INFN
Istituto Nazionale di Fisica Nucleare

ICSC
Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

Update on *“Quality control (QC) of primary vertices based on reconstruction properties”* with ML

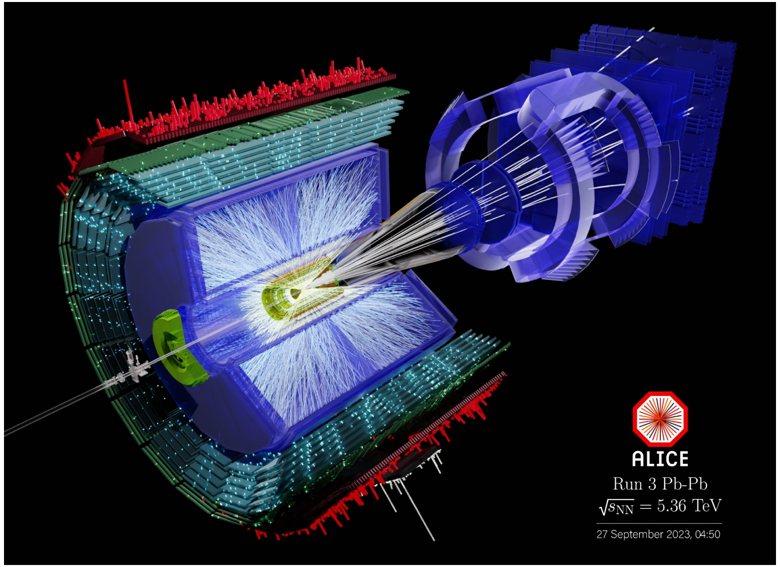
Mattia Faggin, University and INFN Trieste

Bi-weekly WP2 meeting
24th October 2023

Introduction

Motivations

- Run 3 at the LHC: ALICE taking data in continuous readout mode, i.e. trigger-less data
- Signals of different collisions overlap within the ~100us drift time of the Time Projection Chamber (5 Pb-Pb collisions at the max. interaction rate of 50kHz)
- Correct data-to-collision association (space, ***time***) is not known a priori, and multiple primary-vertex findings must be executed within every acquisition time frame



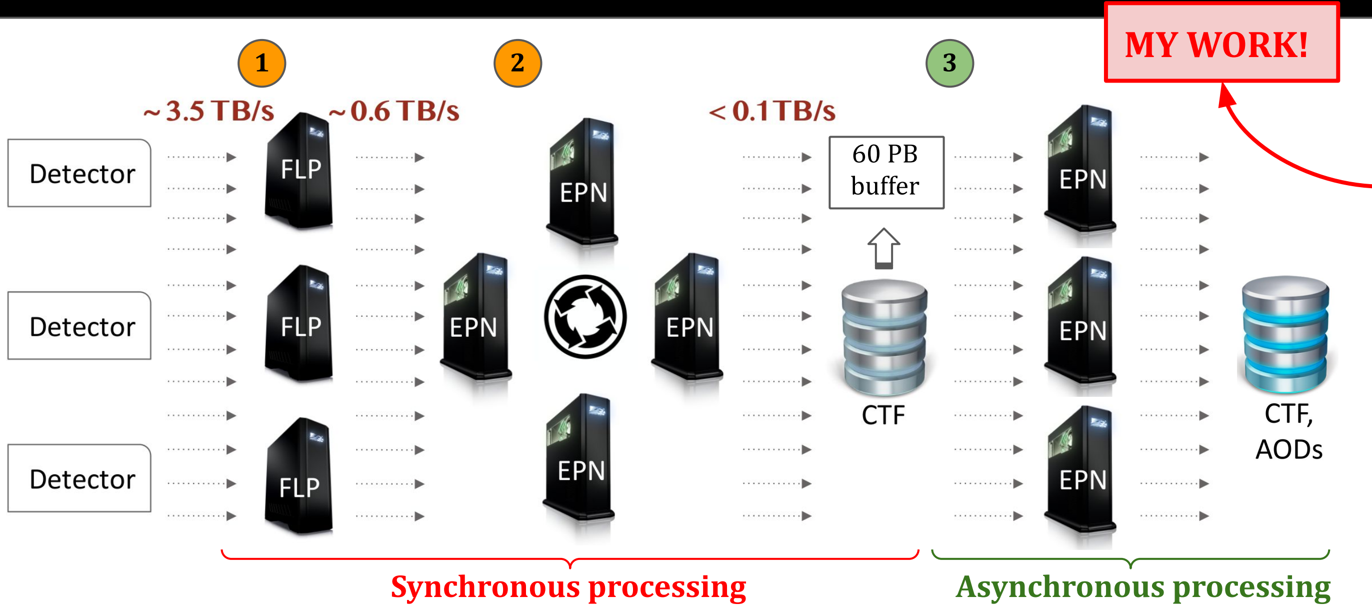
This work

Develop a tool to tag the duplicated primary vertices (PV) based on the PV properties from the detectors and data reconstruction

Binary classification: single vs. duplicated PVs

GitHub repository: https://github.com/mfaggin/monitorPvML_hpc

Data reconstruction in ALICE



3 **Asynchronous processing**

- New reconstruction with final calibrations on EPN, T0 and T1
- **Final Analysis Object Data (AO2Ds)** produced and stored
- ~~CTFs cancelled to free disk space~~

- still a flexible condition, due to imperfect data reconstruction
- original plan: CTF removal after ~1 month from data taking, much slower now

