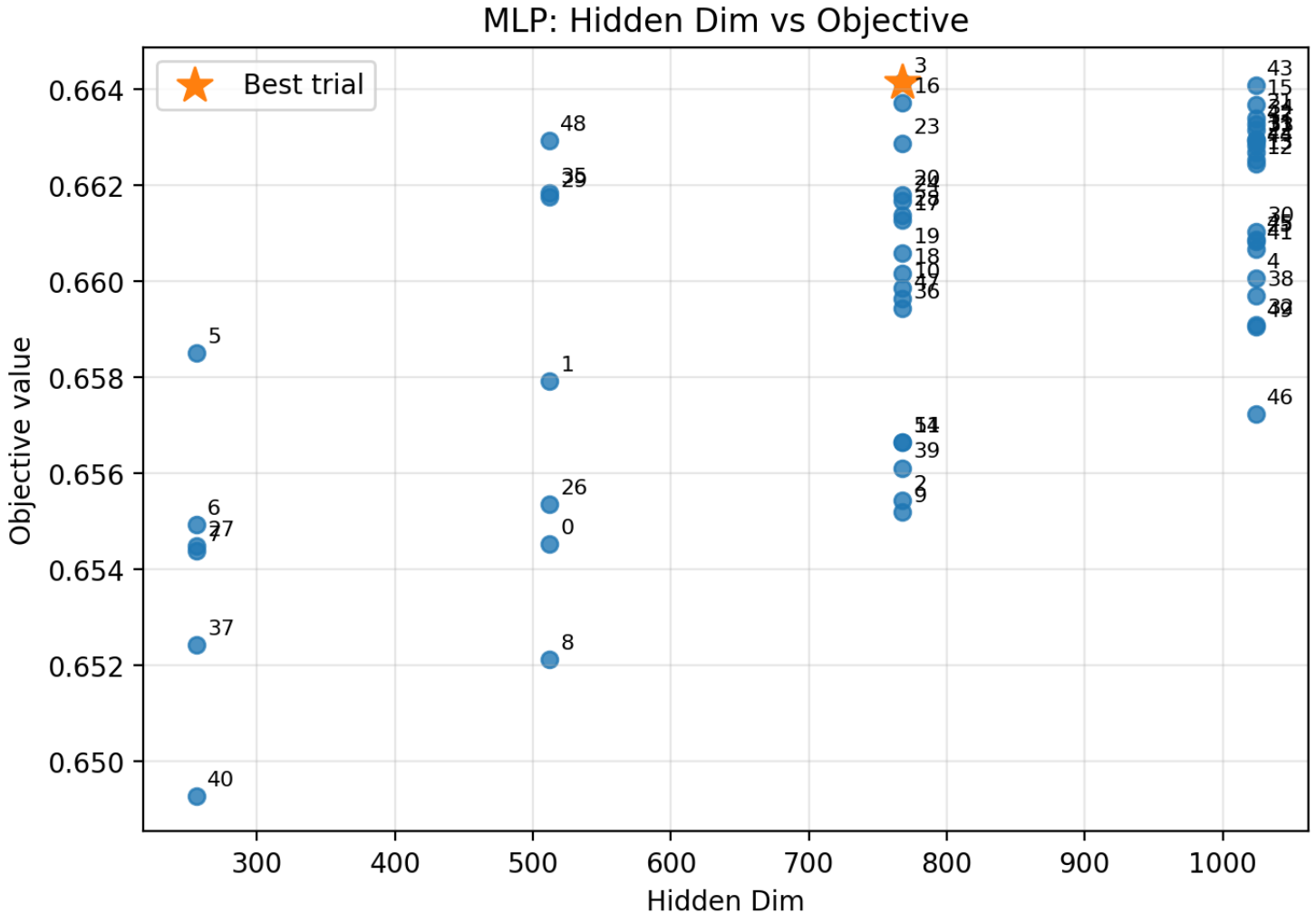
- **ReLU — Rectified Linear Unit:** Simple and efficient activation that keeps positive values unchanged and sets negative values to zero. Often a strong baseline for neural networks.
- **GELU — Gaussian Error Linear Unit:** Smooth activation that weights inputs by their probability under a Gaussian distribution. Commonly used in modern deep learning models and can provide more stable gradients than ReLU.
- **SiLU — Sigmoid Linear Unit:** Also called Swish; a smooth activation where inputs are scaled by a sigmoid function. It can retain small negative signals and often improves optimization in deeper networks.



