LXXV Международная конференция «ЯДРО-2025.
Физика атомного ядра и элементарных частиц.
Ядерно-физические технологии»

Machine-learning-based particle identification

Artem Korobitsin[1], Alexey Aparin[1], Vladimir Popoyan[2]

[1] LHEP JINR, [2] MLIT JINR

Dubna
1-6 July2025

Outline

- Application of Machine Learning (ML) algorithms for particle identification
- ML model: Multi-layer Perceptrons (MLP) and Boosting Decision Trees models
- Data and Feature selection
- Training and testing:
  - ML for PID
  - Comparison with n-sigma method
- Conclusion

Particle identification (PID)

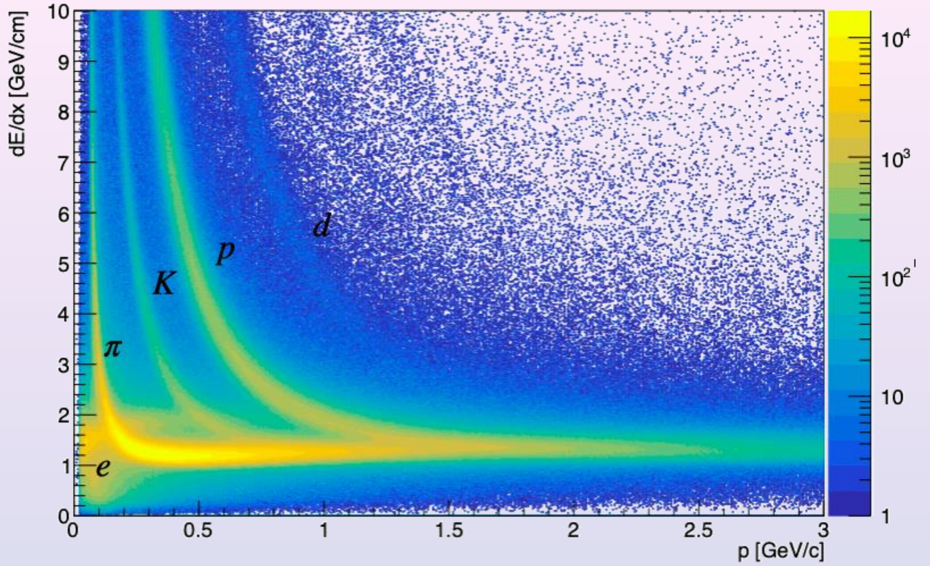
MPD particle identification (PID) is based on Time-Projection Chamber (TPC) and Time-of-Flight (TOF).

Traditional PID (n-sigma method, Bayesian approach):
– a typical analyzer selects particles "manually" by cutting on certain quantities, like the number of standard deviations of a signal from the expected value (nσ)
– most limitations come in the regions where signals from different particle species cross
– "cut" optimization is a timeconsuming task

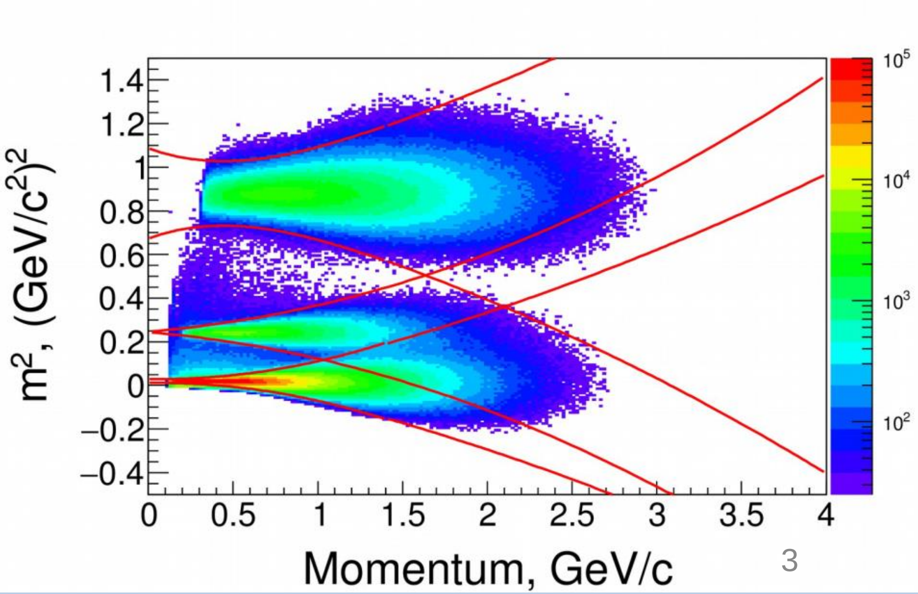
Machine learning PID (ML PID):
- good task for machine learning
- can learn non-trivial relations between different track parameters and PID

Proposed solution for PID:
Build a ML classifier that can outperform traditional PID
Train and validate the classifier on Monte Carlo data

dE/dx vs momentum in TPC



m^2 vs. momentum in TOF



Multi-layer Perceptrons (MLP)

one of the standard method for multi-class and binary classification the evaluation.