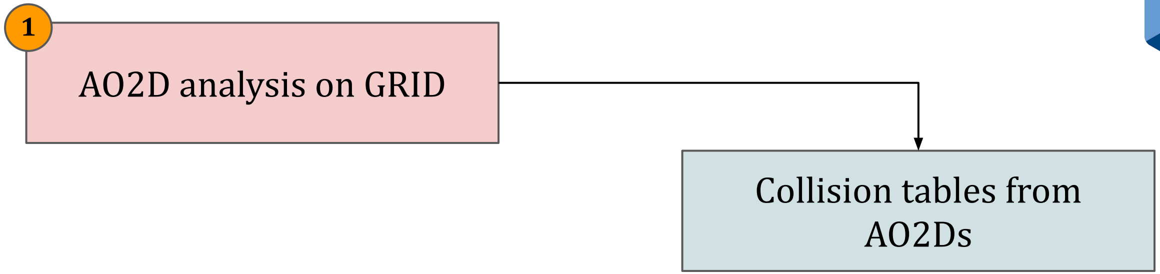
1 **First Level Processors (FLP)**

- First compression (*zero suppression*) of data from detector readout links
- Data division in sub-TFs on each FLP

2 **Event Processing Nodes (EPN)**

- Sub-TF merge in complete TFs
  - 1 TF = 11 ms in 2022 (128 orbs), 2.8 ms in 2023 (32 orbs)
- Synchronous reconstruction, calibration, data compression
- Compressed TFs (CTFs) buffer

Structure of the work

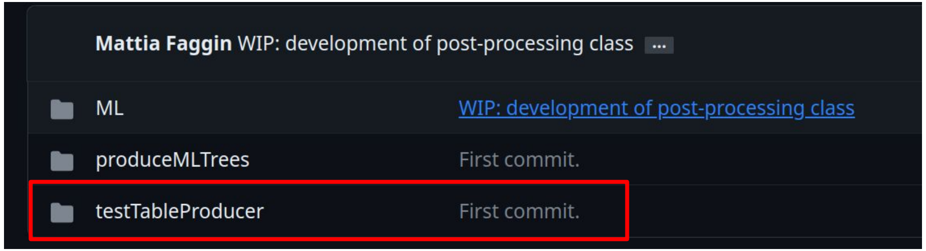


- AO2D analysis done with O^2/O^2Physics workflows
- Parallelized execution on the GRID with the new Hyperloop system

Just for local testing. Full O2/O2Physics framework needed.

- O2/Physics repository: [link]
- Build instructions: [link]

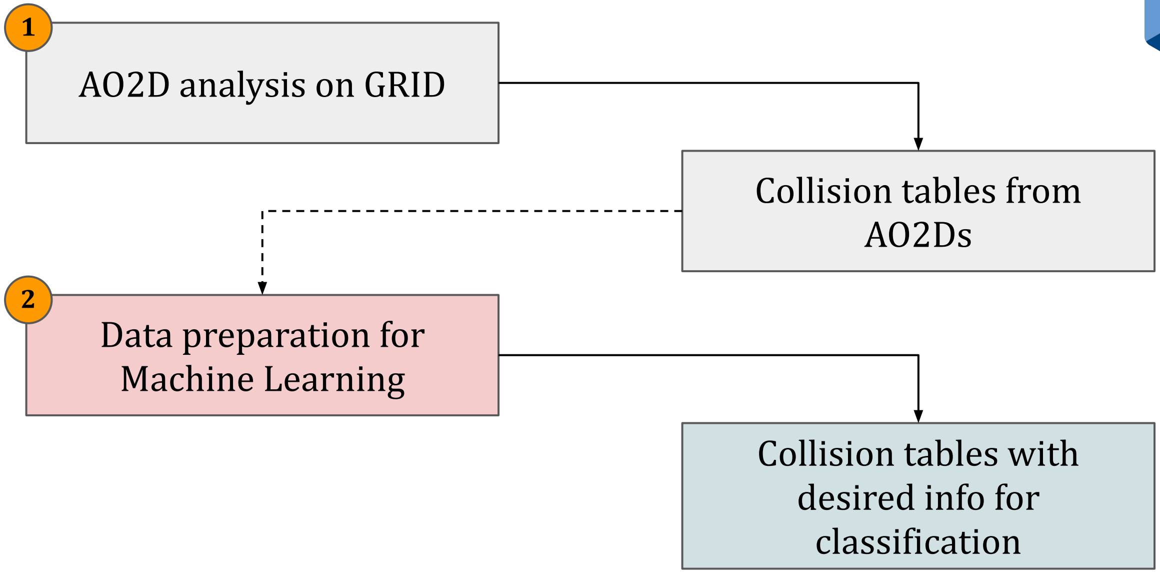
GitHub repository: https://github.com/mfaggin/monitorPvML_hpc



Mattia Faggin WIP: development of post-processing class ...

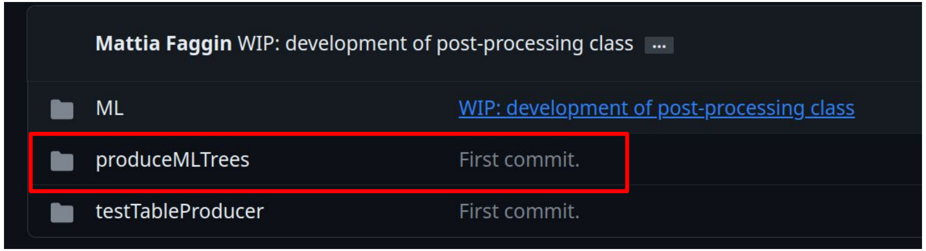
ML	WIP: development of post-processing class
produceMLTrees	First commit.
testTableProducer	First commit.