Pre-PID: MLP Optimization

Hidden layer dimensions tested

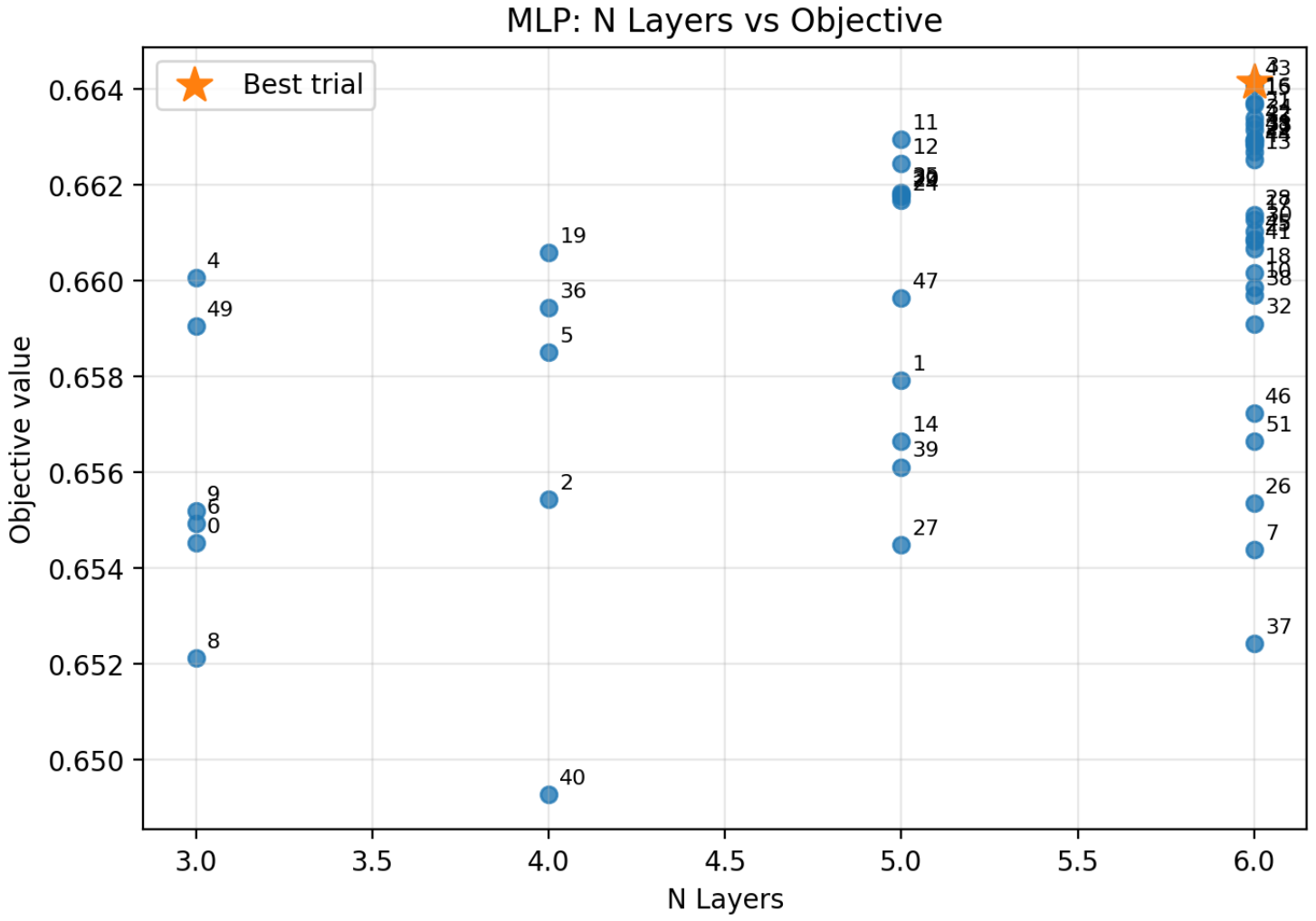
- **256:** Smaller hidden representation with fewer parameters. Faster to train, but may have limited capacity for complex patterns.
- **512:** Medium-sized representation that balances model capacity and computational cost.
- **768:** Larger representation that provided the **best performance** in this experiment, suggesting it captured useful feature interactions without excessive complexity.
- **1024:** Highest-capacity option tested. Can model more complex relationships, but may increase training cost and risk of overfitting



