Muons Have Commitment Issues

Catching decays in flight in MUSE with machine learning (and with a machine that wrote the machine learning)

Ethan Cline

with **Claude** (Anthropic) as pair-programmer

Center for Frontiers in Nuclear Science
Stony Brook University
Stony Brook, NY

Machine Learning Workshop, 2026

This work is supported by the National Science Foundation, grants PHY-2012114 and 2412703 to Stony Brook University. The MUSE experiment is supported by the U.S. Department of Energy, the U.S. National Science Foundation, the Paul Scherrer Institute, and the US-Israel Binational Science Foundation.

Stony Brook University



MUSE

Full disclosure: three authors, one of them human

The physicist

- the physics question
- what counts as a decay
- which detector is where
- saying "no" to bad ideas
- the result is our responsibility

Claude

- most of the ML code
- C++ inference in the cooker
- the checks on all of the above
- a review that found our leakage
- the slides you are looking at

The BDT

- finds 98–99% of the decays
- 10× fewer false alarms than the neural net
- never complains
- cannot attend this workshop

This talk is about the ML, and about what it was like to build it with an AI pair-programmer.

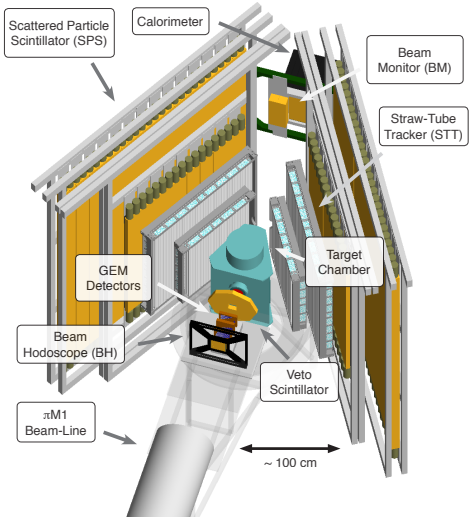
Outline

1. **The physics:** why a muon that decays early is a problem
2. **The ML problem:** labels, priors, missing values, and a pipeline that runs in C++
3. **The models:** boosted trees vs. a neural net, and why the trees win
4. **Generalisation:** does a model trained on one simulation work on another?
5. **The AI in the loop:** what **Claude** built, what it caught, and where it was wrong

The physics

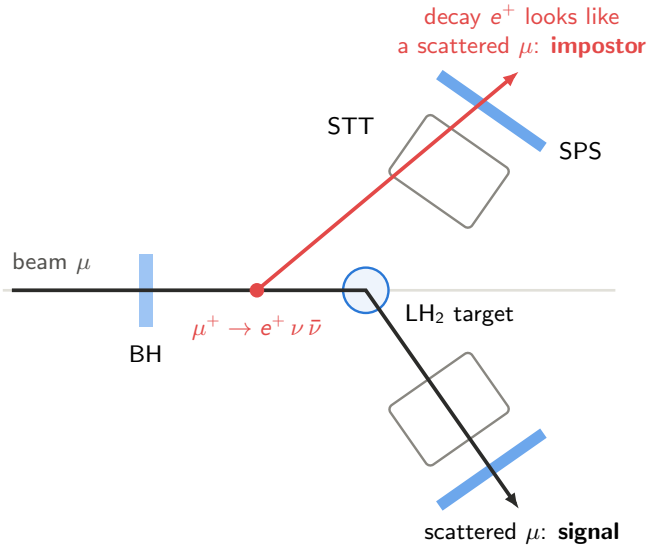
why a muon that decays early is a problem

MUSE: the MUon Scattering Experiment at PSI



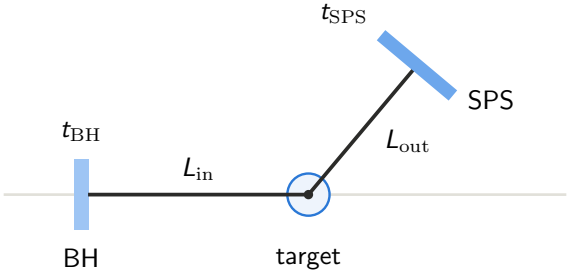
- Is the proton the same size to an electron as to a muon? (*the proton radius puzzle*)
- Scatter $e^{\pm}$ and $\mu^{\pm}$ off liquid hydrogen, at the same time, in the same detector
- Beam momenta 115, 160, 210 MeV/c; the beam is a mix of e, μ and π
- For each event: was this a muon that **scattered** off a proton?