Structure of the work


1
AO2D analysis on GRID
Collision tables from AO2Ds
2
Data preparation for Machine Learning
Collision tables with desired info for classification

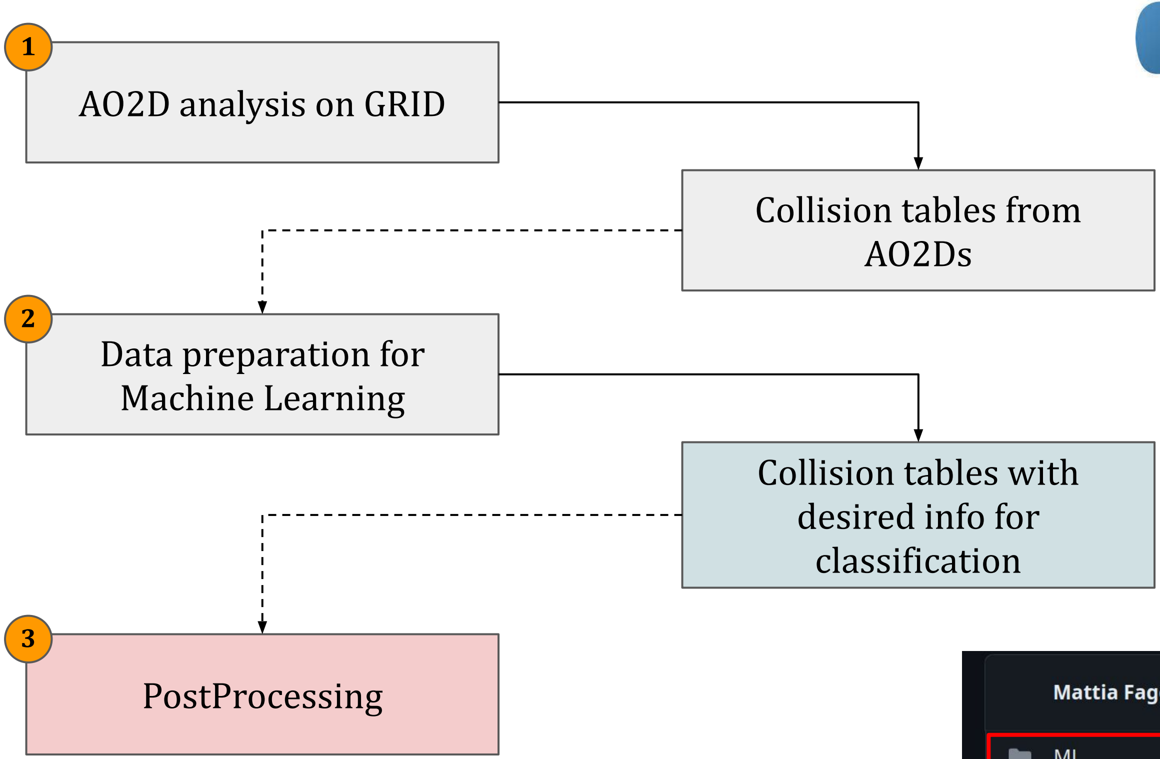
- Data from point (1) written as ROOT TTrees
- New TTrees prepared offline to setup the data as desired for the next step

GitHub repository:
https://github.com/mfaggin/monitorPvML_hpc


Mattia Faggin WIP: development of post-processing class ...

ML	WIP: development of post-processing class
produceMLTrees	First commit.
testTableProducer	First commit.

Structure of the work



1 AO2D analysis on GRID

Collision tables from AO2Ds

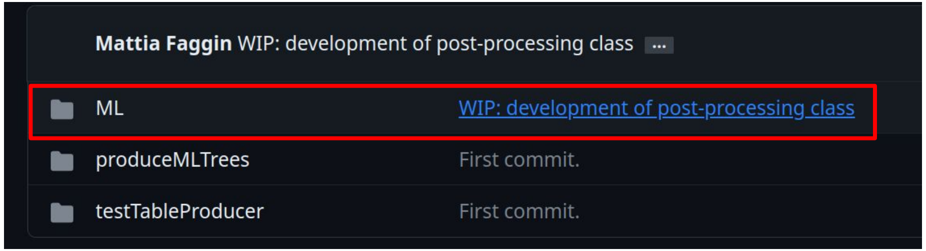
2 Data preparation for Machine Learning

Collision tables with desired info for classification

3 PostProcessing

- C++ ROOT TTree converted into Python Pandas DataFrames via upRoot
- Python libraries for Machine learning (scikit-learn, xgboost) wrapped via Hype4ML package