Pre-PID: MLP Optimization

Number of hidden layers tested

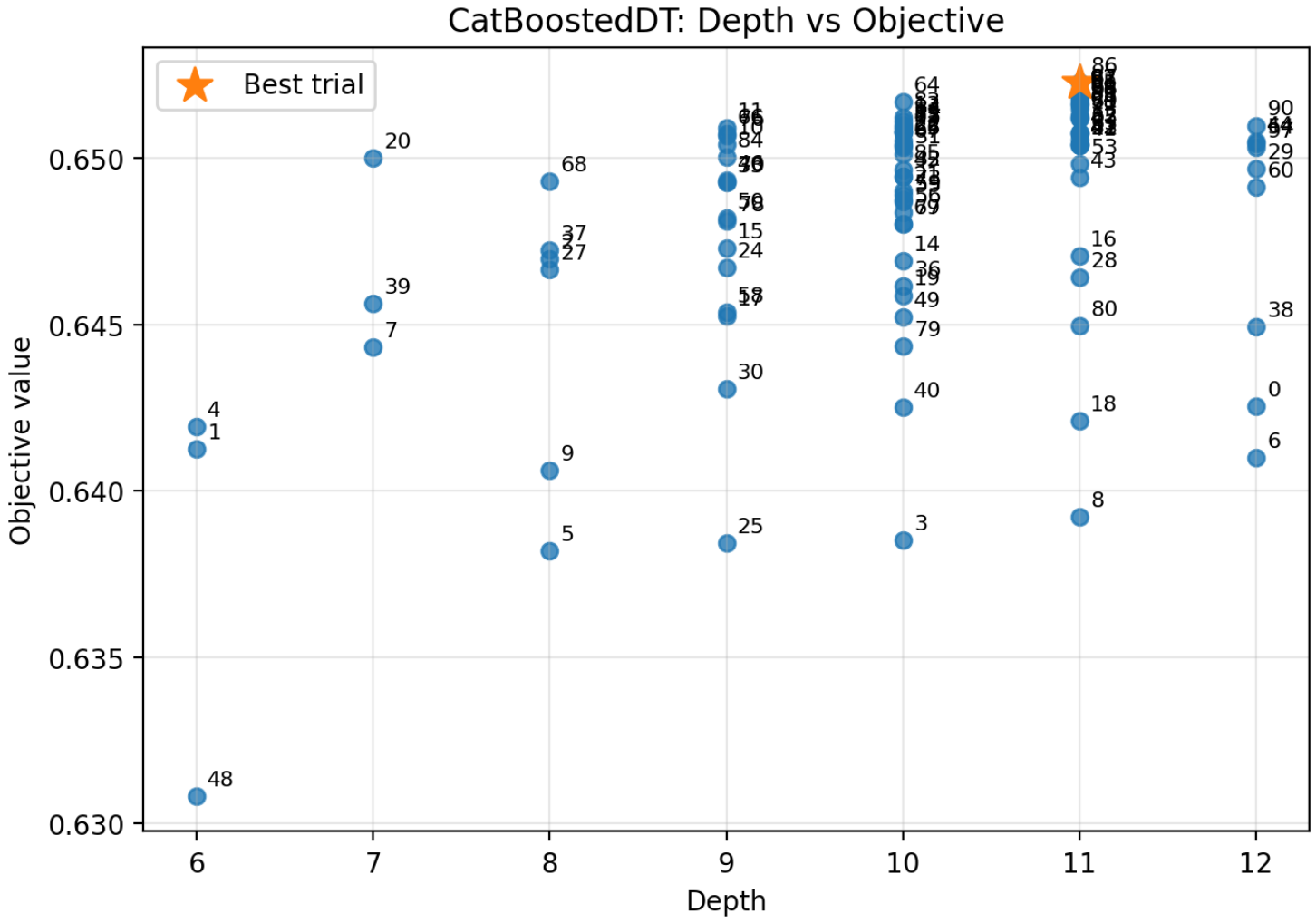
- **3 layers:** Shallower architecture with lower complexity and faster training, but may have limited ability to learn deeper feature interactions.
- **4 layers:** Moderate-depth architecture that increases model capacity while keeping training cost manageable.
- **5 layers:** Deeper architecture that can capture more complex nonlinear relationships, but may require stronger regularization and stable optimization.
- **6 layers:** Deepest option tested and gave the **best performance** in this experiment, suggesting the task benefited from additional model depth.