TPC

TOF

V_x, V_y, V_z

P

eta

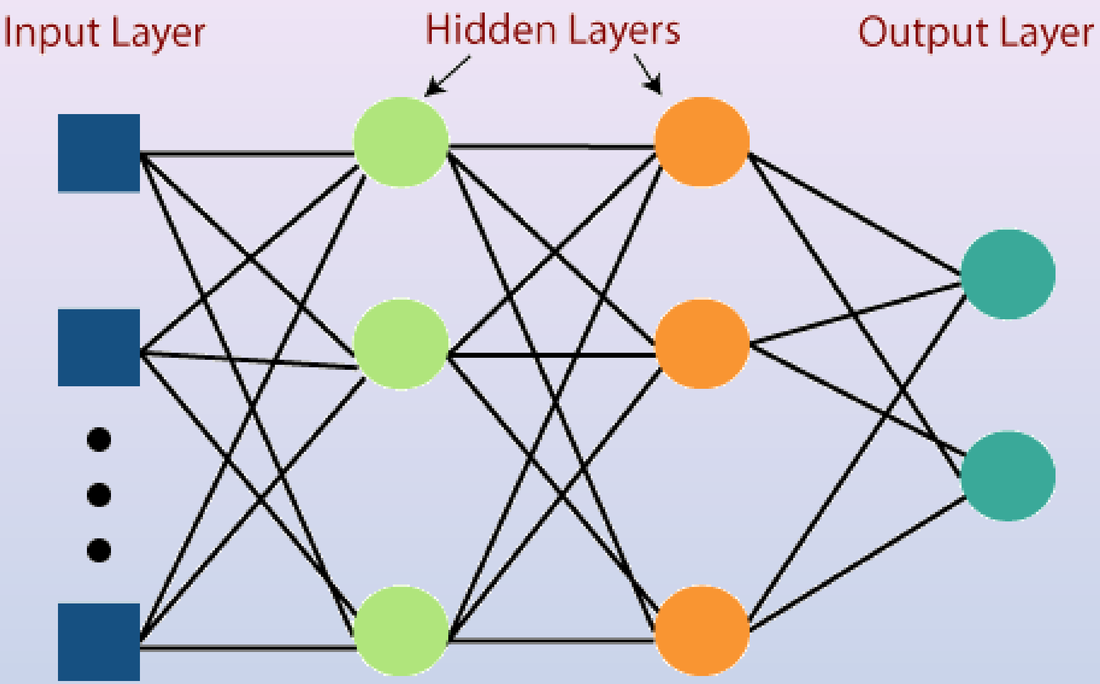
charge

dca

nHits

dE/dx

m^2



π^+

π^-

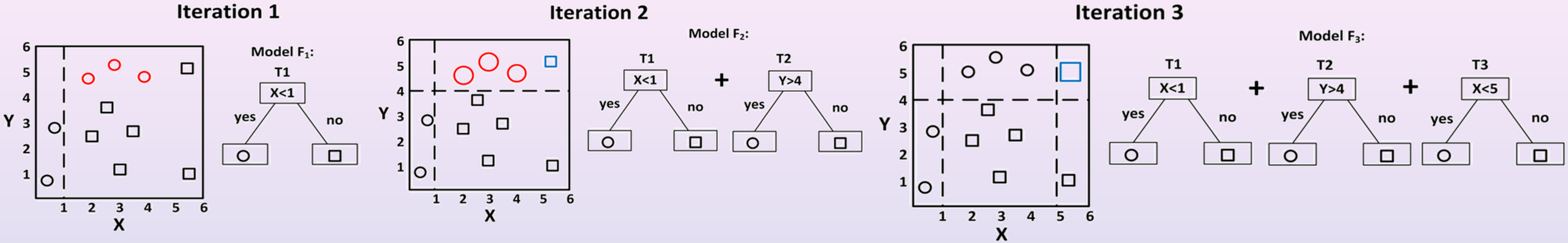
K^+

K^-

p

$\bar{p}$

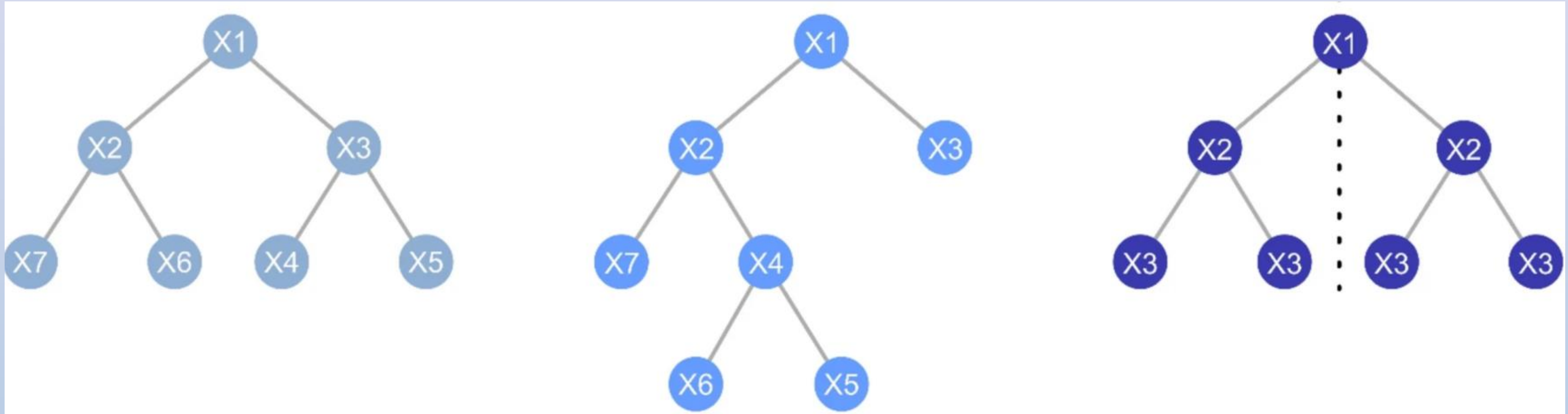
ML Model: Gradient Boosting Decision Trees (GBDT) is a machine learning algorithm that uses gradient boosting on decision trees. At each iteration, trees are added in such a way that the value of the objective function decreases.



Asymmetric Tree: XGBoost , LightGBM

Symmetric Tree : CatBoost , SketchBoost

XGBoost ***LightGBM*** ***CatBoost***



https://www.researchgate.net/figure/A-simple-example-of-visualizing-gradient-boosting_fig5_326379229

Datasets:	prod01	prod04	prod05 (Request 25)	prod06 (Request 29)
Event generator	UrQMD + BOX	UrQMD + BOX	UrQMD	PHQMD
Transport	Geant 4	Geant 4	Geant 4	Geant 4
Impact parameter ranges	0- 16 fm (mb)	0- 16 fm (mb)	0- 16 fm (mb)	0- 12 fm
SmearVertexXY	0.1 cm	1.1 cm	0.1 cm	0.1 cm
SmearVertexZ	24 cm	50 cm	50 cm	50 cm
Colliding system	Bi (83/209) +Bi (83/209)			
Energy	9.2 GeV			

Track selection criteria:

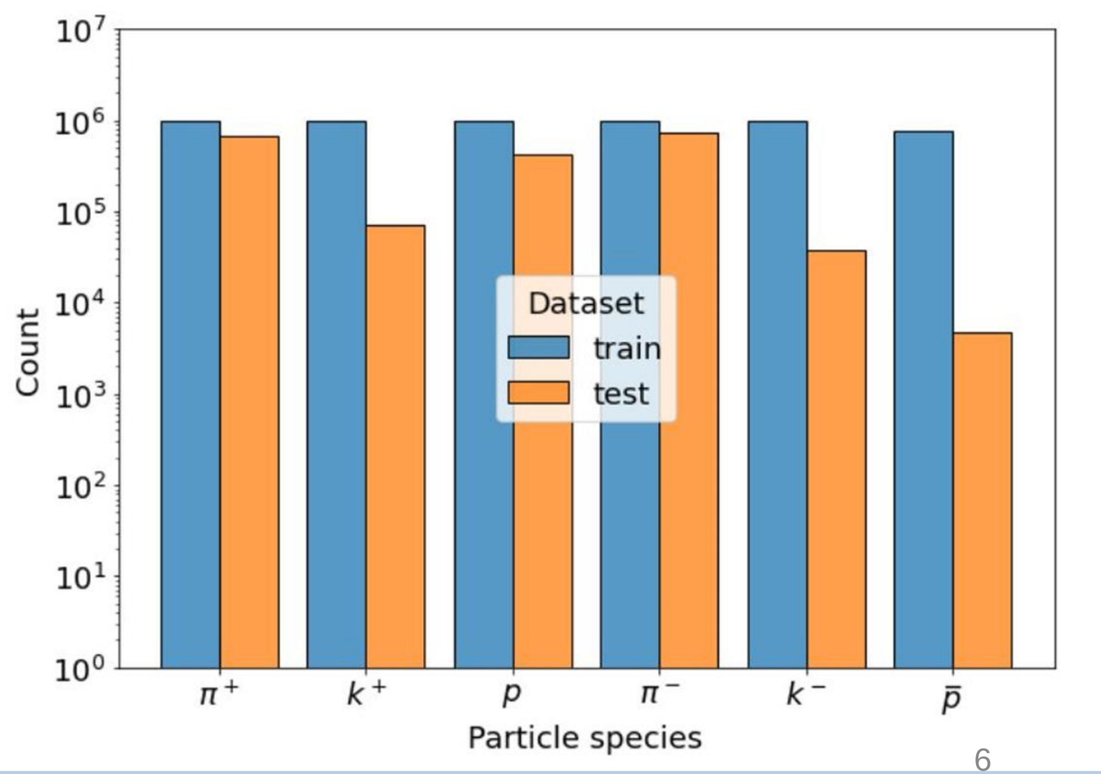
$p < 100$ GeV/c, $|m^2| < 100\ \left(\mathrm{GeV}/c^2\right)^2$,

nHits > 15, |eta|<1.5, dca < 5 cm, |Vz| < 100 cm

Training and validation dataset:

One million elements (tracks) for each of the six classes (particles): $\pi^+, \pi^-, K^+, K^-, p, \bar{p}$

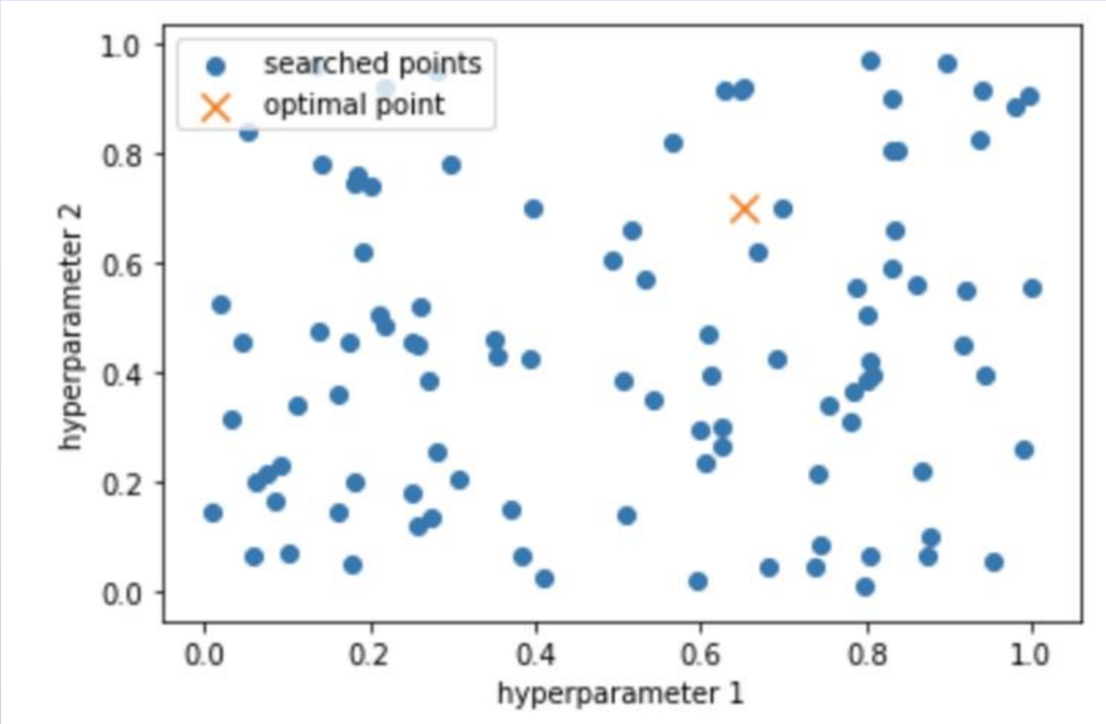
Testing dataset: 50000 events.



