Boosted Decision Trees redesign for oscillation pipeline and beyond

A brief introduction to my PhD project and current status…

Michalis Chadolias

IFIC (CSIC-UV) | VEGA

INSTITUT DE FÍSICA CORPUSCULAR

MINISTERIO DE CIENCIA E INNOVACIÓN

UNIÓN EUROPEA

FONDO EUROPEO DE DESARROLLO REGIONAL

"Una manera de hacer Europa"

AGENCIA ESTATAL DE INVESTIGACIÓN

Physics Motivation

From oscillations to event classification

The NMO signature lives in the difference between track-like and shower-like events.

Three key tasks for ML

- **Signal vs background:** Separate atmospheric ν from the much larger atmospheric μ flux
- **Track vs shower:** Identify the interaction by event topology due to no flavor identification
- **Event vs Noise:** Discriminate events from natural noise

Why this matters

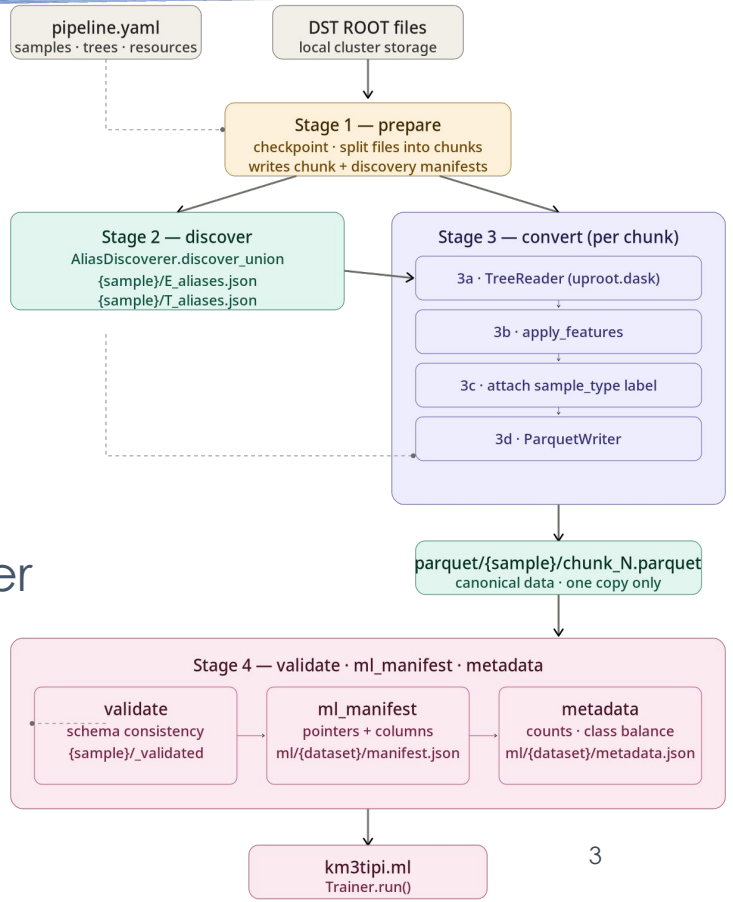
Reconstruction errors, mis-IDs, and selection efficiency impact the analysis. Better classification → tighter NMO constraints.

Introduction to KM3TPI

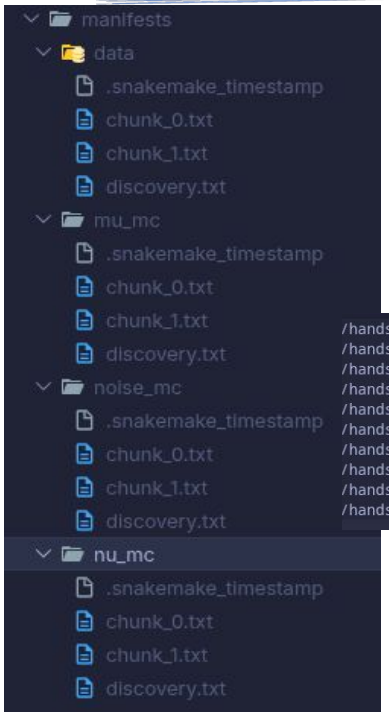
KM3net Tree-based Particle Identification

Idea Board:

- One repo to rule them all!
  - Data converter
  - Machine learning
- Agnostic approach (robust to changes)
- Modular design (km3tpi module)
  - preprocess (subpackage)
  - ml
- Snakemake compatible
  - Project maintained by one developer
  - User friendly for task delegation
- Environment handling
  - Conda
  - Singularity
- Testing & CI/CD

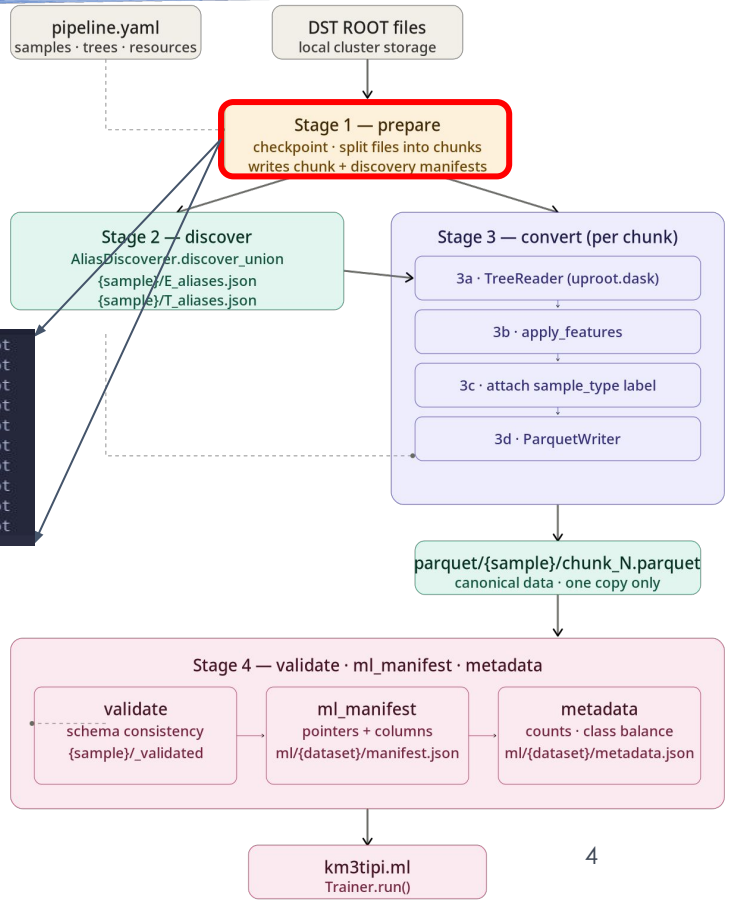


KM3TPI - Preprocess



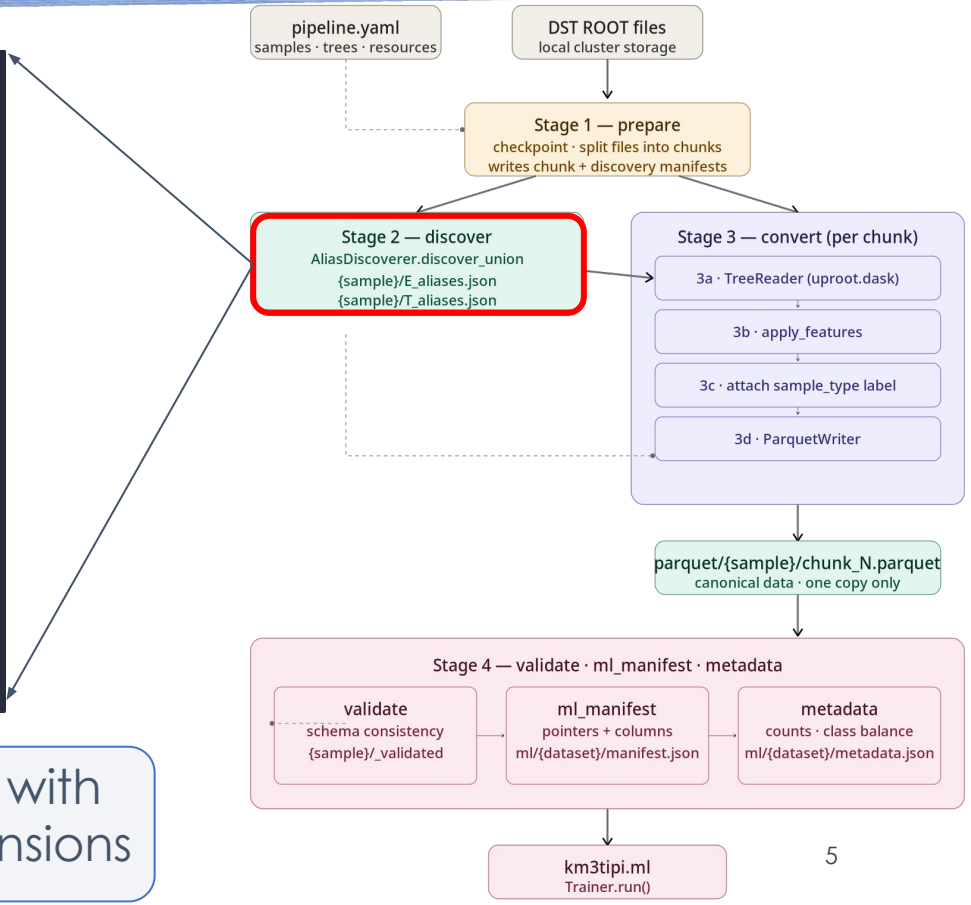

```
/hands-on-session/KM3NeT_00000117_00013853.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013859.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013868.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013871.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013877.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013883.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013906.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013929.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013948.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
/hands-on-session/KM3NeT_00000117_00013960.data.jppmuon_jppshower_dynamic.offline.dst.v9.2.root
```