Pre-PID: CatBoost Optimization

Tree depth values tested

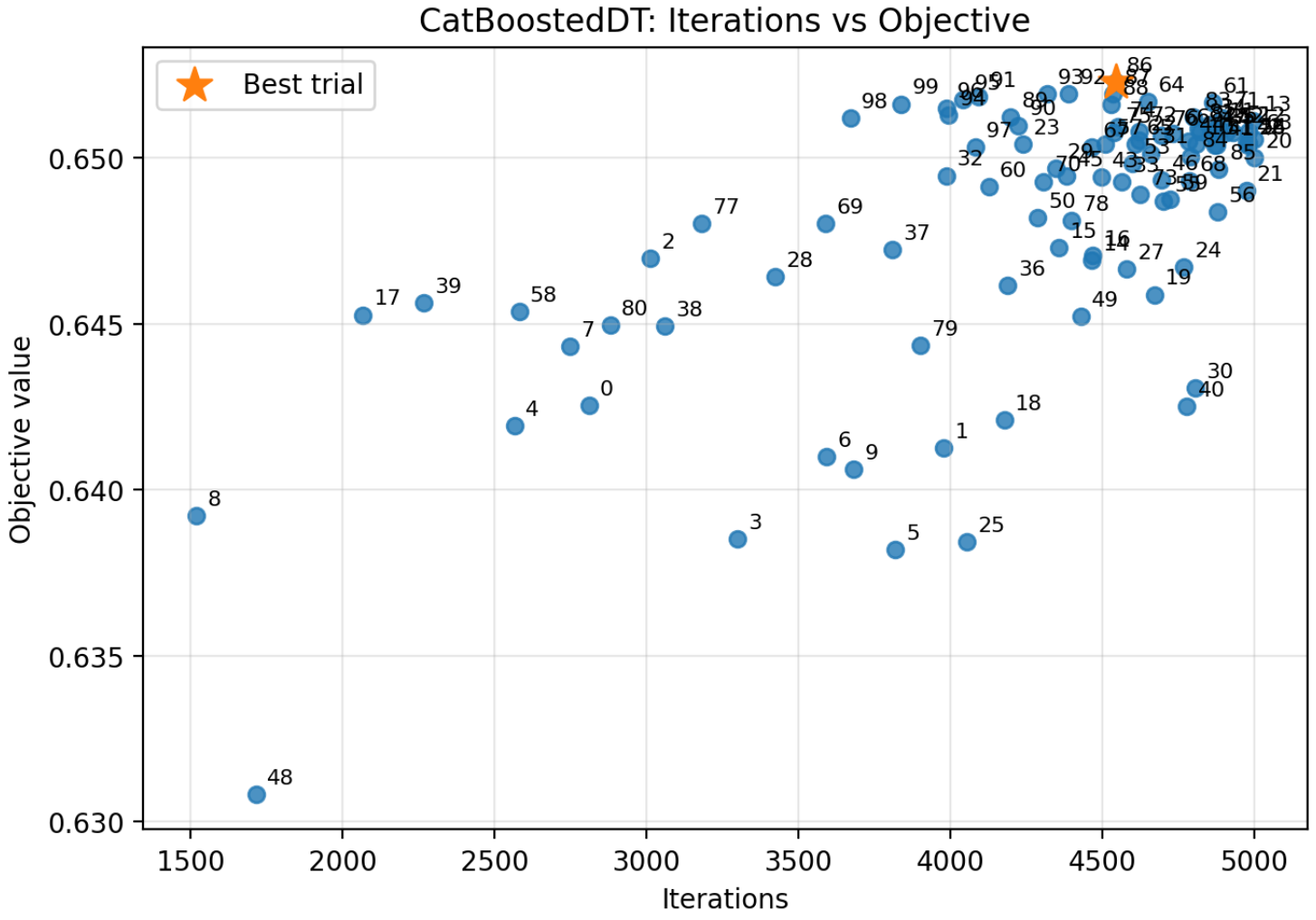
- **Depth = 6:** Shallower trees with lower model complexity. Faster to train and less prone to overfitting, but may miss more complex feature interactions.
- **Depth = 8–10:** Intermediate-depth trees that balance model flexibility, training cost, and generalization.
- **Depth = 11:** Selected by Optuna as the **best-performing value**, suggesting that relatively deep trees were useful for capturing nonlinear feature relationships.
- **Depth = 12:** Deepest option tested. Provides the highest per-tree complexity, but may increase training cost and risk of overfitting.