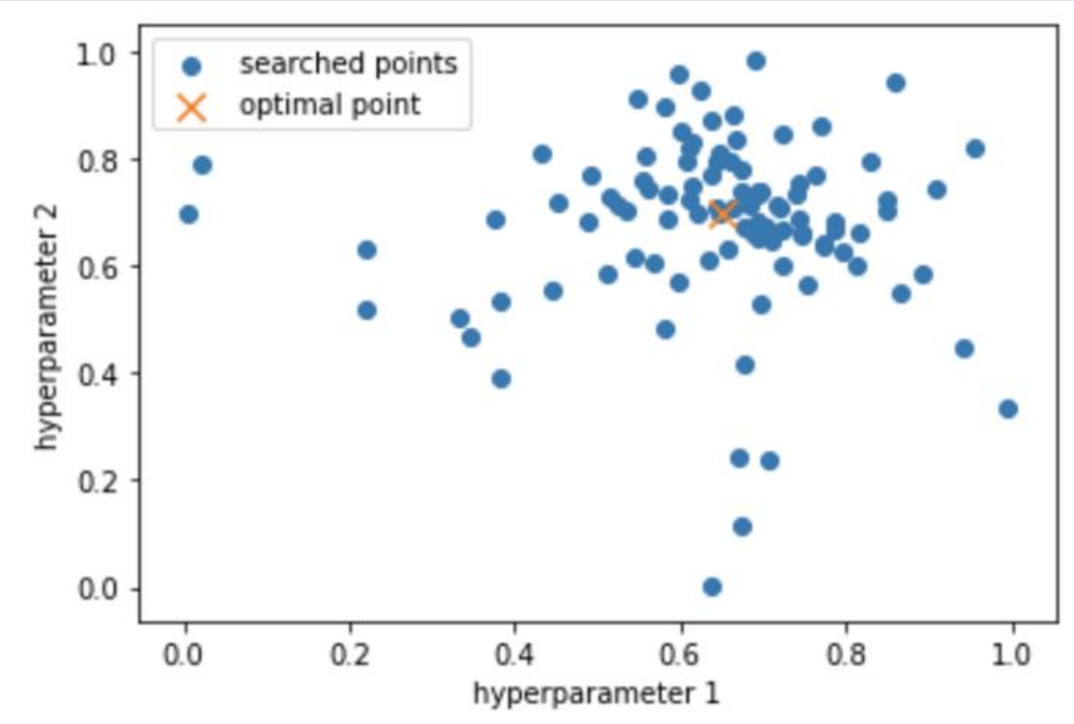
Hyperparameters selection (Select optimal hyperparameters of ML model)

Four commonly used optimization strategies: Grid search, Random search, Hill climbing and Bayesian optimization.
Tree-structured Parzen Estimator (TPE) was used to find the optimal hyperparameters; TPE is a form of Bayesian Optimization

Random search



TPE search



https://miro.medium.com/max/4800/1*tYWqO5BwNDVaM3kP3w1IAg.png

Correlation matrix for all input feature

	P	charge	dE/dx	m^2	nHits	Eta	dca	V_x	V_y	V_z
P	1	-0.01	-0.01	0	-0	-0	0	-0	0	0
charge	-0.01	1	0.1	-0	-0.03	-0	-0	0	-0	0
dE/dx	-0.01	0.1	1	-0	0.01	0	0.12	0	-0	0
m^2	0	-0	-0	1	-0.01	0	0	-0	0	0
nHits	-0	-0.03	0.01	-0.01	1	0.03	-0.21	0	0	0
Eta	-0	-0	0	0	0.03	1	-0	-0	-0	-0.08
dca	0	-0	0.12	0	-0.21	-0	1	-0.01	0	-0
V_x	-0	0	0	-0	0	-0	-0.01	1	-0.01	-0
V_y	0	-0	-0	0	0	-0	0	-0.01	1	-0.01
V_z	0	0	0	0	0	-0.08	-0	-0	-0.01	1

Feature selection

prod 01:

	Feature Id	Importances
0	charge	48.976478
1	p	15.612522
2	m2	13.219858
3	dedx	12.504383
4	dca	2.931781
5	nHits	2.682914
6	eta	1.732293
7	Vz	0.904500
8	Vx	0.757425
9	Vy	0.677845

prod 04:

	Feature Id	Importances
0	charge	52.595520
1	p	16.143578
2	m2	11.179546
3	dedx	9.959441
4	eta	3.202594
5	dca	3.178775
6	nHits	2.890517
7	Vy	0.322261
8	Vx	0.293670
9	Vz	0.234098

prod 05:

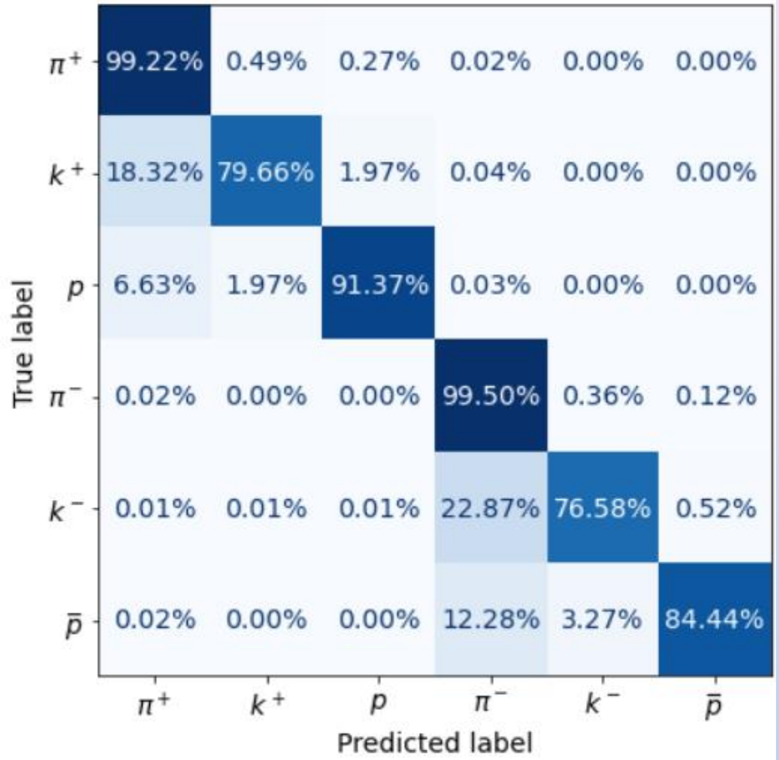
	Feature Id	Importances
0	charge	43.753433
1	p	19.143319
2	dedx	18.371532
3	m2	9.106441
4	dca	3.549774
5	nHits	2.178229
6	eta	1.912249
7	Vz	0.802412
8	Vx	0.630954
9	Vy	0.551657

The bigger the value of the importance the bigger on average is the change to the predicted value, of this feature is changed.

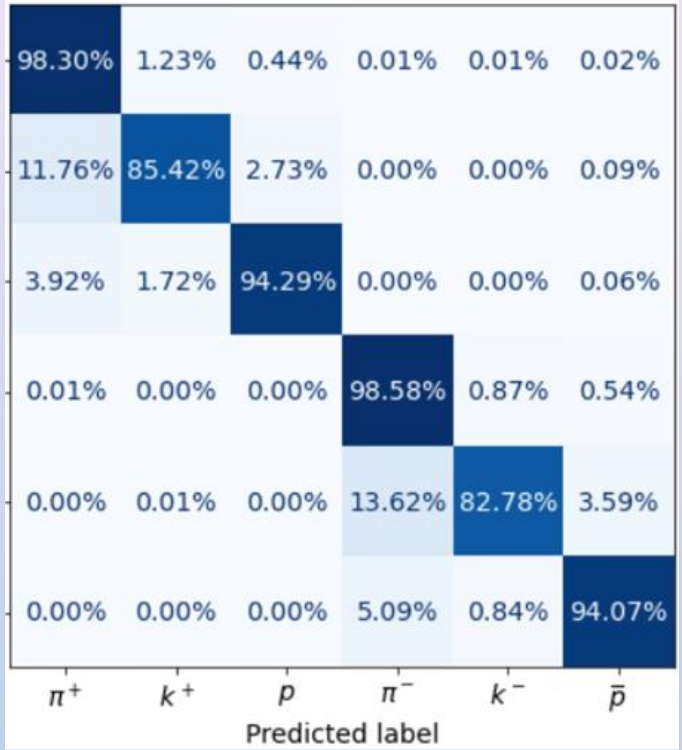
Confusion matrix for the six classes of model

Each column of matrix – predicted value, each row of matrix – true value.

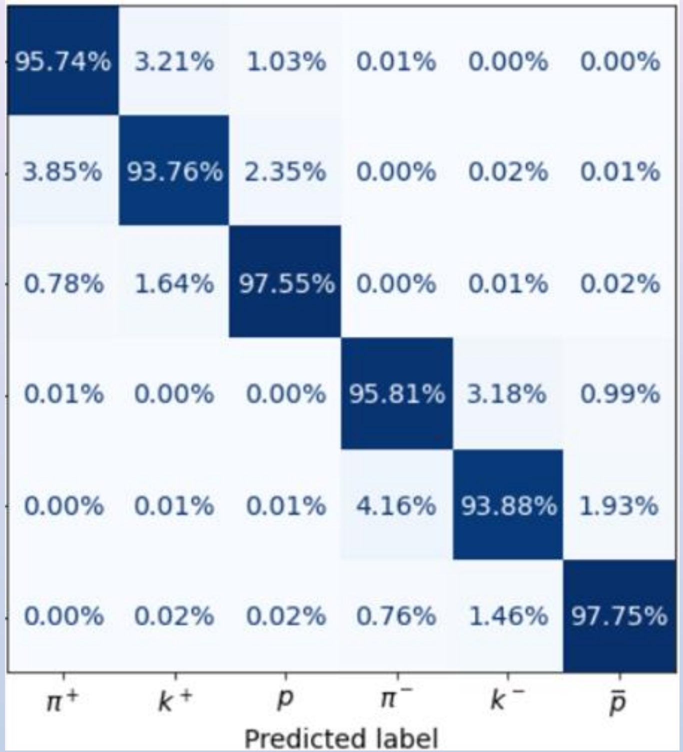
prod 01:



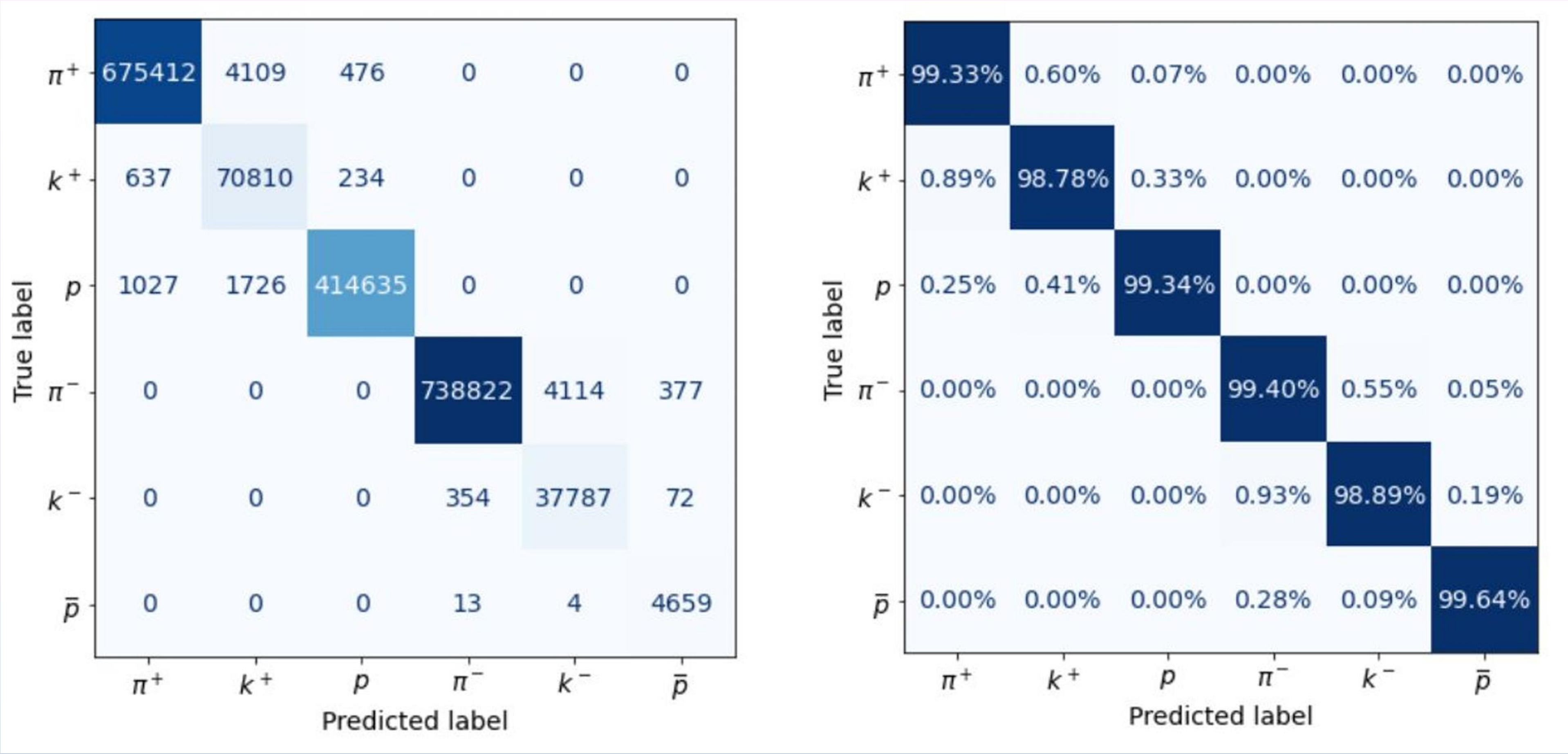
prod 04:



prod 05:

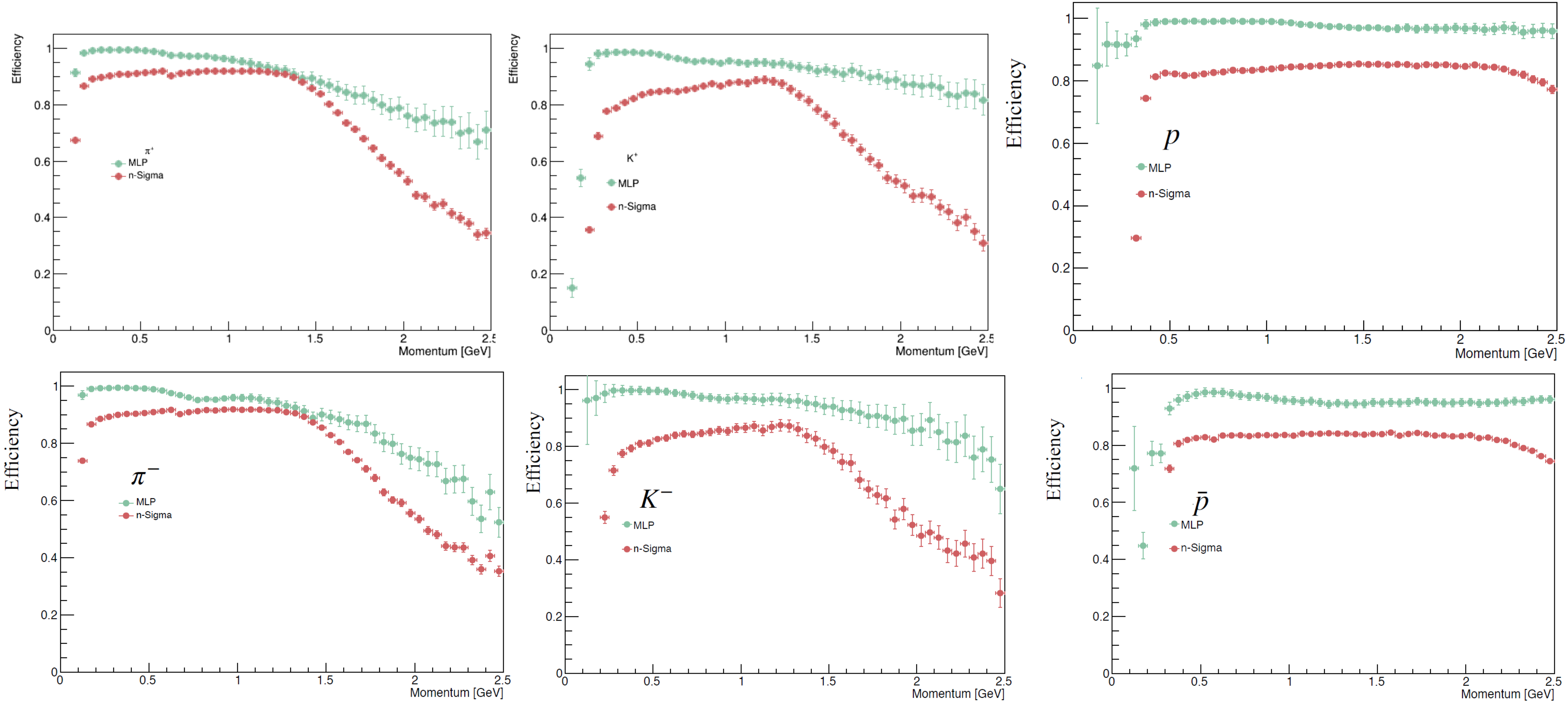


Confusion matrix for the six classes of model



Comparison MLP with n-sigma method

$$Efficiency = \frac{right\ identified\ tracks}{all\ tracks}$$



Multilayer Perceptron Neural Model for Particle Identification in MPD, Phys. Atom. Nucl. 86 (2023) no.5, 845-849

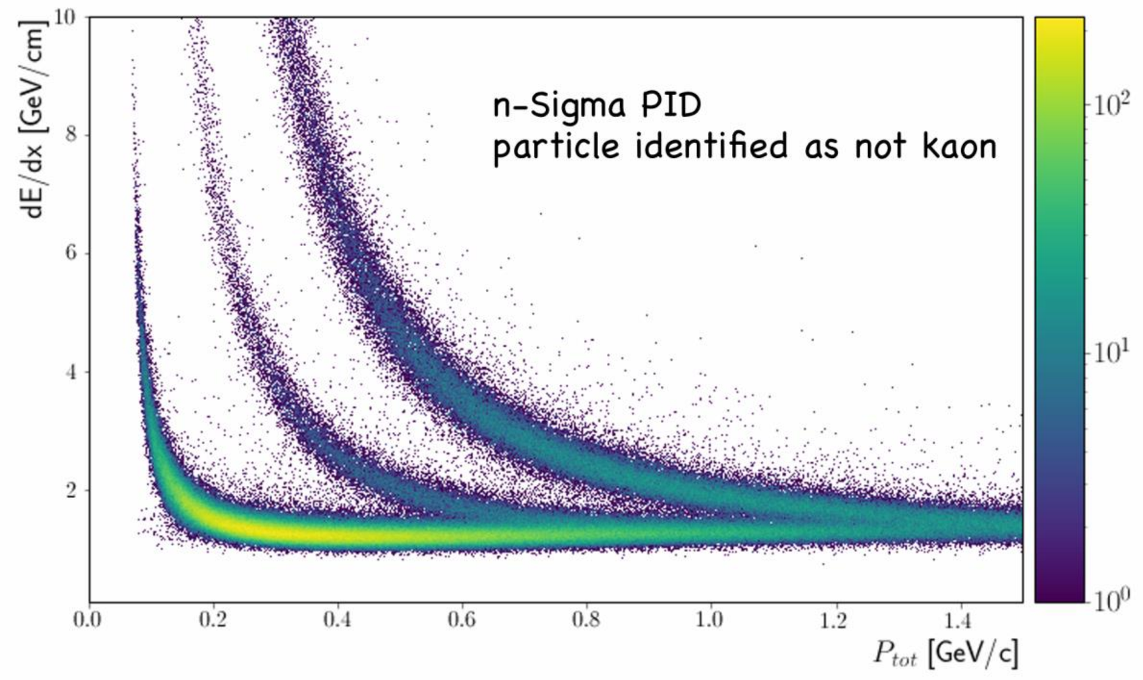
Why does ML approach have better efficiency for each particle species, but it has the same or higher contamination than n-Sigma approach?