GitHub repository: https://github.com/mfaggin/monitorPvML_hpc

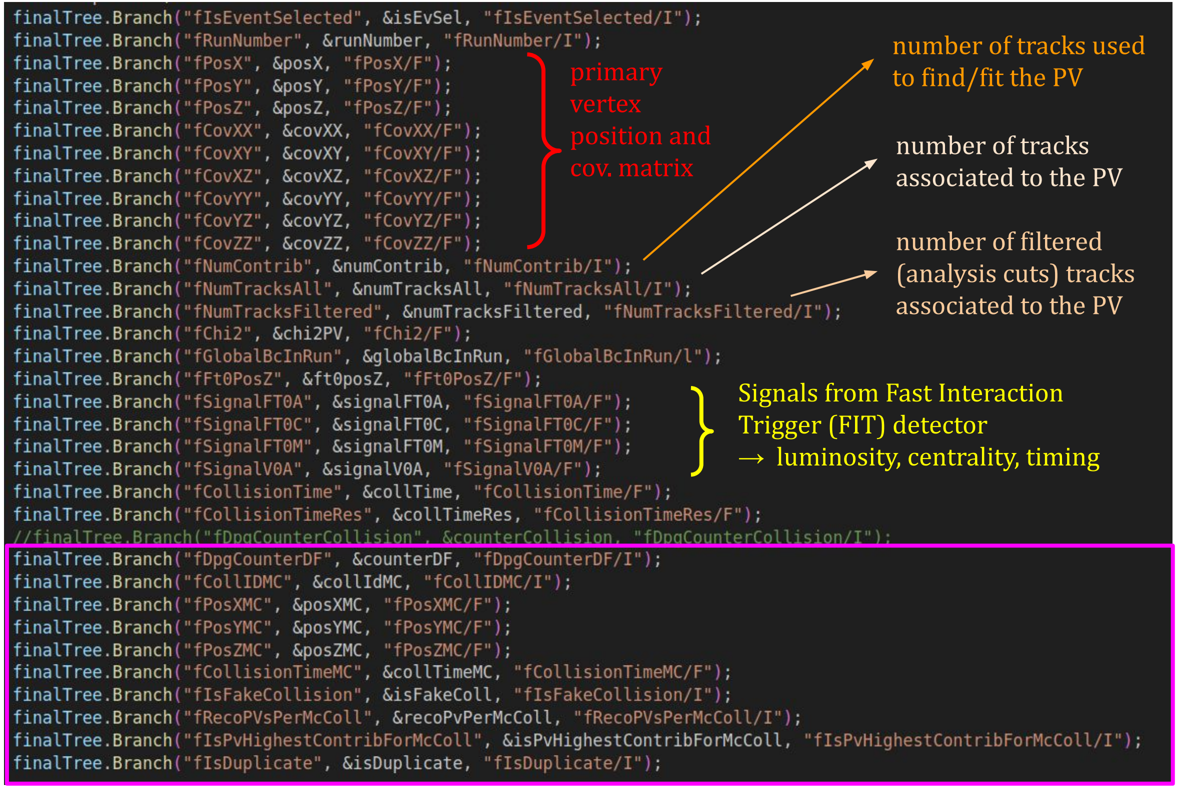


Mattia Faggin WIP: development of post-processing class ...

ML	WIP: development of post-processing class
produceMLTrees	First commit.
testTableProducer	First commit.

TTree preparation for ML

Pb-Pb collisions collected in 2022 (LHC22s) + anchored MC (LHC22l1b2) → **100-200 Hz**



```
finalTree.Branch("fIsEventSelected", &isEvSel, "fIsEventSelected/I");
finalTree.Branch("fRunNumber", &runNumber, "fRunNumber/I");
finalTree.Branch("fPosX", &posX, "fPosX/F");
finalTree.Branch("fPosY", &posY, "fPosY/F");
finalTree.Branch("fPosZ", &posZ, "fPosZ/F");
finalTree.Branch("fCovXX", &covXX, "fCovXX/F");
finalTree.Branch("fCovXY", &covXY, "fCovXY/F");
finalTree.Branch("fCovXZ", &covXZ, "fCovXZ/F");
finalTree.Branch("fCovYY", &covYY, "fCovYY/F");
finalTree.Branch("fCovYZ", &covYZ, "fCovYZ/F");
finalTree.Branch("fCovZZ", &covZZ, "fCovZZ/F");
finalTree.Branch("fNumContrib", &numContrib, "fNumContrib/I");
finalTree.Branch("fNumTracksAll", &numTracksAll, "fNumTracksAll/I");
finalTree.Branch("fNumTracksFiltered", &numTracksFiltered, "fNumTracksFiltered/I");
finalTree.Branch("fChi2", &chi2PV, "fChi2/F");
finalTree.Branch("fGlobalBcInRun", &globalBcInRun, "fGlobalBcInRun/l");
finalTree.Branch("fFt0PosZ", &ft0posZ, "fFt0PosZ/F");
finalTree.Branch("fSignalFT0A", &signalFT0A, "fSignalFT0A/F");
finalTree.Branch("fSignalFT0C", &signalFT0C, "fSignalFT0C/F");
finalTree.Branch("fSignalFT0M", &signalFT0M, "fSignalFT0M/F");
finalTree.Branch("fSignalV0A", &signalV0A, "fSignalV0A/F");
finalTree.Branch("fCollisionTime", &collTime, "fCollisionTime/F");
finalTree.Branch("fCollisionTimeRes", &collTimeRes, "fCollisionTimeRes/F");
//finalTree.Branch("fDpgCounterCollision", &counterCollision, "fDpgCounterCollision/I");
finalTree.Branch("fDpgCounterDF", &counterDF, "fDpgCounterDF/I");
finalTree.Branch("fCollIDMC", &collIdMC, "fCollIDMC/I");
finalTree.Branch("fPosXMC", &posXMC, "fPosXMC/F");
finalTree.Branch("fPosYMC", &posYMC, "fPosYMC/F");
finalTree.Branch("fPosZMC", &posZMC, "fPosZMC/F");
finalTree.Branch("fCollisionTimeMC", &collTimeMC, "fCollisionTimeMC/F");
finalTree.Branch("fIsFakeCollision", &isFakeColl, "fIsFakeCollision/I");
finalTree.Branch("fRecoPVsPerMcColl", &recoPvPerMcColl, "fRecoPVsPerMcColl/I");
finalTree.Branch("fIsPvHighestContribForMcColl", &isPvHighestContribForMcColl, "fIsPvHighestContribForMcColl/I");
finalTree.Branch("fIsDuplicate", &isDuplicate, "fIsDuplicate/I");
```

primary vertex position and cov. matrix

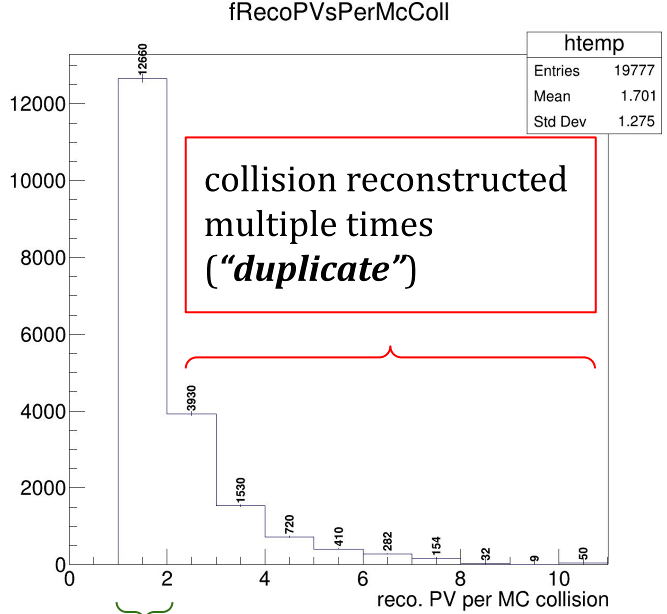
number of tracks used to find/fit the PV

number of tracks associated to the PV

number of filtered (analysis cuts) tracks associated to the PV

Signals from Fast Interaction Trigger (FIT) detector → luminosity, centrality, timing