Muons have commitment issues



- Muon lifetime $\tau = 2.2\,\mu$s
- At 210 MeV/c: $\beta\gamma \approx 2$, mean decay length ≈ 1.3 km
- Rare per event ($\approx 0.13\%$ near the target), but MUSE records a lot of events
- The decay e^+ reaches a scattering arm and **impersonates** a scattered muon

The physicist's first try: one cut on the outgoing speed β



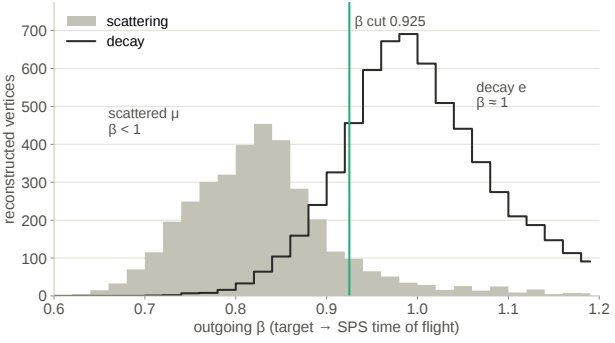
$$\beta_{\mathrm{out}} = \frac{L_{\mathrm{out}}}{c\, t_{\mathrm{out}}}, \qquad t_{\mathrm{out}} = t_{\mathrm{SPS}} - t_{\mathrm{BH}} - \frac{L_{\mathrm{in}}}{\beta_{\mathrm{in}}\, c}$$

decay e: $\beta_{\mathrm{out}} \approx 1$

scattered μ, 210 MeV/c: $\beta_{\mathrm{out}} \approx 0.85$–$0.89$

A one-feature, hand-tuned classifier. Physicists have been doing "ML" for a long time.

Why a single β cut is not enough



Reconstructed vertices in simulation, 210 MeV/c

- Cut: track with $\beta_{\text{out}} > 0.925$ is tagged as a decay
- 85% efficiency, 92% purity
- The distributions overlap: timing resolution, path length, energy loss
- The detector records much more than β.
  $\Rightarrow$ **Give all of it to a classifier**

The ML problem

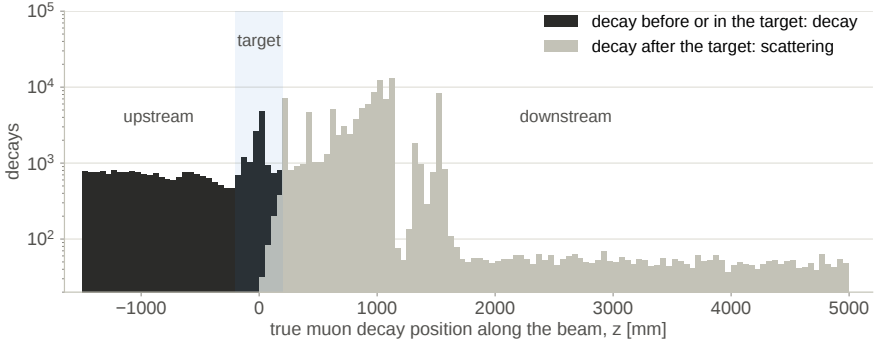
labels, priors, missing values, and a pipeline that runs in C++

The problem, in ML terms

task	binary classification, one event at a time
input	135 tabular features per event: times, energy deposits, track and vertex geometry, hit multiplicities, from 13 reconstruction stages and six detector systems
label	muon decayed *before or inside* the target (from simulation truth)
training data	$> 10^6$ simulated events, three independent productions (μ^+, μ^-, μ^+)
the catches	the class prior in simulation is $\sim$100$\times$ the real one; most features can be missing; the simulation is not the data
deployment	inside the experiment's C++ reconstruction framework (the "cooker"), on every event

Not a big-data problem. The hard parts are the labels, the priors, and getting the same features in training and in production.

Choosing the label is a physics decision



True decay position of every muon that decays in the simulation

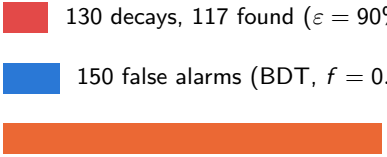
- Decays happen from $-1.5\,\text{m}$ to $+5\,\text{m}$ along the beam; only $\sim$10% are near the target
- A decay *after* the target means the muon already scattered: that event is **signal**, not background
- The first version's label was "did it decay *anywhere*?", metres downstream included: a model answering the wrong question

The class prior: simulation is not nature

- Simulation is **enriched** in decays so there is something to learn from
- Positive class: 13–16% of simulated events
- Real rate over the same region: $\pi \approx 0.13\%$
- Accuracy and average precision on the training mix are **not physics numbers**
- Measure efficiency ε and false-positive rate f, then rescale to the real prior:

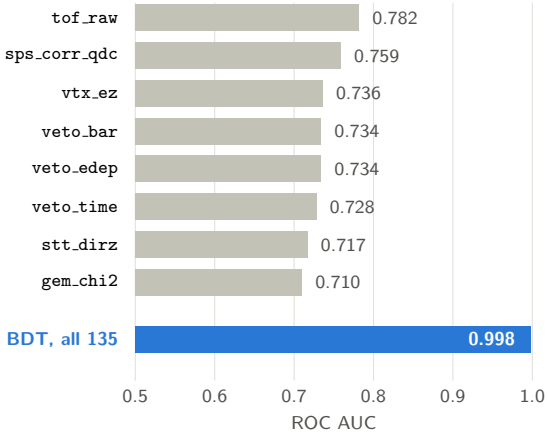
$$\text{purity} = \frac{\varepsilon\,\pi}{\varepsilon\,\pi + f\,(1-\pi)}$$

per 100 000 real muon events



even the BDT's 0.15% false-alarm rate gives purity $\approx 44\%$; the MLP's, 10%