Split files from the chosen directory into sublists for chunk creation



KM3TPI - Preprocess


```
{
  "header": {
    "filename": "
examples/data/KM3NeT_00000117_00013762.mc.gsg_neutrinos.km3sim_sirene_merged.jterbr.jppmuon_jppshower-upgoing_static.offline.dst.v9.0.root
",
    "branch": "T",
    "skip_fields": [
      "hits"
    ],
    "filter_branch": "get_lowest",
    "filter_typename": "/^(?!.*(TObject)).*/"
  },
  "ntrks": {
    "location": "ntrks",
    "index": [],
    "preview": "12",
    "type": "int32"
  },
  "Etot": {
    "location": "Etot",
    "index": [],
    "preview": "135.60697475975138",
    "type": "float64"
  },
  "Emax": {
    "location": "Emax",
    "index": [],
    "preview": "78.3184118121905",
    "type": "float64"
  },
  "Evis": {
    "location": "Evis",
    "index": [],
    "preview": "135.60697475975138",
    "type": "float64"
  },}
```

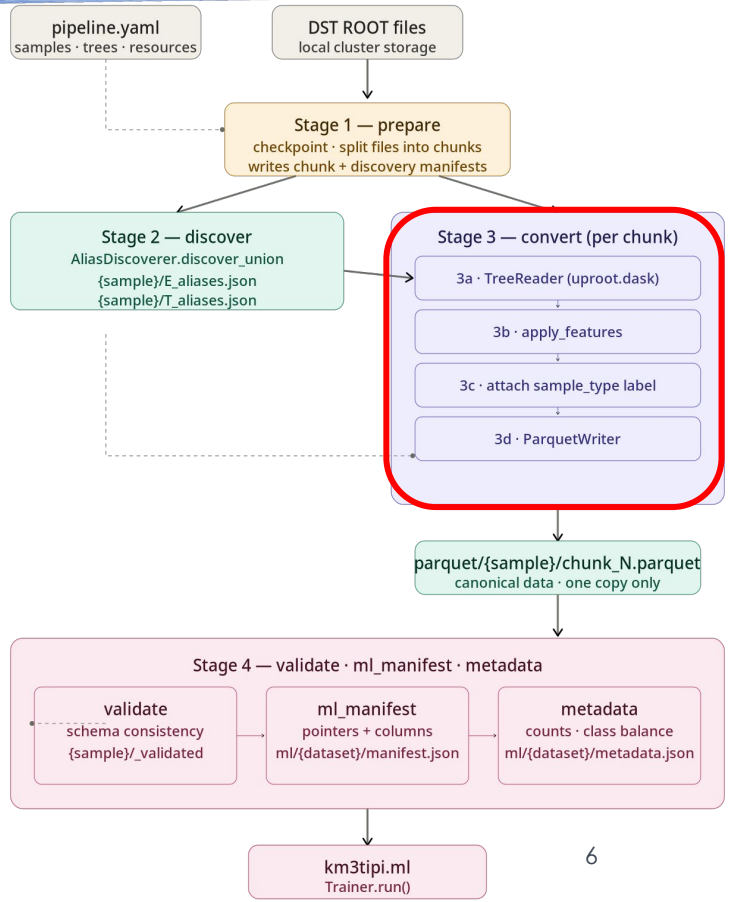
Mapping of the file's inner structure with **uproot** per tree (E,T) & possible extensions