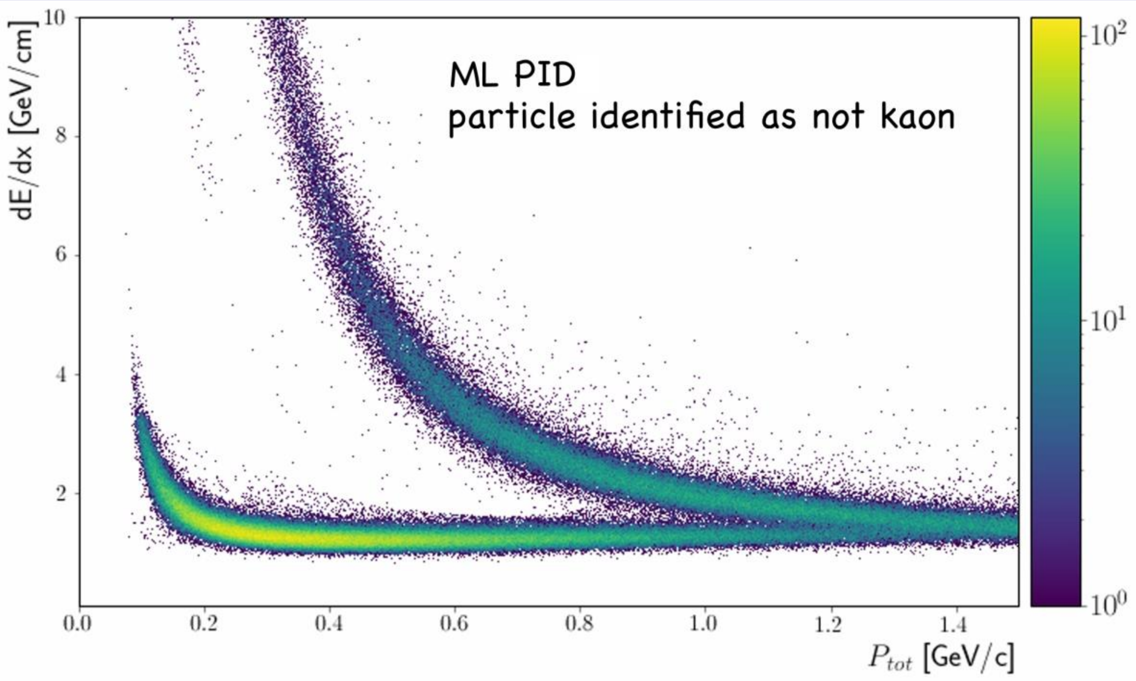
n-Sigma approach identifies particle as particle of a i-species if $N_\sigma \leq \sqrt{N^2_{\sigma^i_{TOF}} + N^2_{\sigma^i_{TPC}}}$

values are in a certain range around mean value for i-species of particle. Where

$$N_{\sigma^i_{TPC}} = \frac{dE/dx - \langle dE/dx \rangle^i}{\sigma^i_{TPC}}, N_{\sigma^i_{TOF}} = \frac{m^2 - \langle m^2 \rangle^i}{\sigma^i_{m^2}}$$

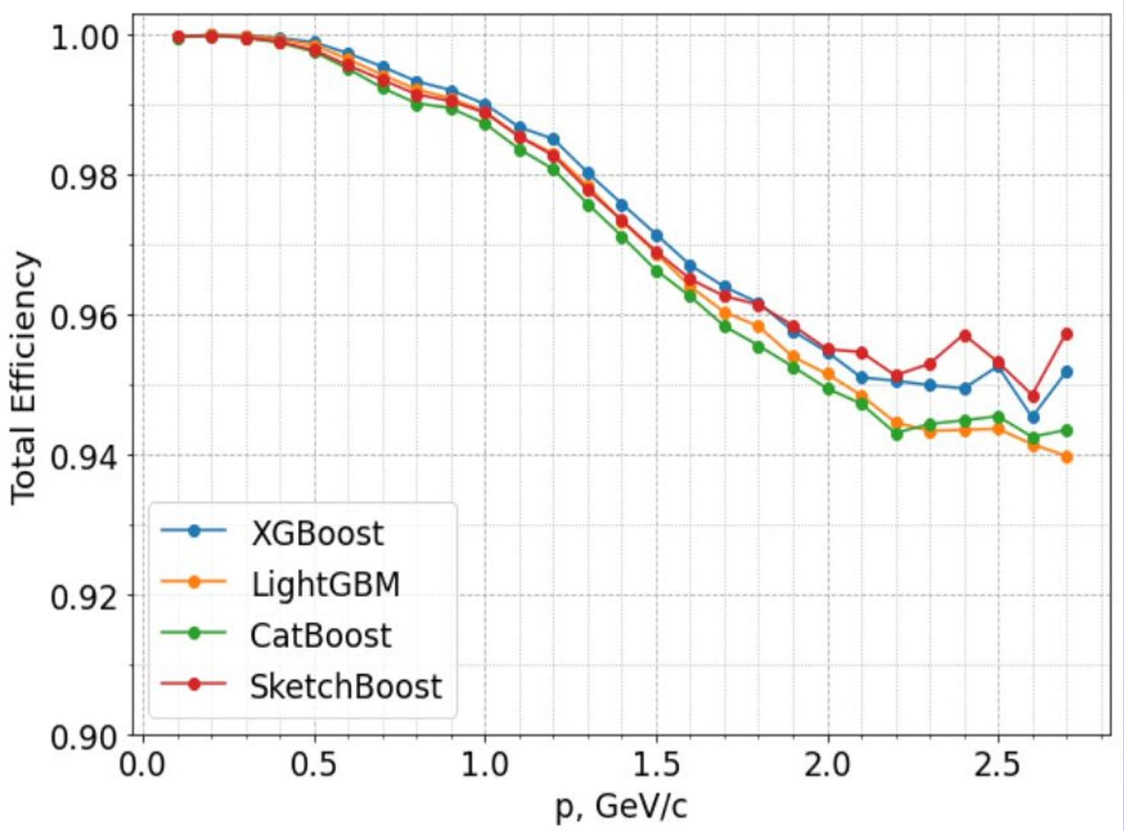
If a particle can be compatible with more than one species, n-Sigma approach does not identify this particle.