No single feature does it



Best single inputs, each used alone as a score, vs. the BDT

- The best single input is a raw time of flight: AUC 0.78
- The information is spread over timing, energy, tracking, vertexing, and the beam monitor *downstream* of the target
- Dropping any one group barely moves the AUC (0.998 is saturated), so we count **misclassified events** instead
- Without the per-wall scintillator times: 47% more errors

Missing is a feature, not a zero

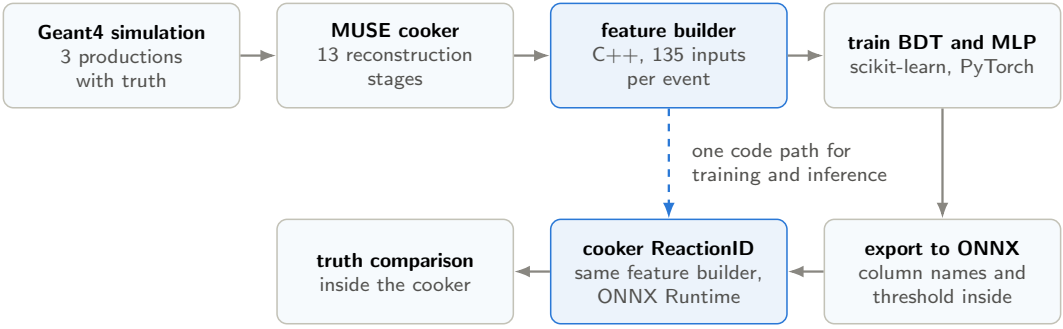
- Detector systems fail to report for physics reasons: no track, no vertex (only $\sim$67% of events have one), no hit
- **NaN means "not measured"**. A sentinel like -10000 is a lie the model will believe
- Trees: NaN gets its own branch at every split
- Network: NaN $\rightarrow$ 0, plus an `_isnan` flag per column. Together with one-hot categories: 135 $\rightarrow$ 438 inputs

A warning sign

A BDT trained on **only** the 105 "is missing" flags reaches AUC **0.91**.

How reconstruction fails carries most of the signal, and real data may fail differently than simulation.

One feature builder for training and inference



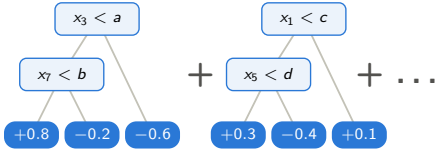
- Training/serving skew is the classic silent killer. Here it **cannot** happen: the CSV and the cooker vector come from the same C++ calls
- Both models export to one contract, `float32[N,135]` in, one probability out, so the C++ has one code path; the model is a config option
- The MLP's scaling and one-hot encoding live inside its ONNX graph, so the C++ never reimplements them

The models

boosted trees vs. a neural net

Two machine-learning techniques

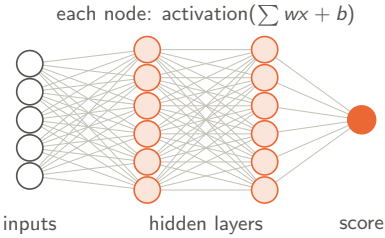
Boosted decision trees (BDT)



score $= \sigma\left(\sum \text{leaves the event lands in}\right)$

- Many shallow trees, each fixing earlier errors
- Cuts on one input at a time: scale-free
- Missing values take their own branch

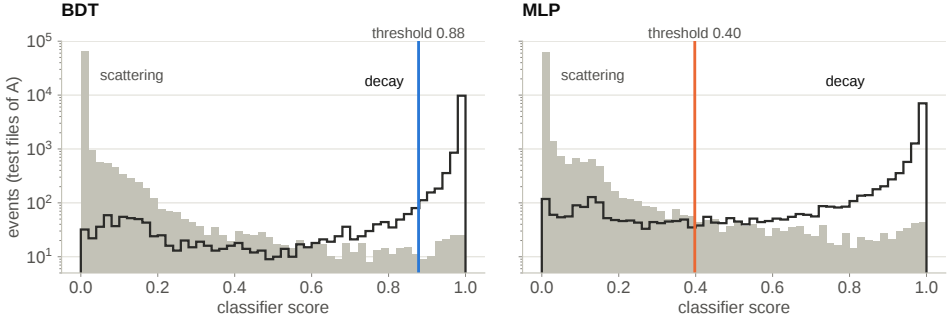
Neural network (MLP)



- Layers of weighted sums and nonlinear activations
- Smooth in all inputs at once: needs scaling
- Missing values must be filled in and flagged