KM3TPI - Preprocess

Stage 3 — Convert (per chunk)

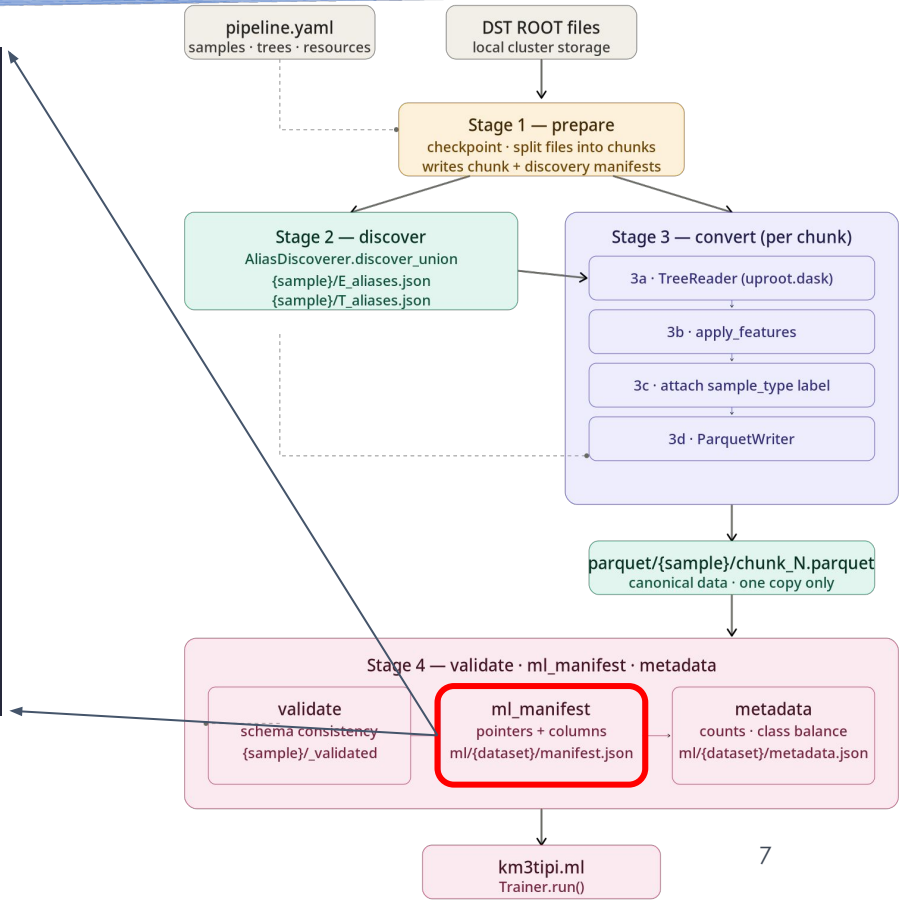
- One fused dask graph: read → features → label → write
- Lazy by default, only the branches in the alias JSON ever touch disk
- Memory bounded per tree, not per chunk
- One canonical Parquet per chunk

Still active investigation to reduce the **RAM** request for each parallel job

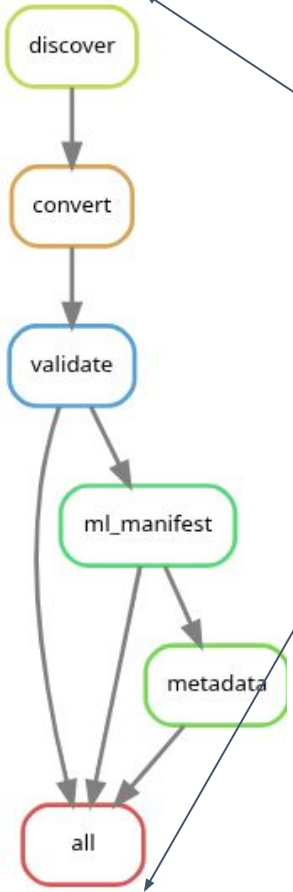


KM3TPI - Preprocess


```
{
  "dataset": "nu_vs_mu",
  "description": "Neutrino vs atmospheric muon classifier",
  "samples": {
    "nu_mc": {
      "path": "results/hands-on-session/parquet/nu_mc",
      "n_chunks": 2,
      "n_events": 31006,
      "all_columns": [
      ],
      "sample_type": 2
    }
  },
  "columns": [
  ],
  "n_columns": 130,
  "n_total_events": 2022386,
  "column_strategy": "common"
}
```