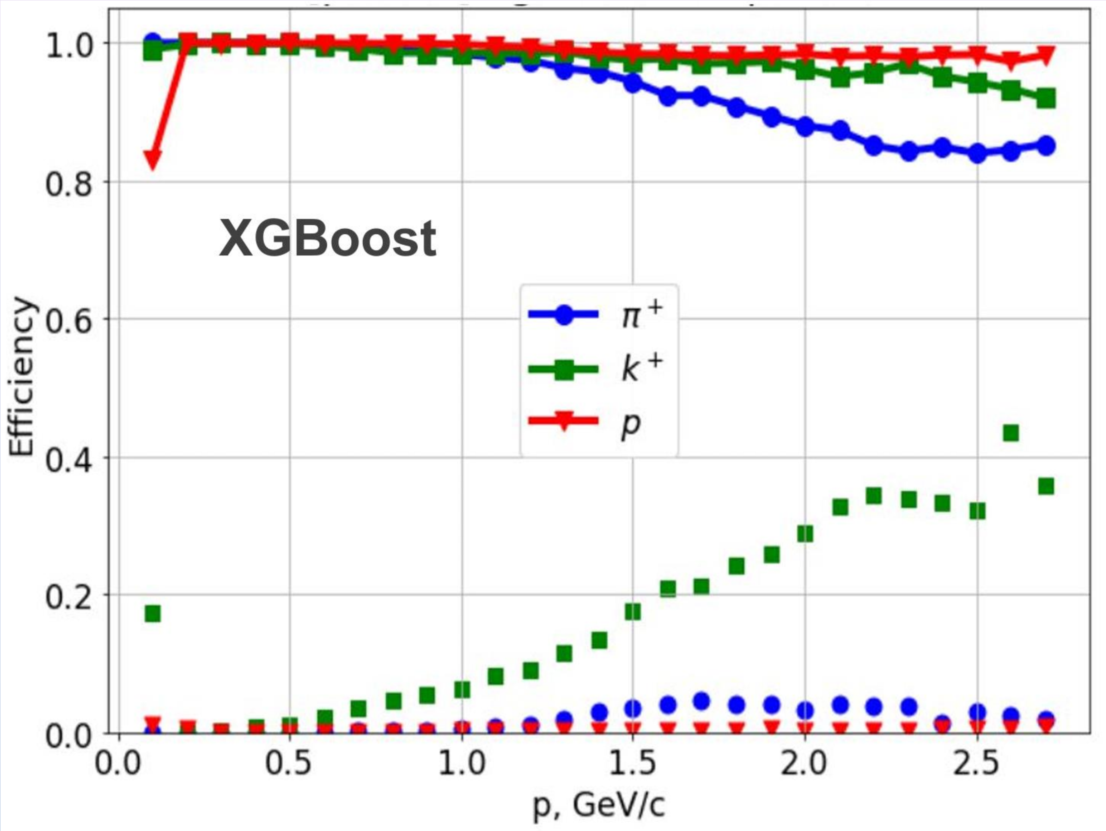




Comparative analysis of the algorithms. Efficiency

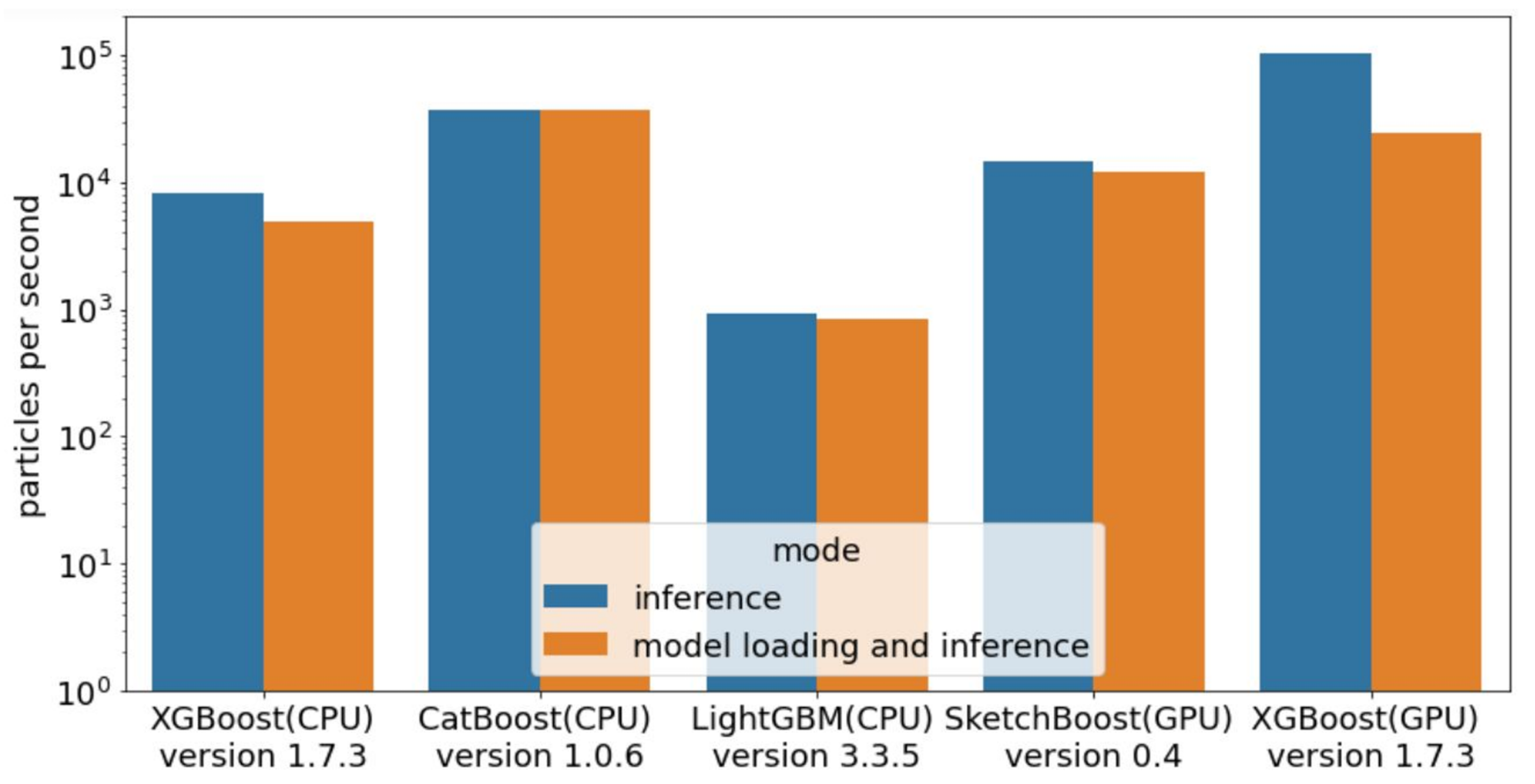
	XGBoost	LightGBM	CatBoost	SketchBoost
Total Efficiency	0.99327	0.99235	0.99138	0.99239





Machine Learning Application for Particle Identification in MPD, Phys. Atom. Nucl. 86 (2023) no.5, 869-873
Gradient Boosted Decision Tree for Particle Identification Problem at MPD Phys. Part. Nucl. Letters (2025)

Comparative analysis of the algorithms. Inference time



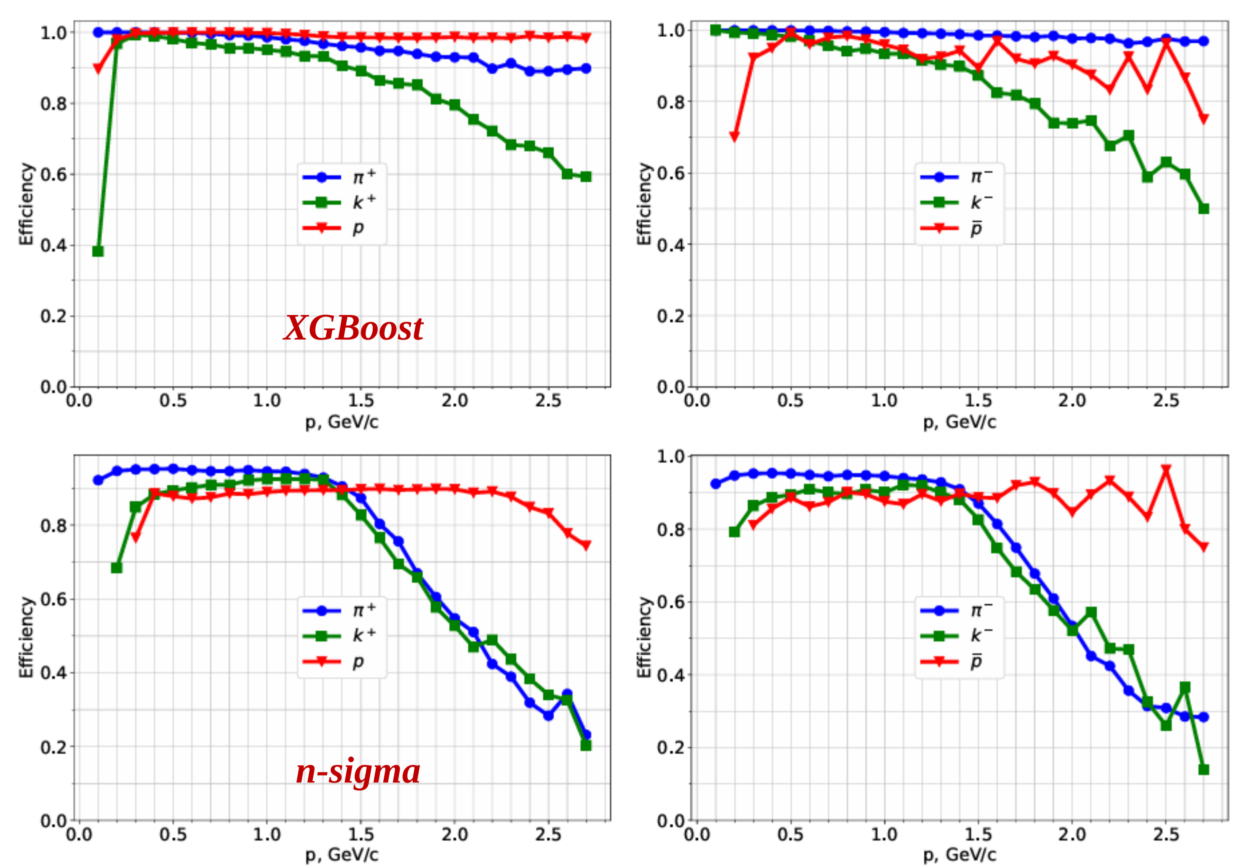
GPU: Nvidia Tesla V100-SXM2 NVLink 32GB HBM2