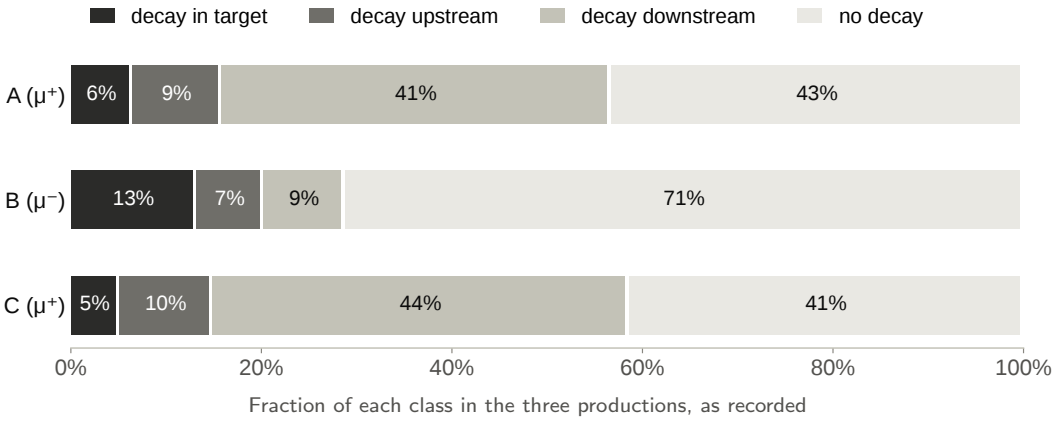
Our two classifiers

	Boosted decision trees	Neural network (MLP)
library	scikit-learn `HistGradientBoosting`	PyTorch
inputs	135 raw values	438 after one-hot, missing flags, scaling
missing values	handled natively	zero + "is missing" flag
model	$\approx$ 400 trees (early-stopped)	4 hidden layers, 256–512–1024–256
threshold	90% efficiency on a 15% training holdout	same



Classifier score on A's test data; vertical line is the threshold

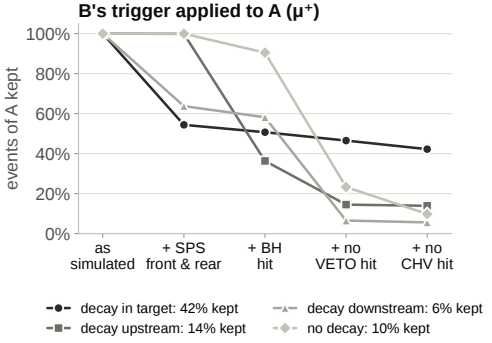
Three independent simulations

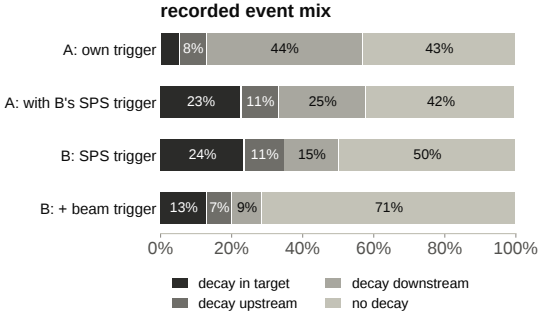


Fraction of each class in the three productions, as recorded

Same physics, different class balance: 5–6% target decays in A and C, 13% in B.

The trigger sets the class balance, not the physics

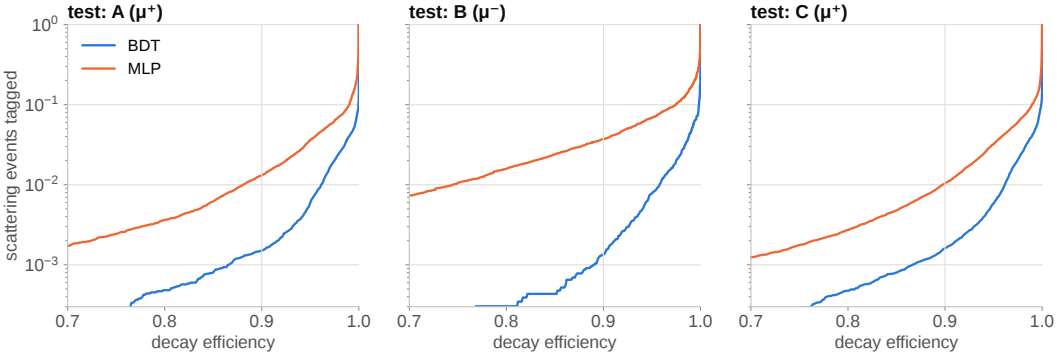




Left: fraction of A's events kept as B's trigger requirements are added. Right: the recorded mix.

Applying B's trigger to A reproduces B's mix. A built-in domain shift between our training sets, and a preview of what real data, with its own trigger, will do.

Held-out performance



Models trained on all three productions

scattering events tagged at 90% efficiency	A (μ^+)	B (μ^-)	C (μ^+)
BDT	0.15%	0.13%	0.16%
MLP	1.3%	3.7%	1.0%

The BDT's false-positive rate is $\sim 10\times$ lower than the MLP's at the same efficiency.

Why do the trees win?