Pre-PID: CatBoost Optimization

Boosting iterations tested

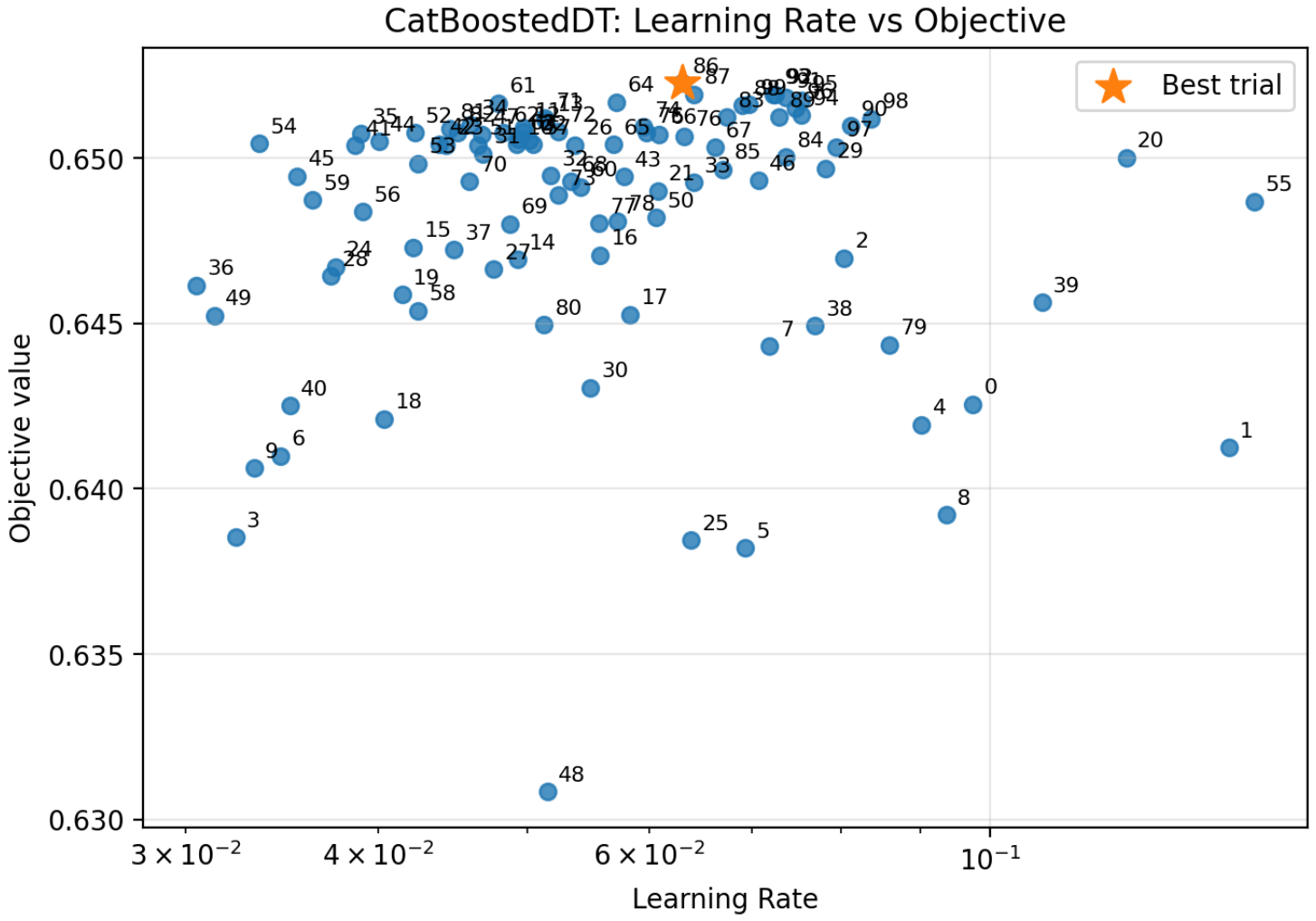
- **1500 iterations:** Smaller ensemble with faster training. May underfit if the model needs more boosting steps to capture complex patterns.
- **2500–4000 iterations:** Intermediate range that balances training time and predictive performance.
- **4543 iterations:** Selected by Optuna as the **best-performing value**, suggesting that the model benefited from many boosting steps.
- **5000 iterations:** Largest ensemble size tested. Can improve performance but increases training time and may require regularization to avoid overfitting