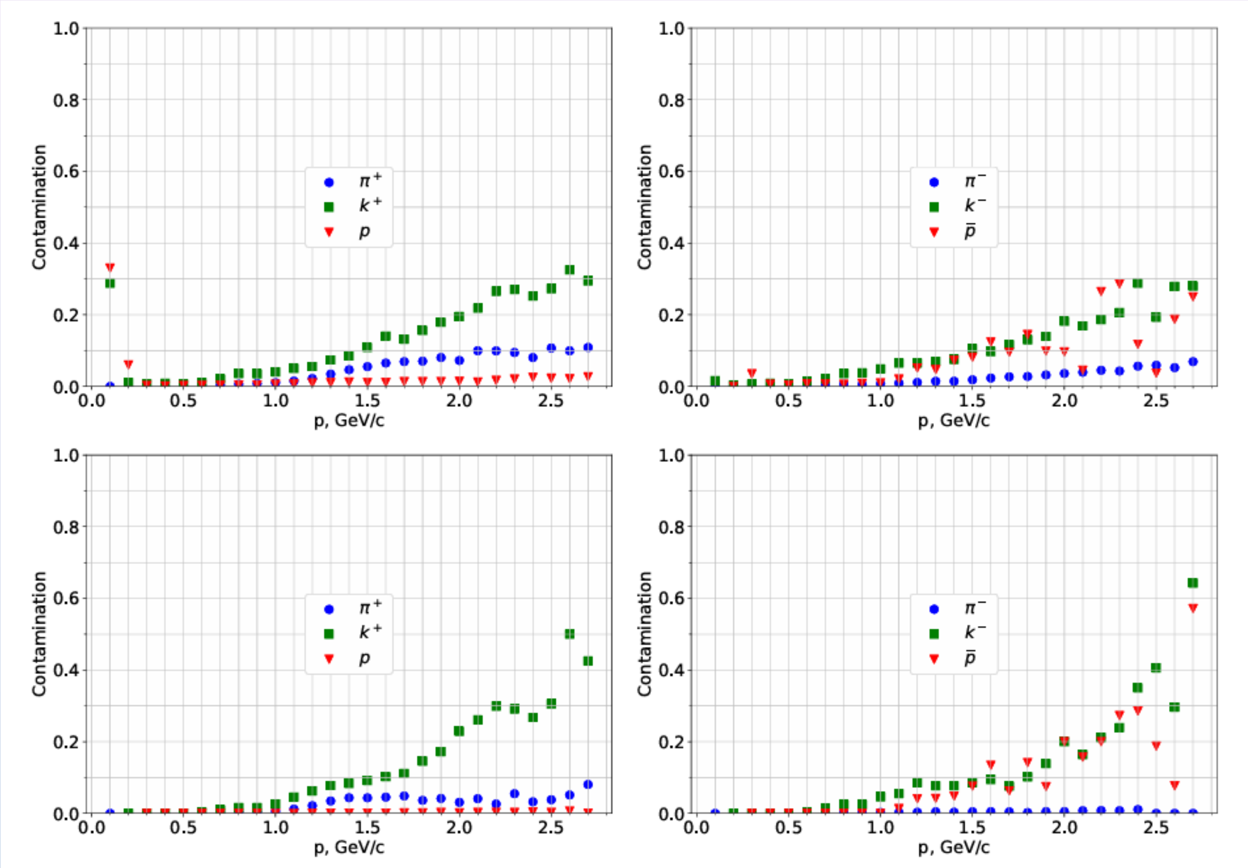
CPU: Intel Xeon Gold 6148 CPU @ 2.40 GHz 20 Cores / 40 Threads

Comparison XGBoost with n-sigma method

$$Efficiency = \frac{right\ identified\ tracks}{all\ tracks}$$

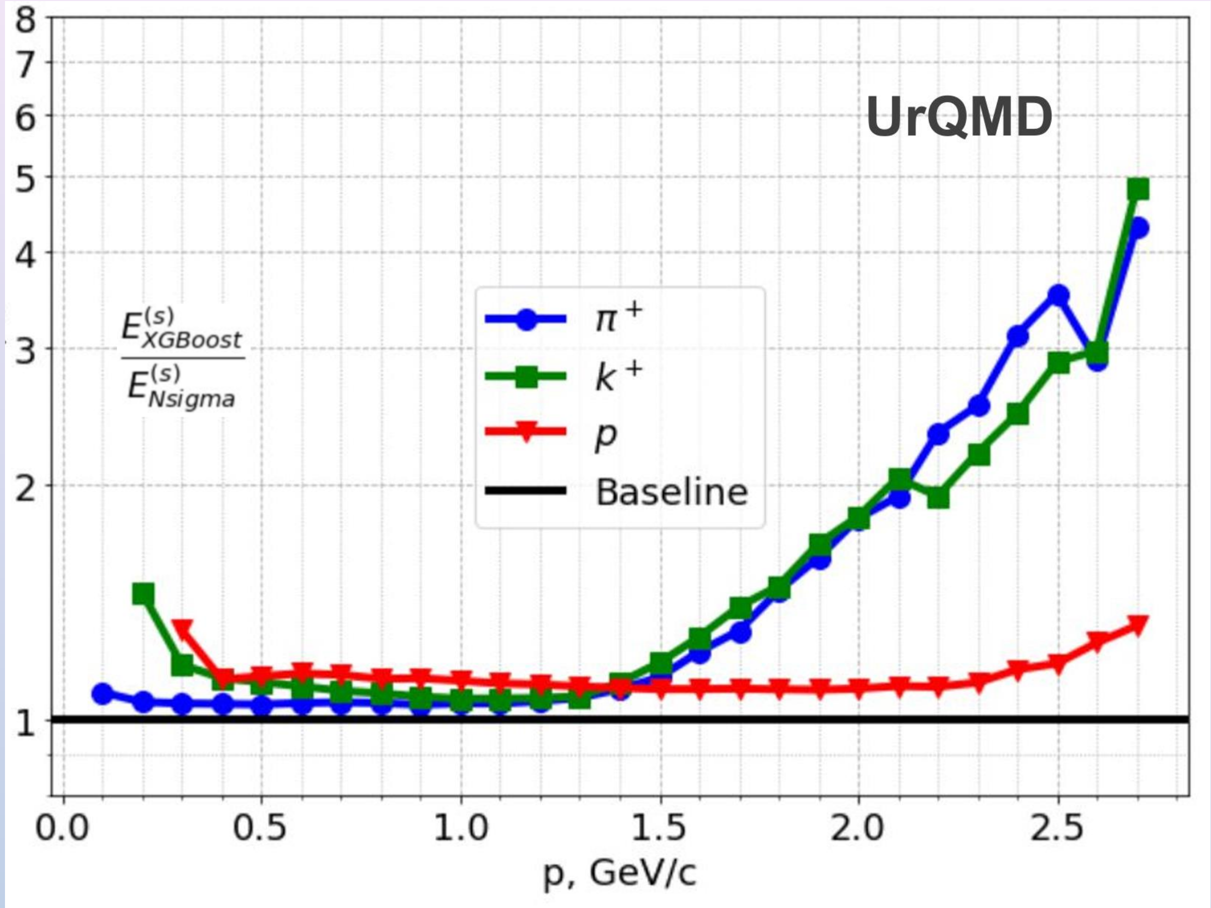
$$Contamination = \frac{wrong\ identified\ tracks}{identified\ tracks}$$