Useful only in MC collisions



collision reconstructed multiple times (***“duplicate”***)

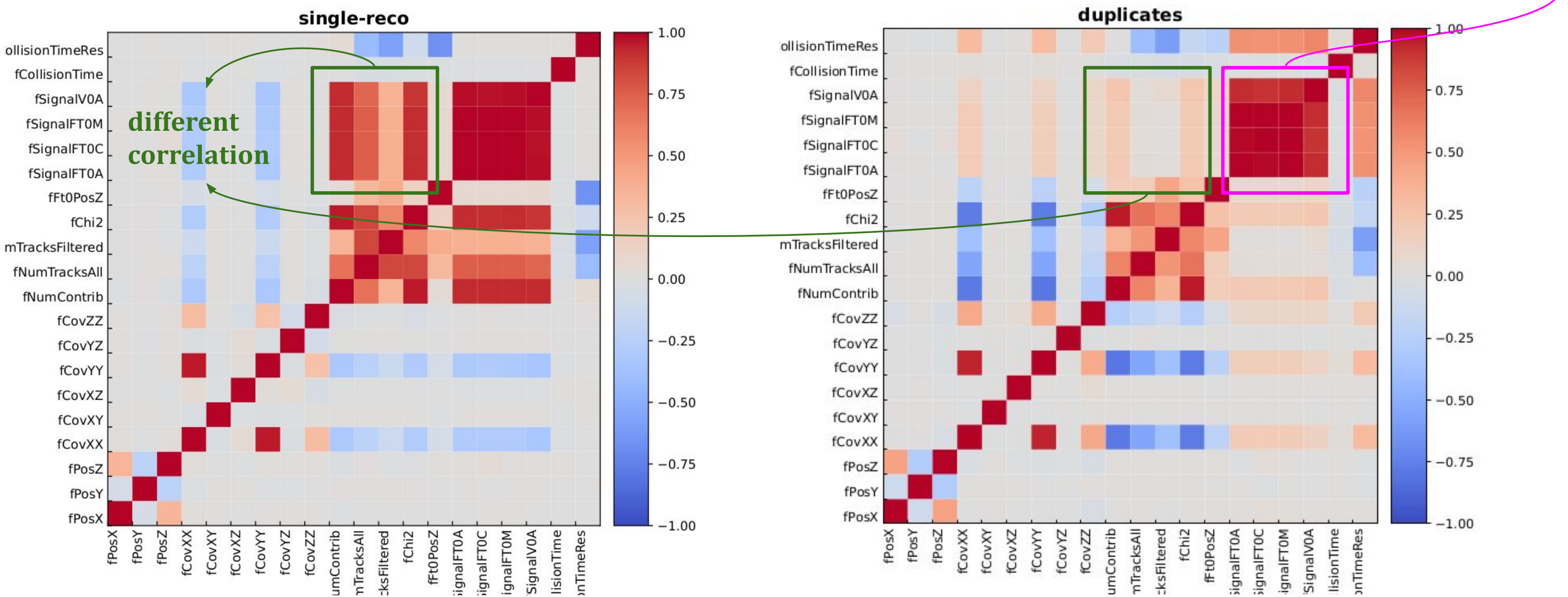
collision reconstructed 1 time (***“single”***)

BDT training

- Classifier: BDT with XGBoost
- Hyperparameter determination: optimization with Optuna (Bayes optimization)
- Training variables: fNumTracksAll, fCovXX,YY,ZZ, fChi2, fSignalFT0MfCollisionTimeRes, fSignalV0A

- Input (50% train, 50% testing):
  - single: 37354
  - duplicate: 15952

strong correlation → exploited to reduce training variables (→ reduce overfitting)