KM3TPI - Snakemake Preprocess




```
Job stats:
job              count
-----------  -------
all                  1
metadata             2
ml_manifest          2
prepare              4
validate             4
total               13
```

Prepare = checkpoint:
inspects data, then expands jobs

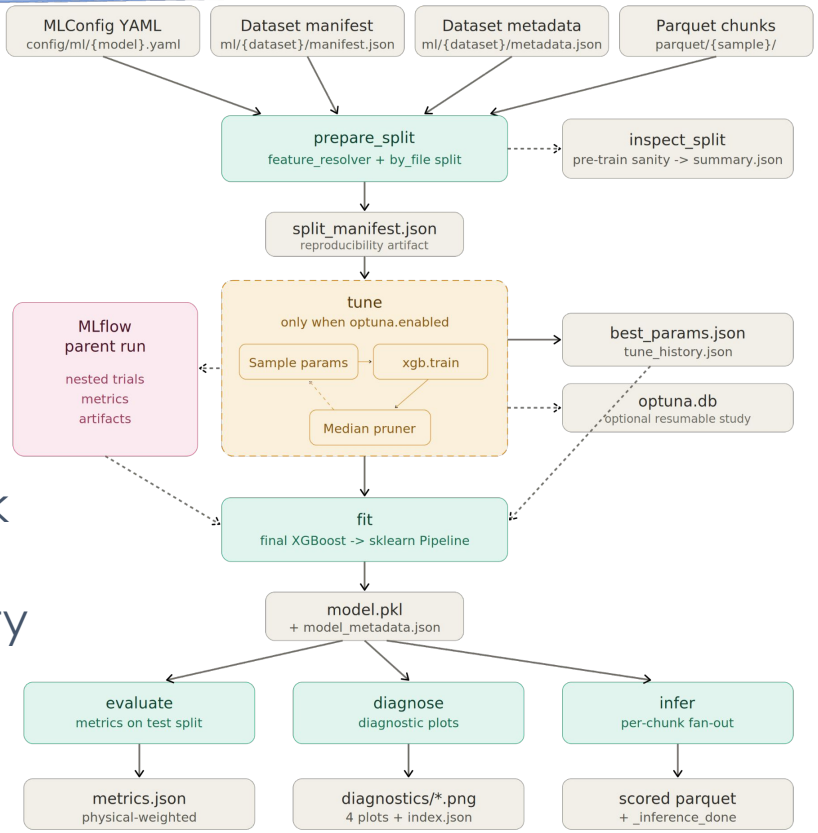
Introduction to KM3TPI - Machine Learning

Selected archetype:

- MLProcessor: main entrypoint
- Trainer: training & optimisation

Design choices:

- Schema versioning (metadata)
- Split by file & row
- Optimisation with Optuna & callback failsafes
- Class balancing: weights & multiplicity
- Deracator based cuts
- Monitoring with MLFlow