- **Step functions:** a detector either fired or it did not. Trees cut, networks prefer smooth functions
- **Missing values:** a native branch vs. $\sim$300 extra flag and one-hot inputs the network has to learn to use
- **Mixed units:** ns, MeV, mm, bar numbers. Trees do not care; networks need careful scaling
- **Useless inputs:** many of the 135 columns add little. Trees ignore them; networks fit noise

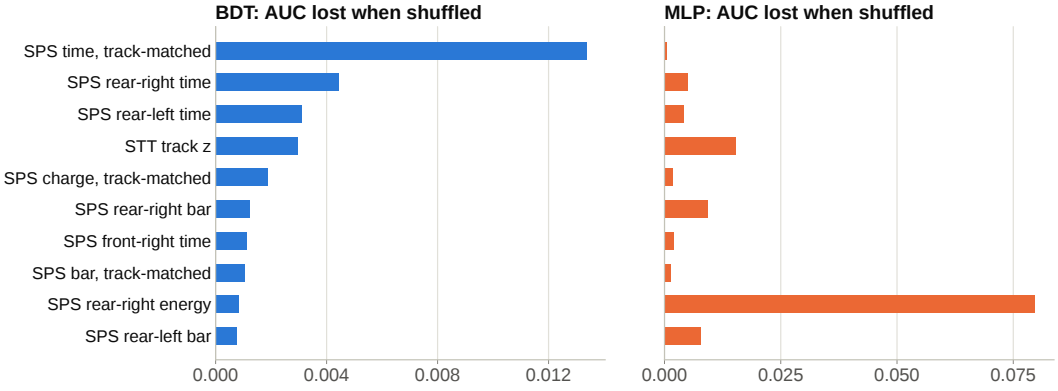
Not just us

Same conclusion across 45 tabular benchmarks:

L. Grinsztajn, E. Oyallon, G. Varoquaux, *Why do tree-based models still outperform deep learning on typical tabular data?*, NeurIPS 2022 Datasets & Benchmarks

Lesson for physics: try a gradient-boosted tree **first**. It is the strong baseline the fancy model has to beat.

What the models use



Drop in AUC when one input is shuffled; A's test events, models trained on all three productions

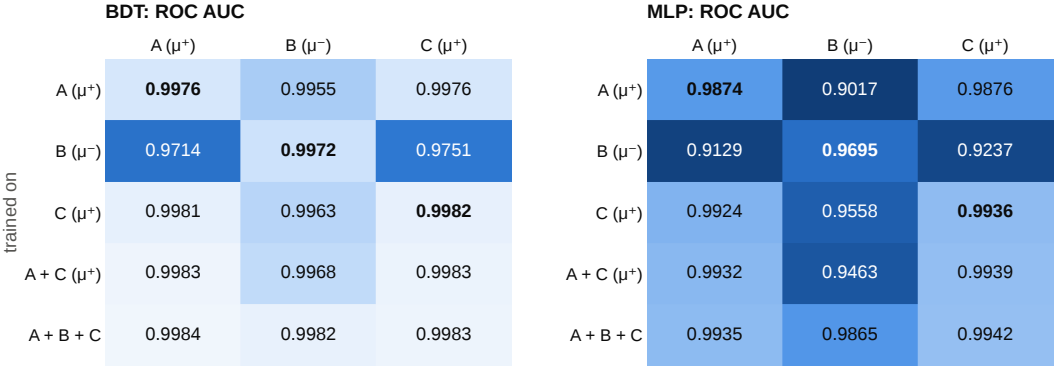
Both models rely most on scattered-particle (SPS) timing; the MLP also leans heavily on an SPS energy deposit.

Reassuring: the physics we expected (β, i.e. timing) is what the trees found.

Generalisation

does a model trained on one simulation work on another?

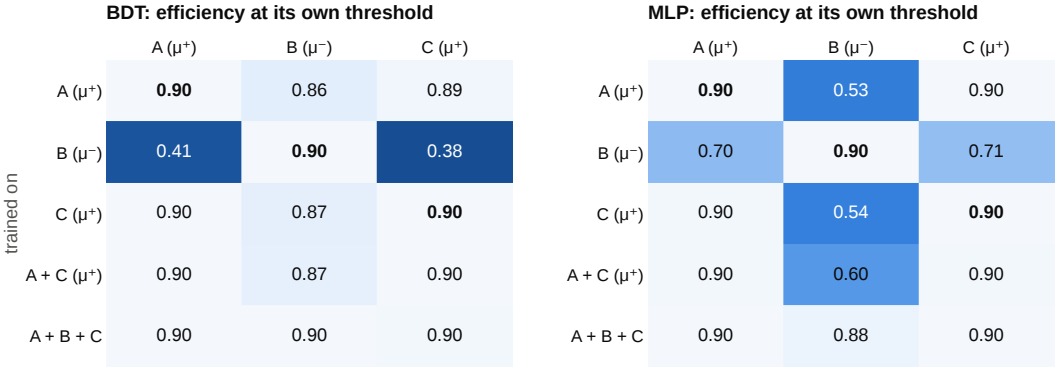
Train on one simulation, test on another



ROC AUC. Rows: training sample; columns: test sample; darker = lower AUC

BDT: no loss from A→C, or from A/C→B. Trained on B alone, it transfers worse to A and C.