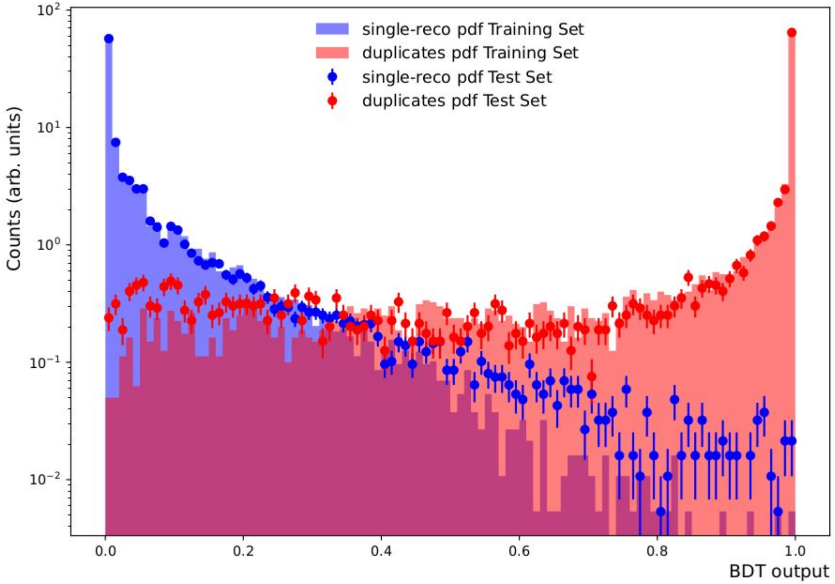


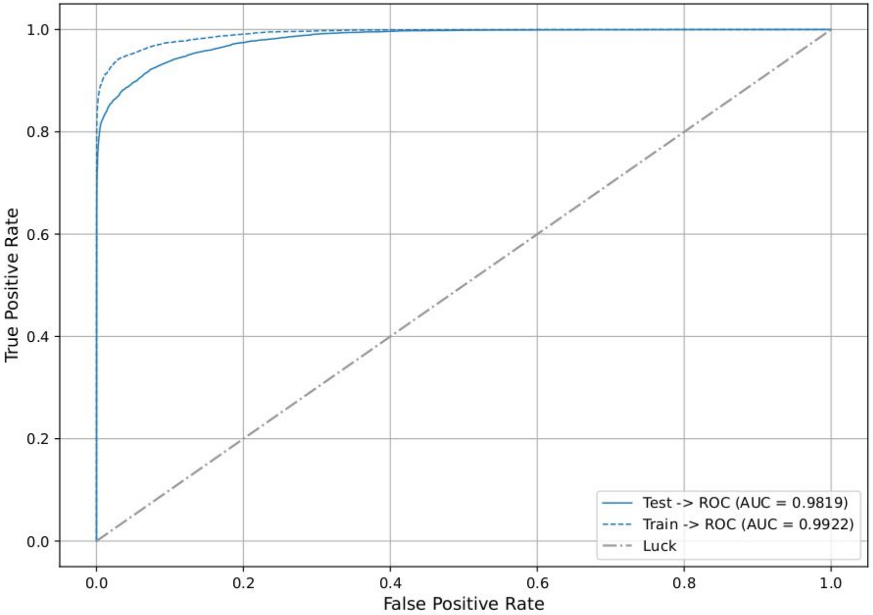
different correlation

Training and testing performance on MC

- Classifier: BDT with XGBoost
- Hyperparameter determination: optimization with Optuna (Bayes optimization)
- Training variables: fNumTracksAll, fCovXX,YY,ZZ, fChi2, fSignalFT0MfCollisionTimeRes, fSignalV0A

- Input (50% train, 50% testing):
  - single: 37354
  - duplicate: 15952





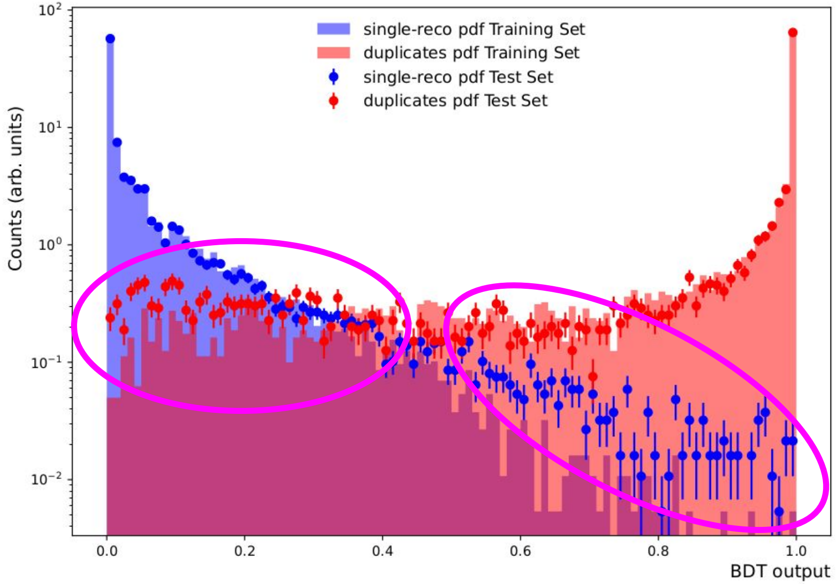
Training and testing performance on MC

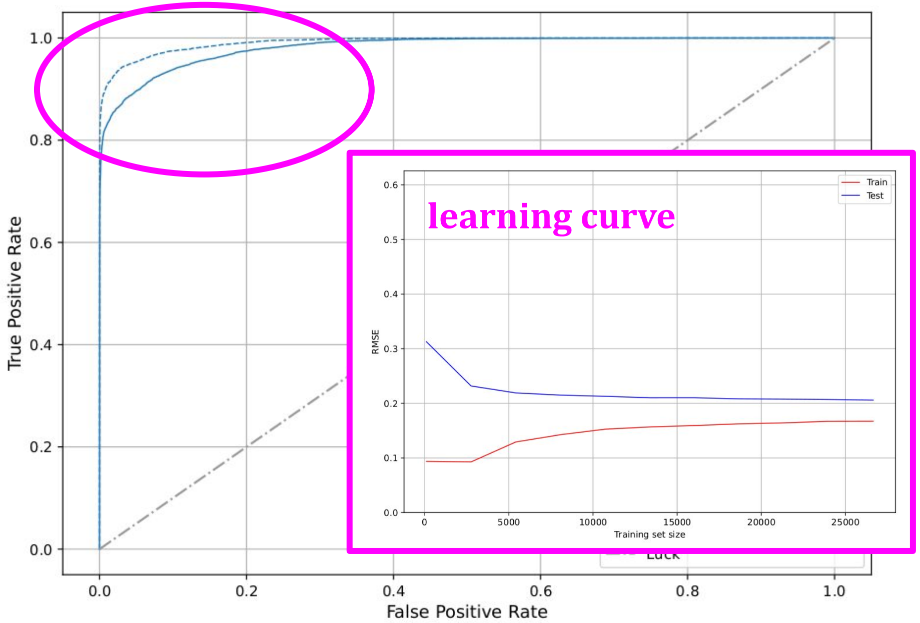
- Classifier: BDT with XGBoost
- Hyperparameter determination: optimization with Optuna (Bayes optimization)
- Training variables: fNumTracksAll, fCovXX,YY,ZZ, fChi2, fSignalFT0MfCollisionTimeRes, fSignalV0A

- Input (50% train, 50% testing):
  - single: 37354
  - duplicate: 15952

Slight overfitting?