ML Snakemake Pipeline

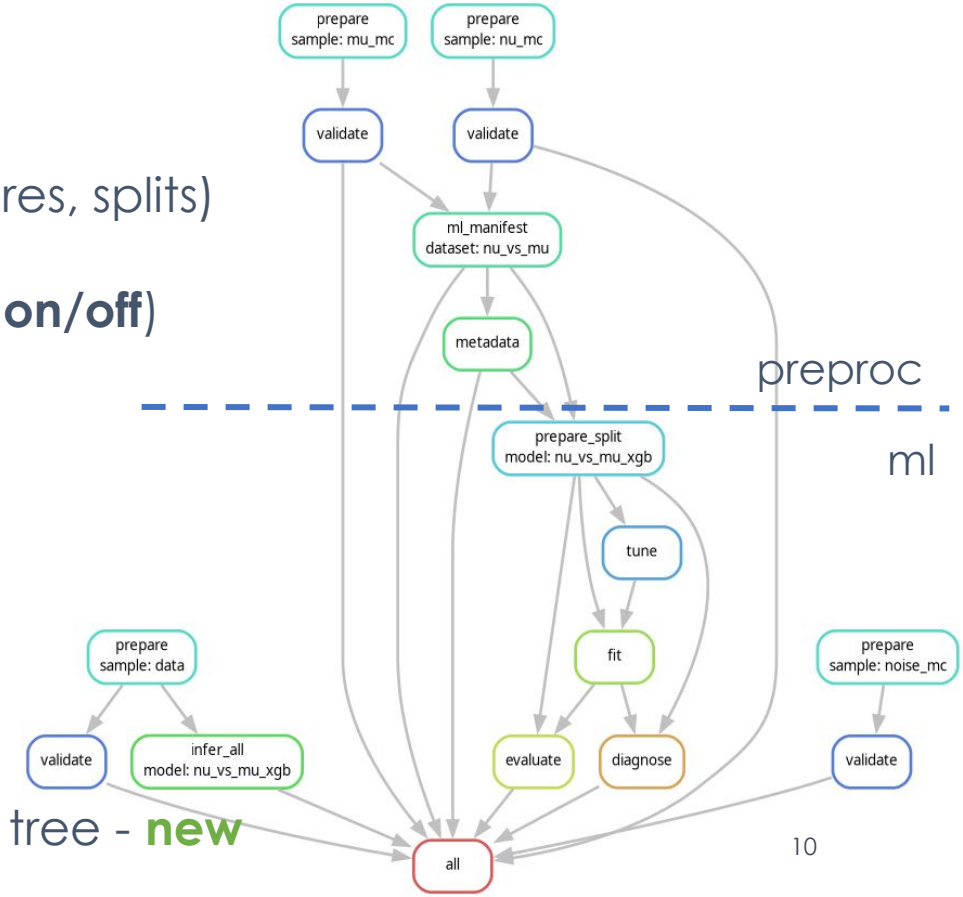
branch: features/ml-pipeline

Steps:

1. data processing (filtering, features, splits)
2. sample inspection (train) - **new**
3. hyper-parameter optimisation (**on/off**)
4. model final training/fitting
5. model evaluation
6. diagnostic plot checks
7. dataset inference

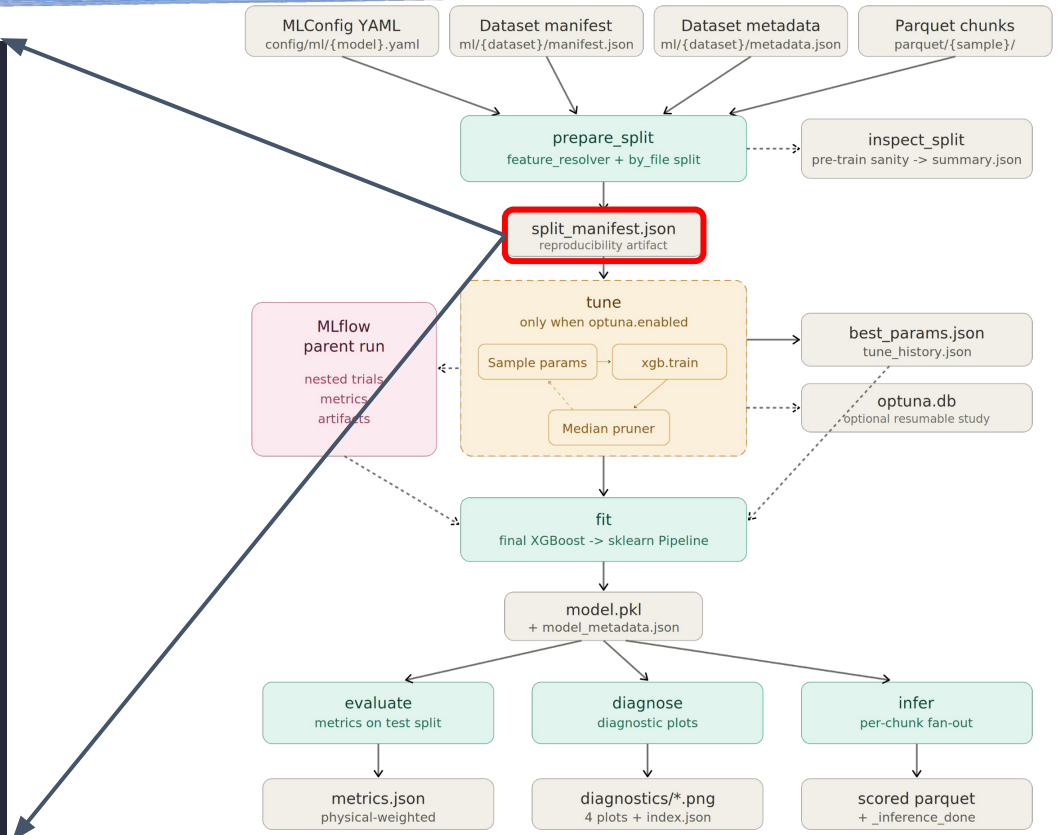
Inference steps:

- Applied to parquet chunks
- Create lightweight ROOT score tree - **new**



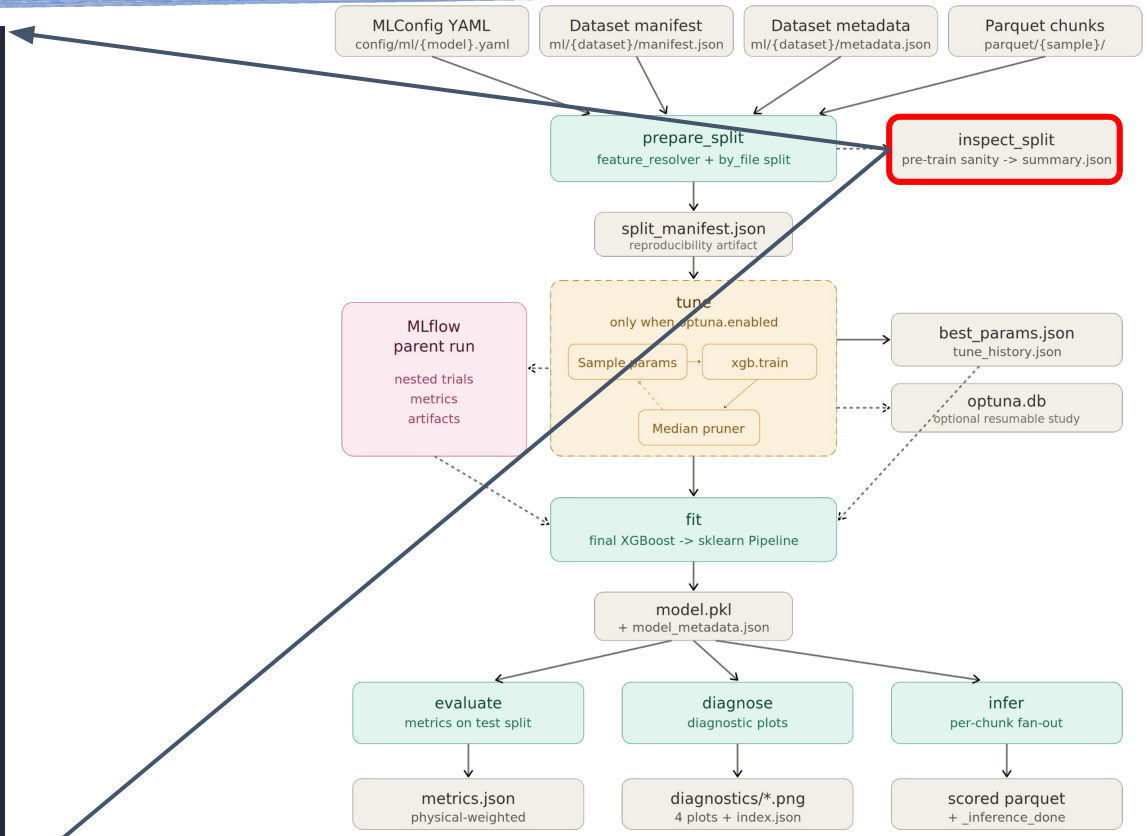
KM3TPI - Machine Learning


```
{
  "model": "nu_vs_mu_xgb",
  "dataset_ref": "nu_vs_mu",
  "dataset_manifest": "
/media/SteamML/km3tpi/pipeline/results/hands-on-session/ml/nu_vs_mu/manifest.json",
  "dataset_metadata": "
/media/SteamML/km3tpi/pipeline/results/hands-on-session/ml/nu_vs_mu/metadata.json",
  "target": "sample_type",
  "weight_column": "",
  "features": [
    "gandalf_Qup",
    "gandalf_Qdn",
    "gandalf_nHits",
    "QupOvernHits",
    "trks.E_0",
    "trks.E_1",
    "trks.dir.z_0",
    "trks.dir.z_1"
  ],
  "n_features": 8,
  "cuts": [],
  "cuts_strict": false,
  "cut_columns": [],
  "metadata_columns": [
    "mc_trks.E_0",
    "mc_trks.dir.z_0",
    "sample_type"
  ],
  "fill_value": null,
  "split": {
    "strategy": "by_file",
    "test_size": 0.2,
    "val_size": 0.2,
    "seed": 42
  },}
```