Pre-PID: CatBoost Optimization

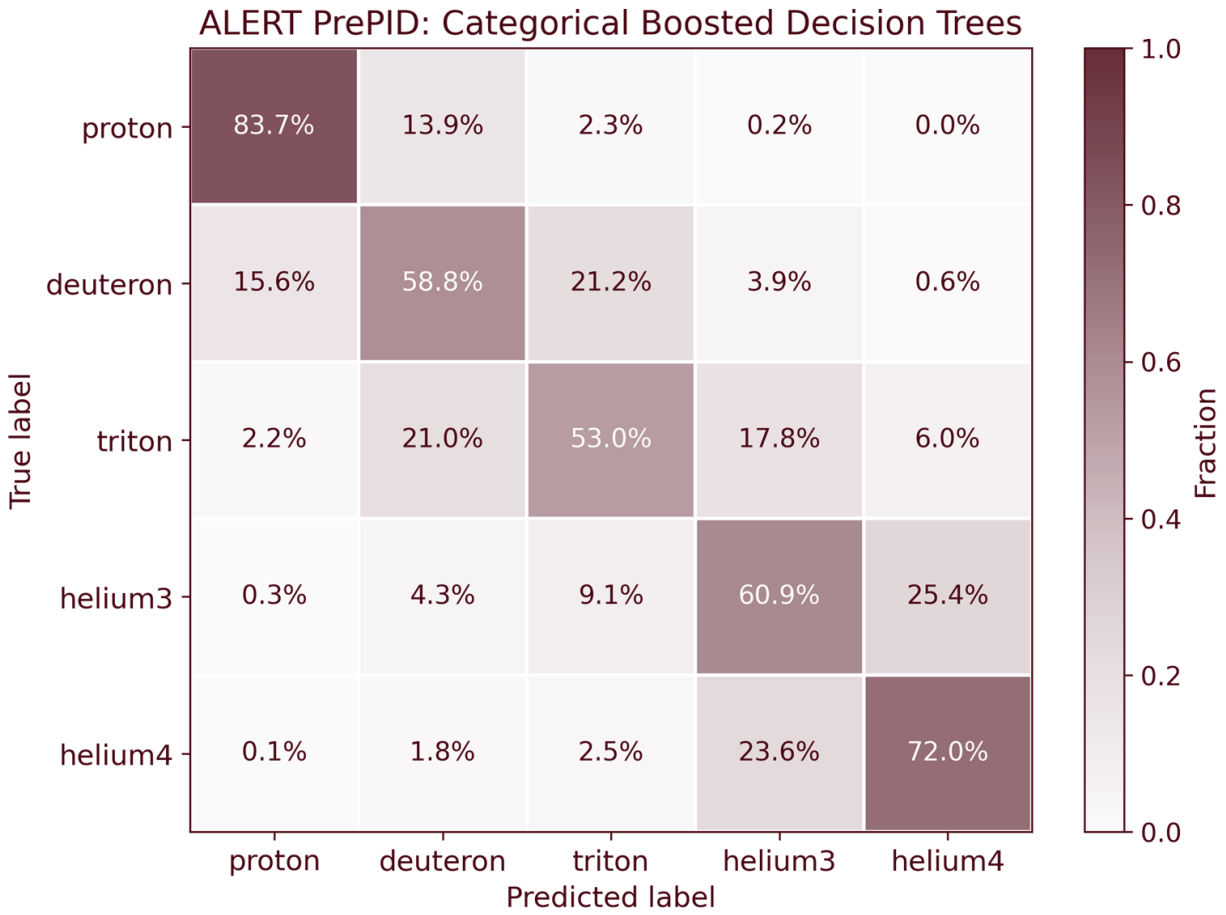
Learning rate values tested

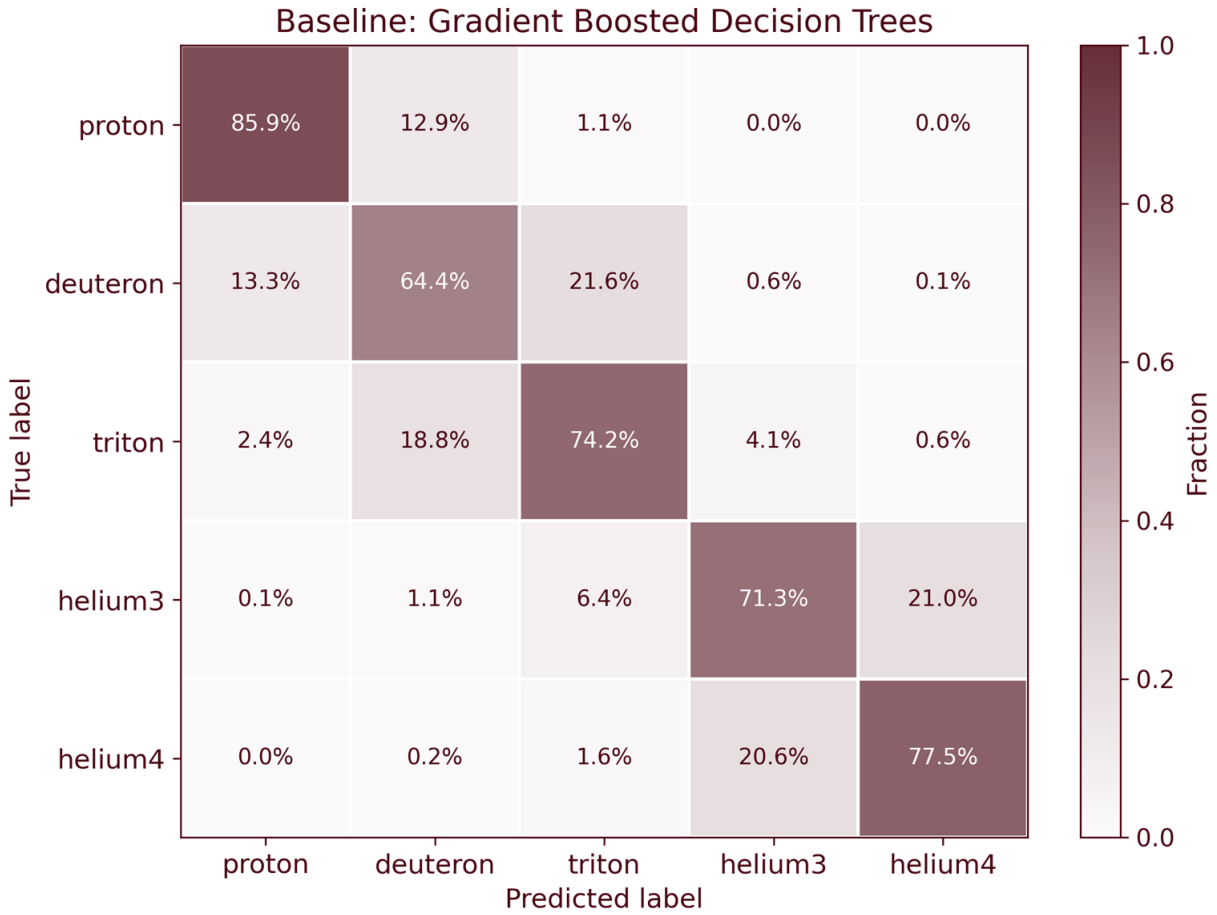
- **0.03:** Smaller learning rate with slower, more gradual updates. Can improve generalization but usually requires more boosting iterations.
- **0.063:** Selected by Optuna as the **best-performing value**, suggesting that moderate updates worked best for this model.
- **0.10:** Intermediate learning rate that can speed up training, but may require stronger regularization.
- **0.15:** Largest learning rate tested. Allows faster updates, but can increase the risk of unstable optimization or reduced generalization if too aggressive.