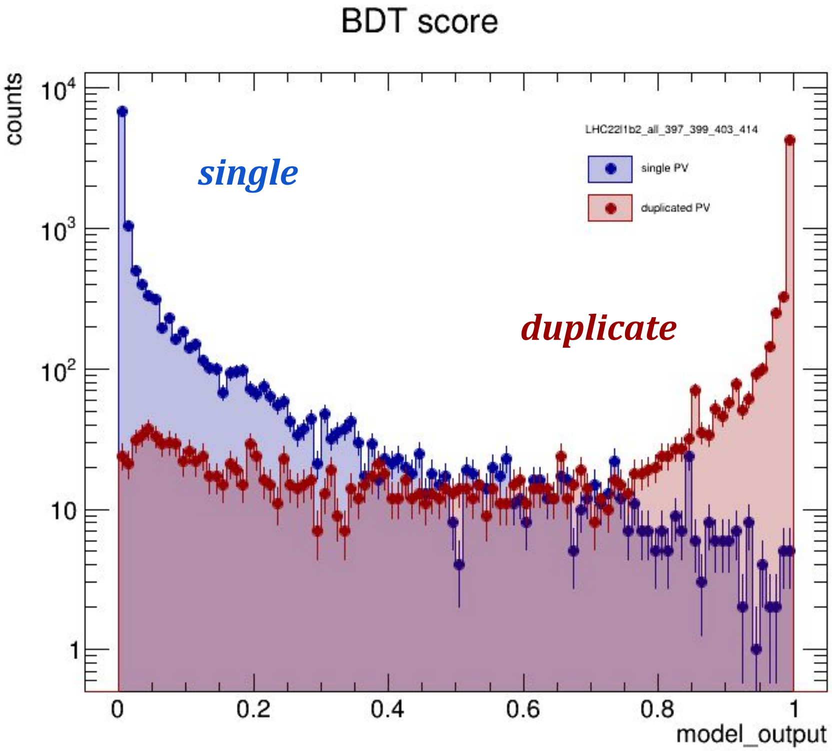
- hard to reduce # training variables
- probably more inputs needed (currently not available)





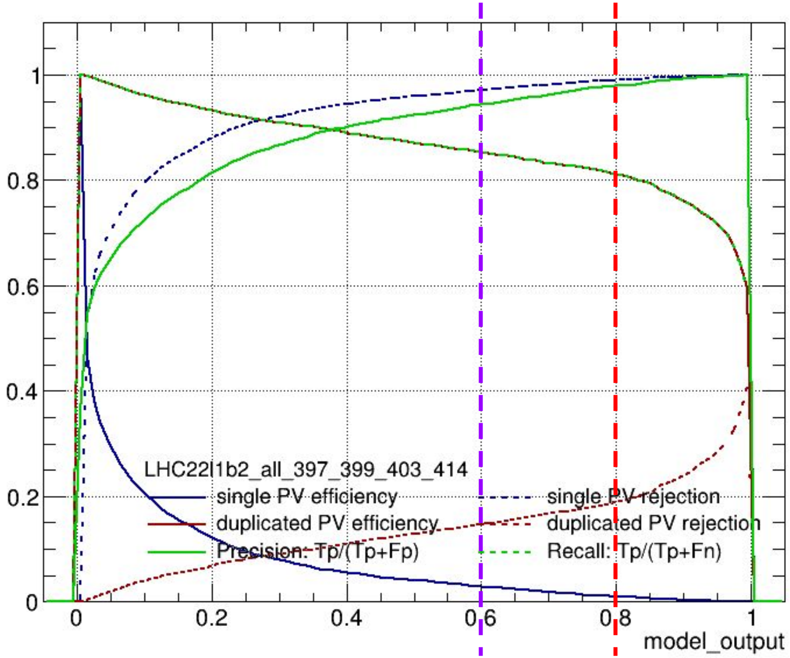
Application on MC

- Application on a MC sample independent from that used for the training and testing



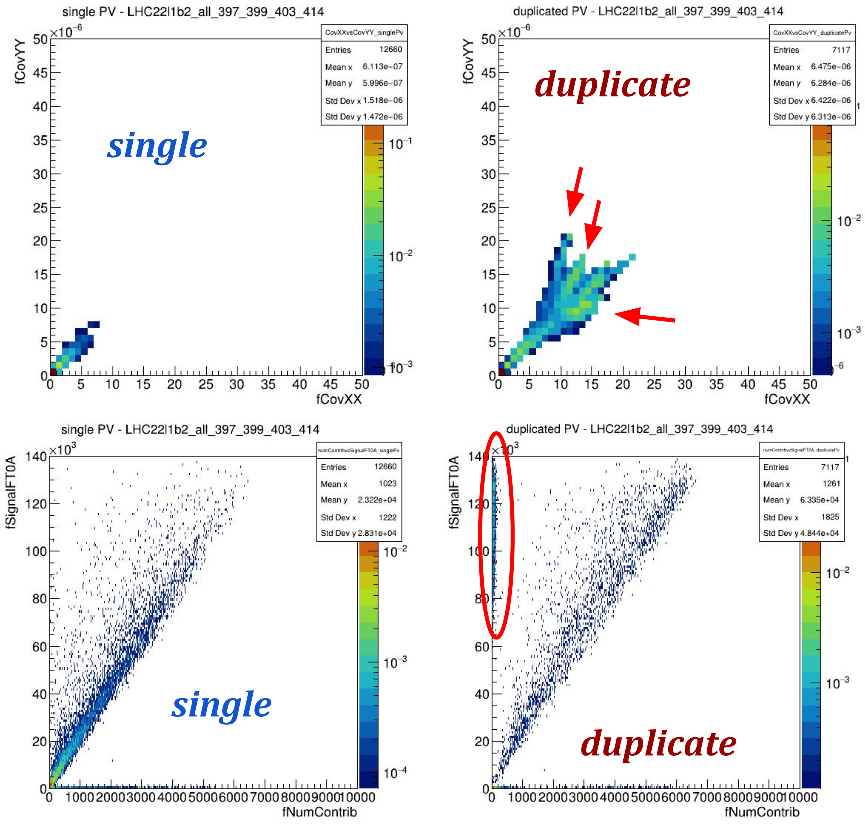
duplicate efficiency ~ 85%
duplicate purity > 90%