The ranking transfers, the operating point does not

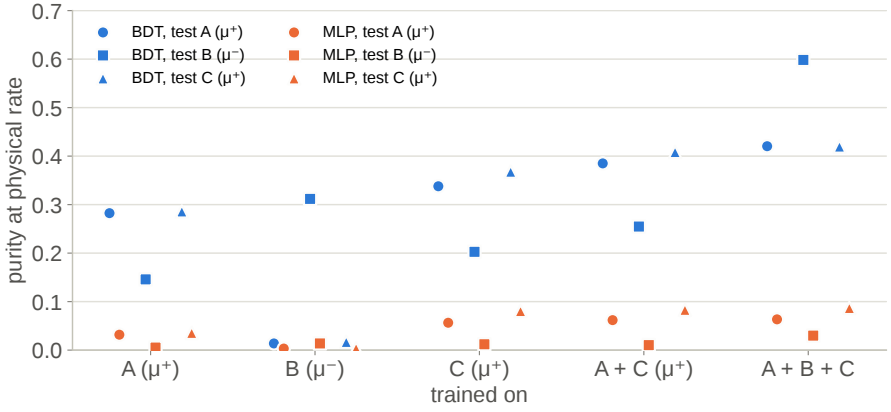


Decay efficiency at each model's own threshold, set for 90% on its training holdout; darker = further below 90%

Trained on B alone, the BDT keeps only 41% and 38% of the μ^+ decays. Pooled training holds 90% on all three.

AUC can look fine while the threshold is miscalibrated: **check efficiency at the threshold, not only AUC**.

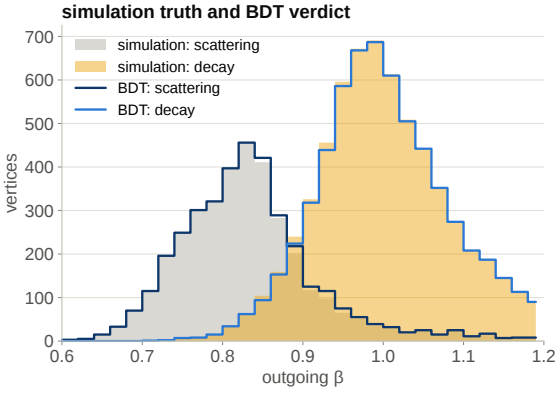
Purity at the real decay rate

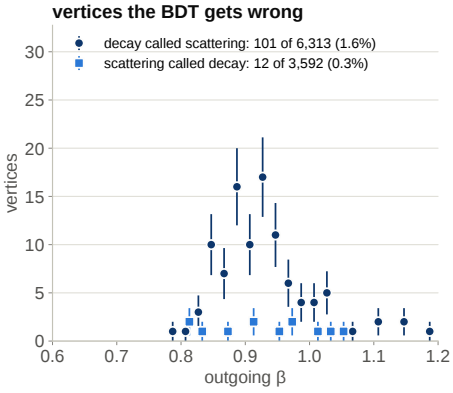


Purity rescaled to the physical prior, by training set and test sample

More diverse training data helps every test sample. The MLP never gets above 10%.

What the BDT flags





Left: filled, simulation truth; lines, the BDT's verdict. Right: incorrect predictions.

Of 6 313 true decays, 101 are missed; of 3 592 scatters, 12 are wrongly flagged.
The misses pile up at the edges of the label region, where "before the target" and "in the target" blur.

Scorecard

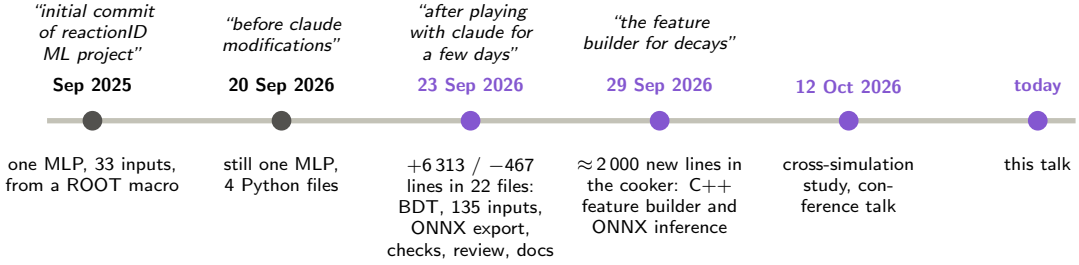
	BDT	MLP	β cut
efficiency, reconstructed vertices	98–99%	91–94%	84–86%
purity, reconstructed vertices	$\geq 99.7\%$	90–94%	91–92%
scattering events tagged at 90% efficiency	0.13–0.16%	1.0–3.7%	–

- The BDT beats both the neural net and the physicist's cut
- Pool several simulations for training; a single production does not generalise
- Real data next: 115 and 160 MeV/c, mixed $e/\mu/\pi$ beam, recalibrated times

The AI in the loop

what **Claude** built, what it caught, and where it was wrong

The git log tells the story



- Tool: **Claude** in **Claude Code**, an agent in the terminal that reads the repository, runs code, and edits files
- The physicist commits; the commit messages are the physicist's own
- Fun fact: ≈ 2 600 of those 6 313 new lines are scripts that *generate slides*. Priorities.