Post-KF PID Models

- GBDT comparison with pre-KF PID

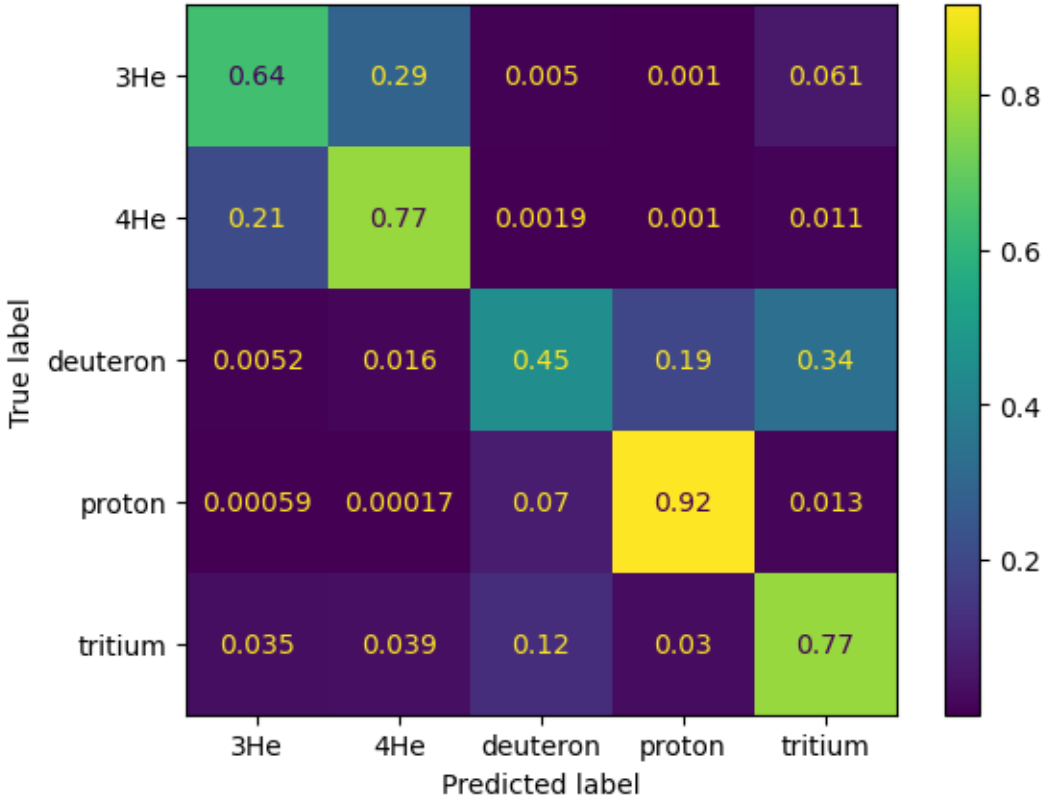




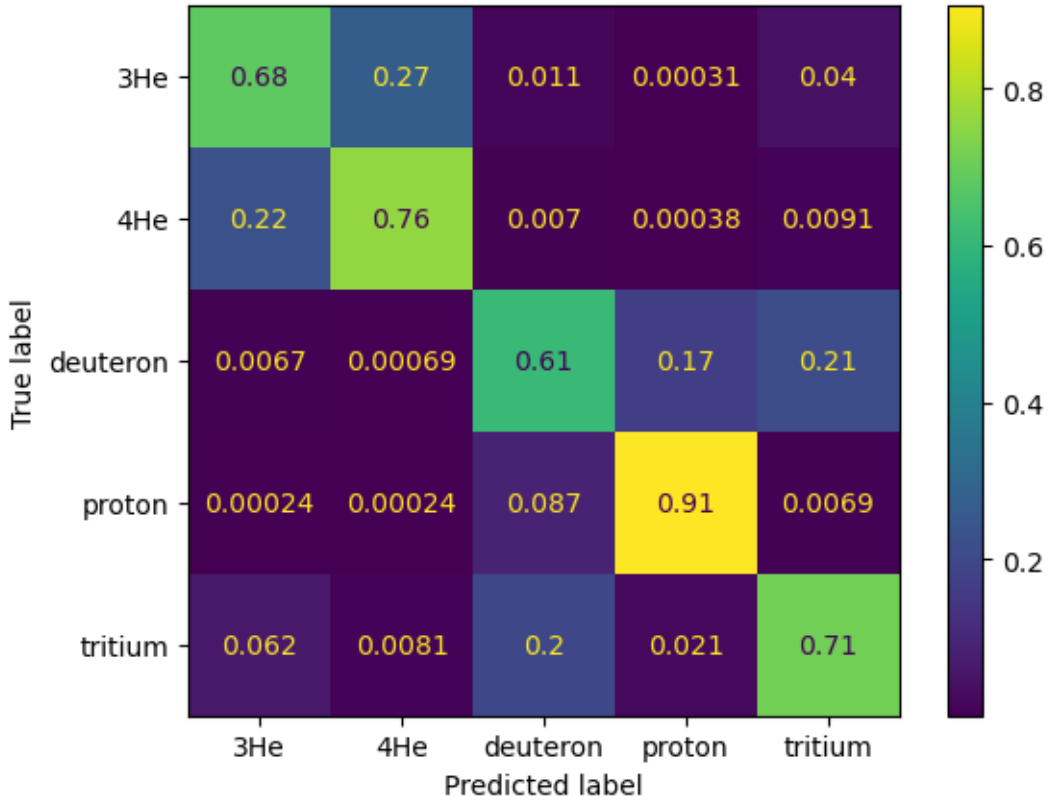
Post-KF PID: Different Approaches --abandoned

- Transforming non-regular variables (e.g., p_x, p_y, p_z) into physically meaningful coordinates (p, θ, ϕ) aligns the inputs with the detector geometry and underlying kinematics
- This method reducing correlations and making patterns easier for the model to learn.