duplicate efficiency > 80%
duplicate purity > 95%



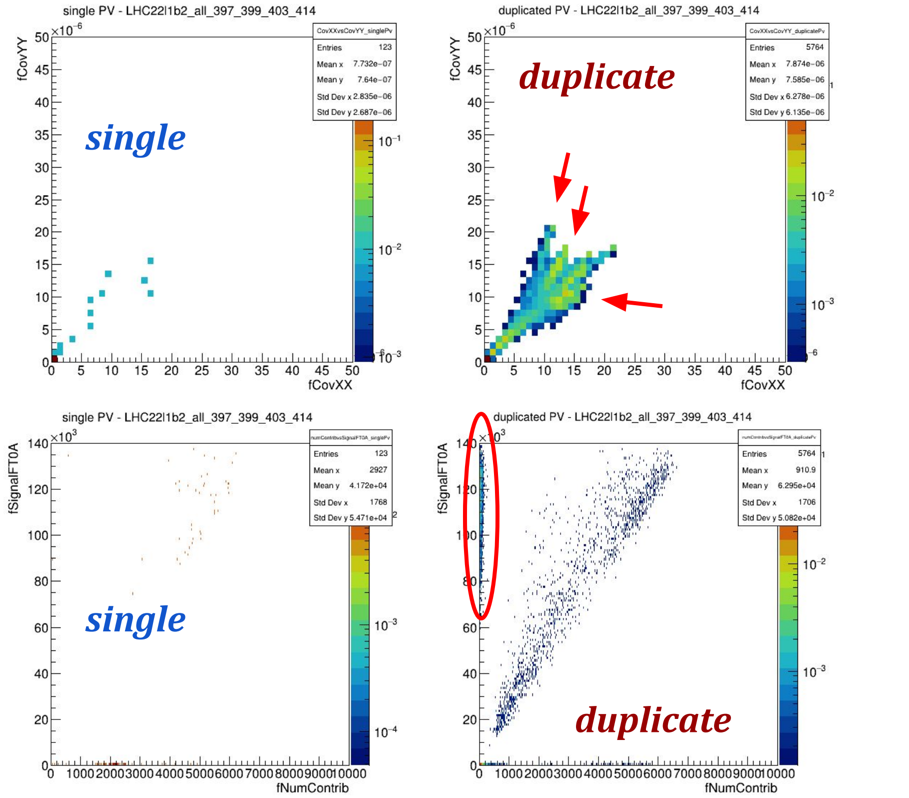
Application on MC

No BDT selections



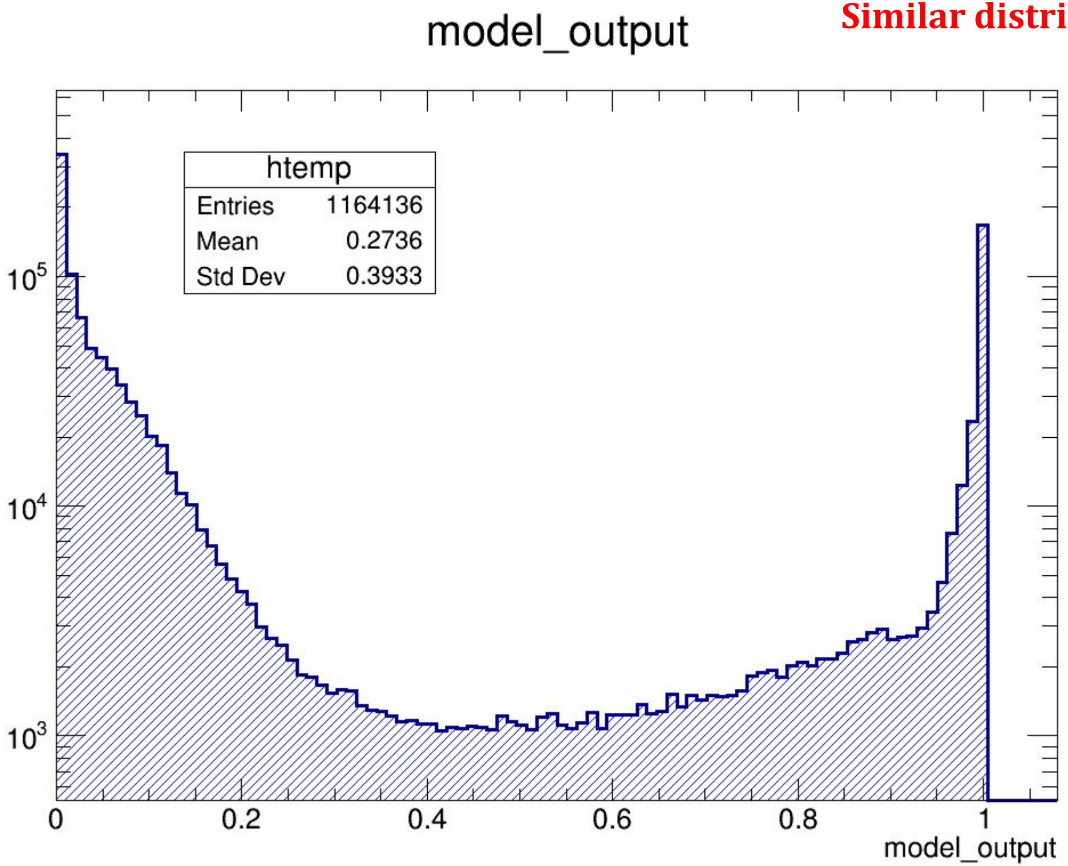
BDT score > 0.8

features still present after the selection!



Application on data

Similar distribution as observed in MC!



Summary and outlook

Summary

- Machinery to build a binary classification to tag duplicated vertices in place

- First validation on MC productions anchored to the collected data done

- First application on collected data done and promising

Outlook / next steps / criticalities

- Post-processing code to be completed to completely handle also the application on data (so far: full post-processing available only for application on MC)

- Strategies to use the tagged duplicated vertices to be discussed
  - duplicates need to merged… who with who?

- Data reconstruction in fast evolution
  - imperfections (and bugs) found on a daily basis
  - detector calibrations still partial (e.g. TPC distortions)
  - MC productions not perfectly reproducing the properties of data (e.g. track properties)

Thank you for your attention