Before and after

	before	with **Claude**
models	one MLP	BDT (default) and MLP, same interface
inputs	33	135, from six detector systems
label	decayed anywhere	decayed before or inside the target
features	ROOT macro; its header declared 37 columns, its rows wrote 21	one C++ library, used by both training export and inference
missing values	sentinels: -1, -10000	NaN + flags; trees handle NaN natively
deployment	Python only	ONNX in the C++ cooker; model chosen by config
evaluation	loss and score plots	ROC, efficiency at threshold, purity at the real prior, per-region breakdown, ablation
verification	none	ONNX vs. native; cooker vs. Python on every event; leakage audit
documentation	none	README and a 560-line guide for the next student

Also found in the old code: a stray copy of an unrelated straw-tube project, the source of mysterious "Hit Radii" labels on the plots.

Claude as reviewer: it found our data leakage (and its own)

issue	effect	fix
MLP early-stopped on the validation file used to quote results	optimistic metrics	early stopping on a 15% training holdout
thresholds chosen at 90% efficiency *on the validation file*	"90% efficiency" true by construction	thresholds set on the holdout
report refitted its own trees instead of loading the shipped model	report and production could diverge	loads the shipped model
false-positive rate counted downstream decays as background; prior 0.003	purity overstated	FPR on true non-decays; prior 0.0013
β-cut benchmark calibrated on the file it was scored on	circular comparison	calibrated on training file
C++ threshold option silently overwritten by model metadata	user setting ignored	explicit setting wins, logged
7% of output rows repeat the previous event (upstream blinding)	stale verdicts	entry-index check

The threshold and report bugs were in code **Claude** itself had written three days earlier.

Honest numbers are lower, and that is the point

	AUC	threshold	efficiency at threshold	purity at real prior
BDT	0.9984 → 0.9984	0.893 → 0.862	90.08% → 91.09%	63.0% → **33.5%**
MLP	0.9960 → 0.9949	0.569 → 0.462	90.01% → 90.36%	25.8% → **7.8%**

Internal review, September 2026: before → after the fixes; one μ^+ production, validation file

- Before: 90% efficiency was exactly 90.0% because the threshold was *picked on that file*
- After: 91.1%. Not exactly 90 is the sign the file was not used to tune
- AUC did not move. **Purity halved**. Leakage hides in the operating point, not in the ROC curve
- An AI that helps you write faster also helps you fool yourself faster. Ask it to review too

Trust, but verify

Checks that run every time

- **ONNX vs. native model**, before shipping
- **Cooker vs. Python, event by event**: 101 380 events, mean $|\Delta\text{score}| = 5 \times 10^{-8}$, 3 above 10^{-4}
- **Train/validation overlap**: 1 shared row in 101 380
- **Single-input leakage screen**: best single input AUC 0.78, so no input gives the answer away
- **Schema check**: the C++ refuses to run a model that wants a column it does not produce

Things it dug out

- A bug in `skl2onnx`: trees with NaN splits export bool attributes ONNX rejects. Bites "exactly ours" (its words); worked around, guarded by the check
- Merging simulation files with ROOT's `hadd` silently corrupts the run number → wrong timing calibration for every event
- The SPS PID columns were constant, a missing calibration upstream

Rule we followed: **Claude** writes the check along with the code, and the check's output is what gets believed.

Where the physicist had to step in

- Put the beam monitor *upstream* of the target. It is downstream, which is why it helps
- Assumed the decay positron skips the reconstructed vertex. It leaves from it
- Re-derived the vertex in the exporter instead of reading the vertexing plugin: two definitions of one quantity
- Turned a small shared library into a plugin plus a new data tree. Not needed
- Answered a one-line build fix with a shared CMake module across four packages: *"far too convoluted"*

It writes the lessons down

Claude Code keeps notes between sessions:

```
- Plugins own their quantities
- Plots belong in the cooker
- MUSE detector layout
- decayfeatures is a library
- The decay point is a vertex
- Constants belong in init XML
- Minimal review fixes
```

A collaborator's commit, 9 Oct 2026:

`"Go home Claude you're drunk"`

Strong on software, weak on *your* detector. Like a new student it needs a physicist, but it learns faster.

Claude also wrote this talk

The prompt, in full:

> *"There is an existing presentation for muon decay identification. Create an expanded presentation and come up with a funny title. The focus of the presentation should be on the use of ML to identify muon decays, as the talk is for a ML workshop. Also highlight the use of Claude in generating the ML algorithm, and the talk itself"*

What it did:

- Found the conference talk in the repository and reused its figures and diagrams
- Read the git history, the review deck, the result files and its own notes to get the numbers
- Wrote the LaTeX and TikZ, compiled it, and checked the pages

Titles that did not make the cut:

- Dead Muons Don't Scatter
- It's Not a Muon, It's a Positron in a Trench Coat
- Gradient Boosting for the Dearly Departed
- Attention Is All You Need (to Pay to Timing)
- 2.2 Microseconds of Fame

The internal review deck was generated by a script **Claude** wrote, straight from the result files, so its numbers cannot drift from the analysis.