Using p_x, p_y only (dropping p_z)



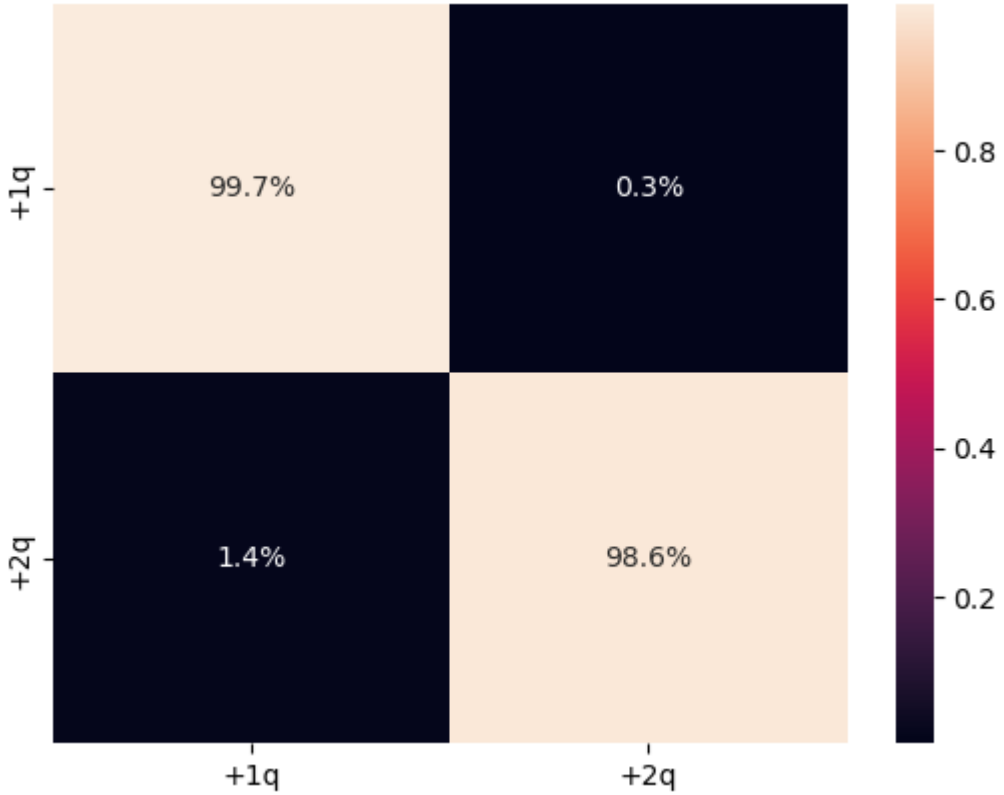
Using P, ϕ instead of p_x, p_y



Post-KF PID: Different Approaches --abandoned

- Based on a suggestion, an attempt was made to use two models to predict charge and mass separately
- Results in increased error and unrealistic predictions (`q=1 & m=4` or `q=2 & m=2`)
- A custom loss-function could make it work, but is not worth the effort

Predict charge only



Predict mass only