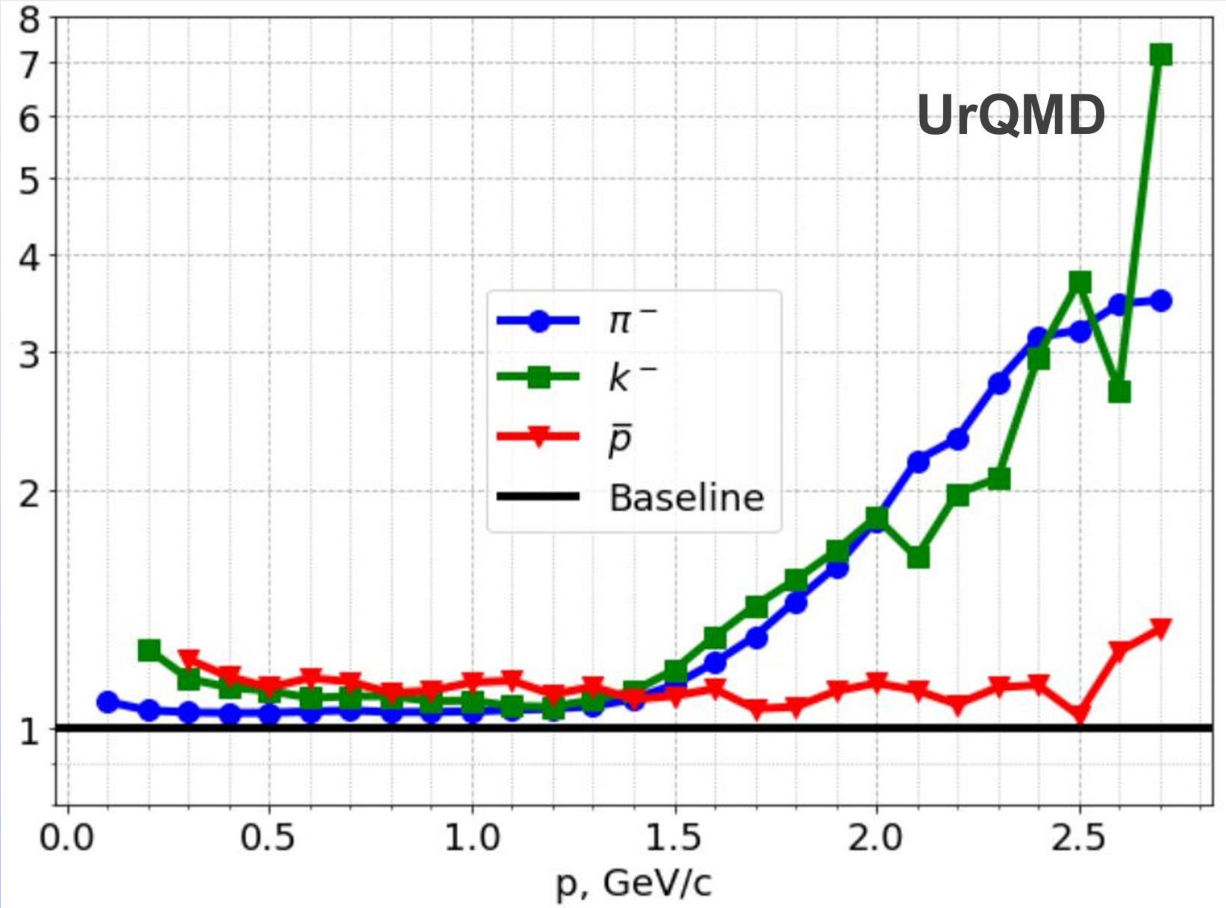




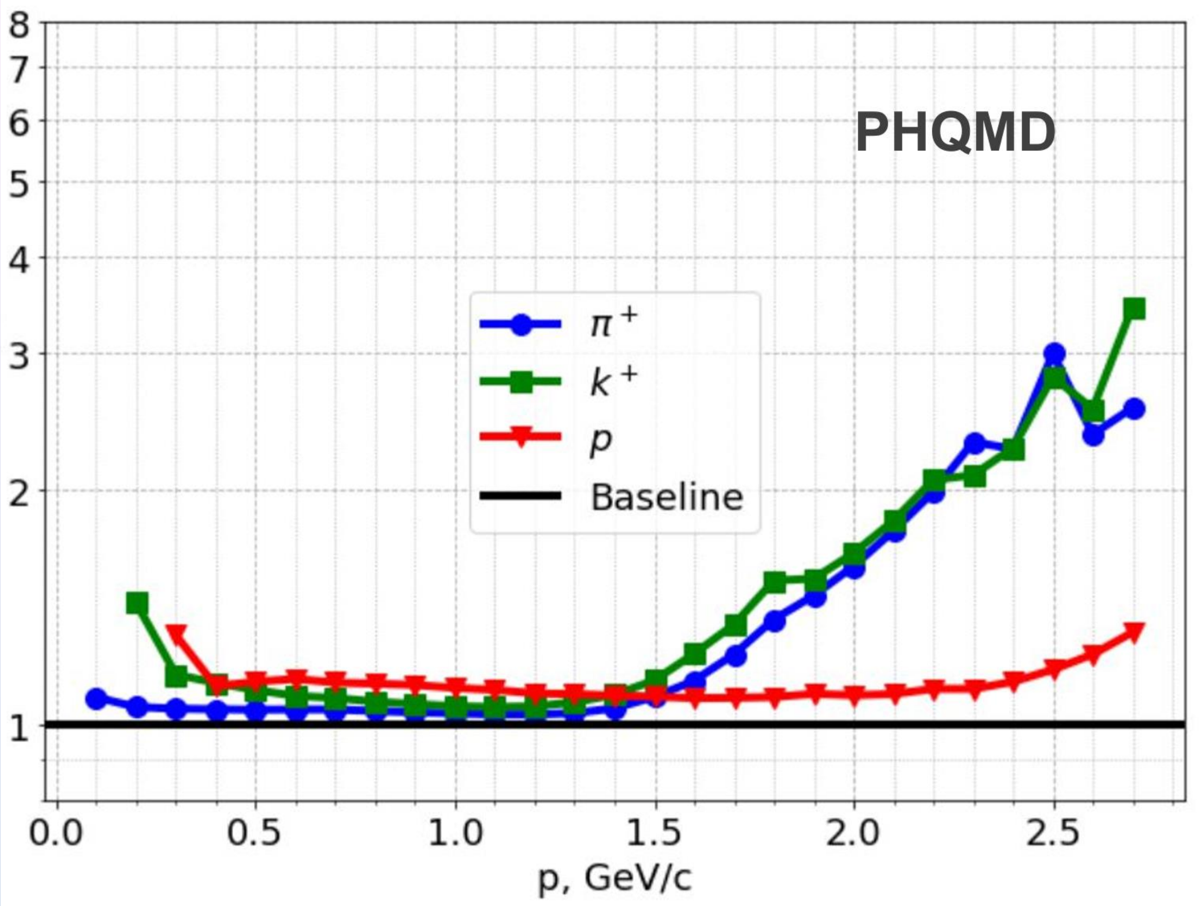
Momentum Particle Identification in the MPD Experiment Phys. Part. Nucl. (2025)

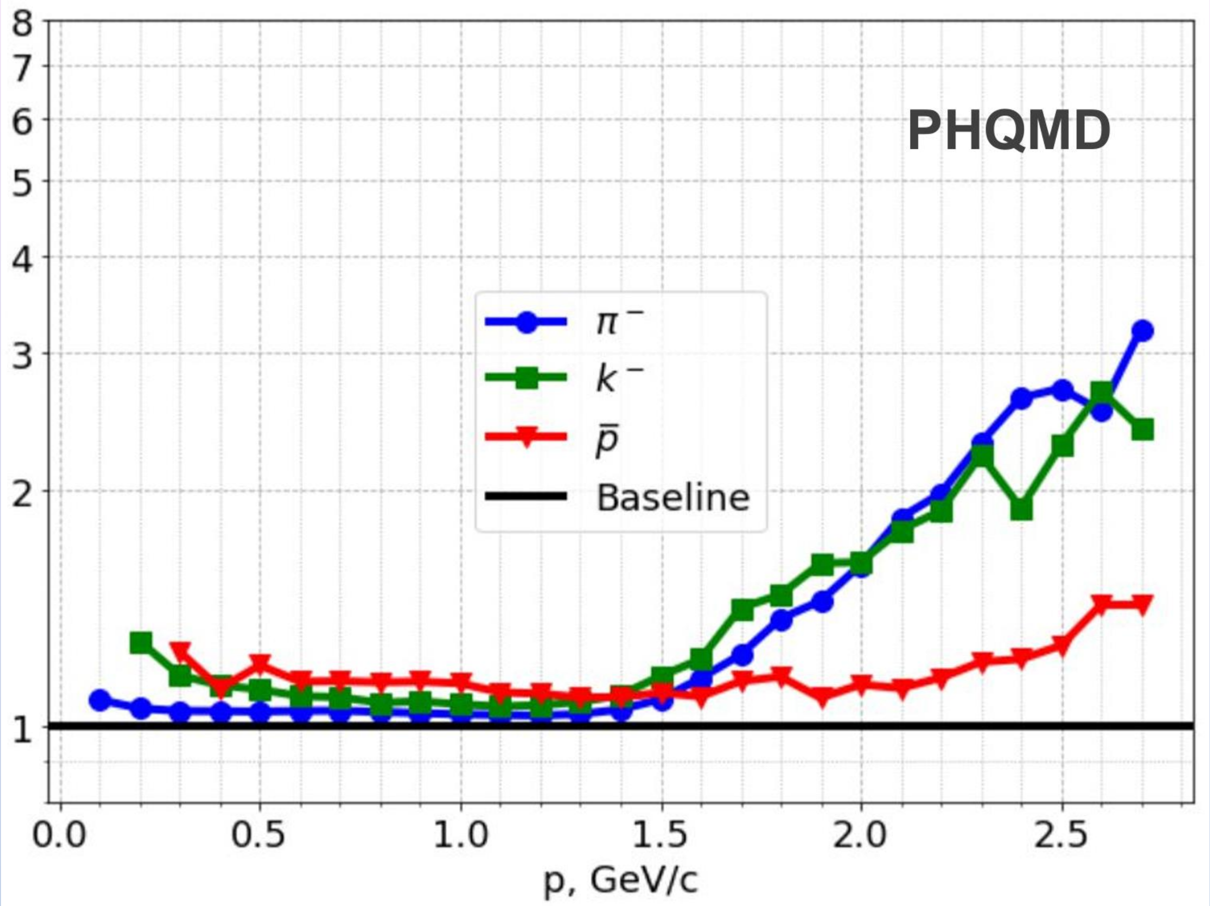
Efficiency ratio of XGBoost and n-sigma method





Efficiency ratio of XGBoost and n-sigma method





MPDROOT: MpdPidMl

```
#include <xgboost/c_api.h>
#include "MpdPidMl.h“

MpdPidMl pid_ml; // default model
// MpdPidMl pid_ml(“name_model.ubj");

pid_ml.fillData(variables); // variables: full momentum, eta, dE/dx, mass squared, charge, Vz, nHits
pdgCode = pid_ml.GetMaxProb;
```

Conclusion

ML-based PID outperforms traditional PID, especially in the low and high momentum region.

Training needed only once for each data set – no need for manual cut optimizations.

Shown improvement for a wide datasets of MC simulation data.

Thank you for your attention

This work was supported by RSCF under grant N 22-72-10028