Lessons for physicists working with an AI pair-programmer

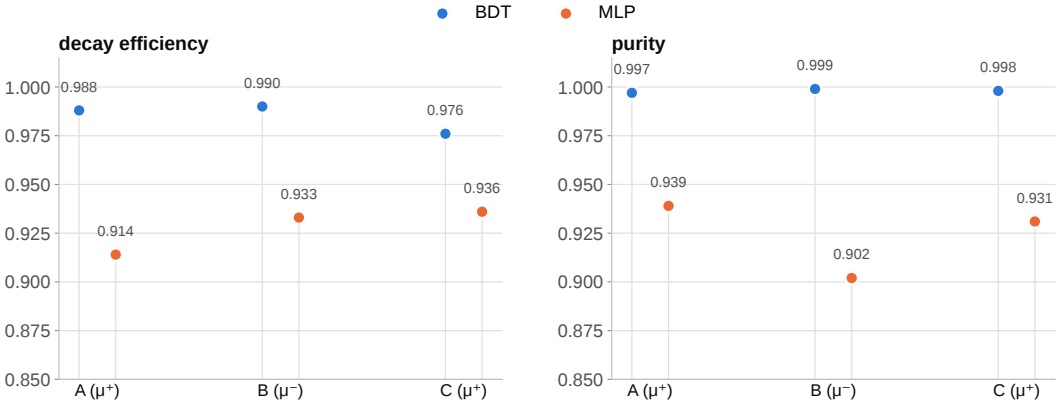
1. **Great at the plumbing** that takes students months: ONNX export, C++ integration, build systems, parity checks, documentation
2. **A tireless reviewer.** Ask it to review its own code and yours; it found the leakage that inflated our purity by $2\times$
3. **It does not know your detector.** Every physics statement needs a physicist to check it
4. **Write the conventions down.** Corrections kept as notes stop the same mistake next week
5. **Believe the checks, not the prose.** Ask for the verification script along with the code
6. **The ML basics still matter most:** label definition, real-world priors, held-out data used once, efficiency at the operating point

Summary

- Muons that decay in flight impersonate scattered muons; a single β cut finds 85% of them
- A boosted decision tree on 135 detector features finds 98–99% with $\geq 99.7\%$ purity on reconstructed vertices, and beats a neural net by $\sim 10\times$ in false-positive rate
- It runs inside the experiment's C++ reconstruction, from the same feature code it was trained on
- Training on several simulations is what makes the operating point transfer
- Most of the code, the checks, the review that caught our leakage, and this talk were written with **Claude**; the physics and the judgment calls were not
- Next: real data at 115 and 160 MeV/c with a mixed beam, and relative instead of absolute times

Thank you! Questions for either author welcome.

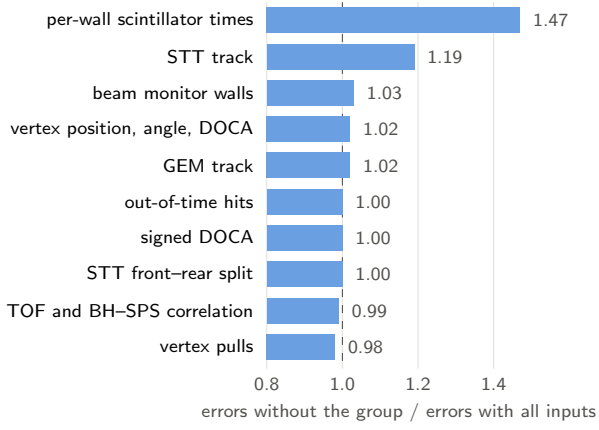
Backup: BDT vs MLP, per production



Reconstructed vertices, held-out files of each production, models trained on all three productions

BDT: 98–99% efficiency with $\geq 99.7\%$ purity on every production; MLP: 91–94% efficiency, 90–94% purity.

Backup: feature-group ablation



- BDT retrained with one group of inputs removed
- AUC stays 0.998 in every case; the error count moves
- Timing carries the most unique information; much of the rest is redundant
- Only the 105 missing-value flags: AUC 0.91
- Only the 14 RF-timing inputs: AUC 0.976

Backup: what the cooker writes for every event

field	meaning
`score`	decay probability; NaN when nothing scored the event
`is_decay`	`score > threshold`
`threshold`	operating point, from the model's own metadata
`inputs_valid`	a model ran on this event
`model_id`	1 BDT, 2 MLP, 0 none
`run`, `entry`	provenance; a row is its own result only when `entry` matches the tree index

- Model chosen by config: `-c ReactionID:setModel:"decay_mlp.onnx"`
- Each ONNX file carries its own input column names, threshold and model tag
- The plugin maps the names onto the feature builder at startup and refuses to run if any is unknown