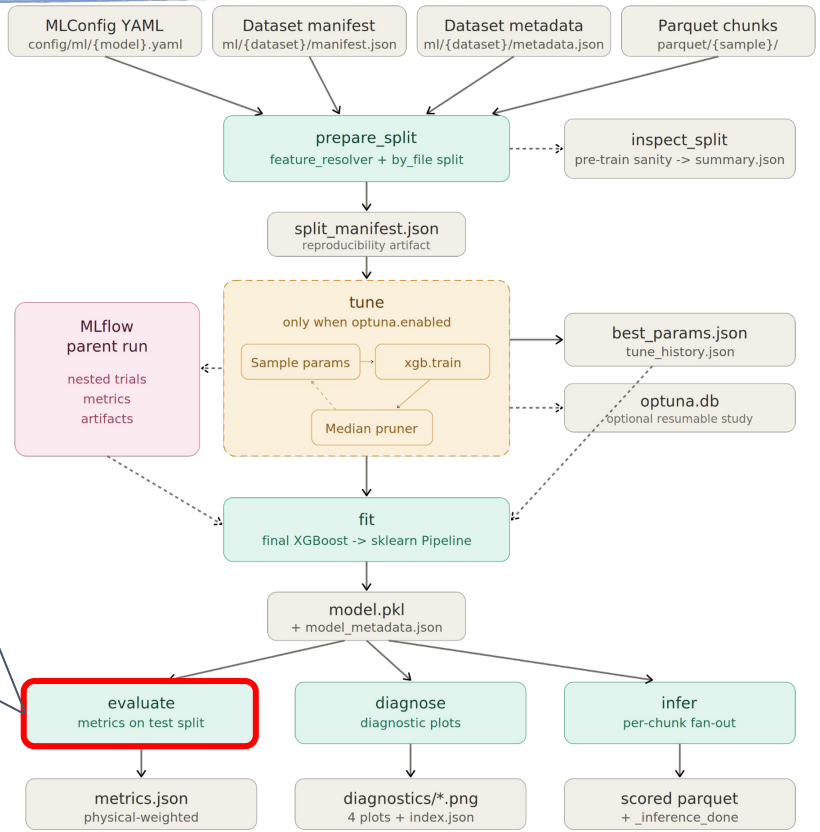
KM3TPI - Machine Learning


```
{
  "feature_count": 8,
  "feature_names": [
    "gandalf_Qup",
    "gandalf_Qdn",
    "gandalf_nHits",
    "QupOvernHits",
    "trks.E_0",
    "trks.E_1",
    "trks.dir.z_0",
    "trks.dir.z_1"
  ],
  "class_mapping": {
    "0": "1",
    "1": "2"
  },
  "diagnostics_primary_split": "train",
  "splits": {
    "train": {
      "n_events": 626133,
      "n_features": 8,
      "class_counts": {
        "0": 11193,
        "1": 614940
      },
      "weight_stats": null,
      "missingness_overall": 1.9963809605946342e-07
    },
    "val": {
    },
    "test": {
    }
  },
  "metadata_columns": {
    "train": [
      "mc_trks.E_0",
      "mc_trks.dir.z_0",
      "sample_type"
    ],
    "val": [],
    "test": []
  },
  "plots_saved": [
  ]
}
```

KM3TPI - Machine Learning


```
{
  "model_dir": "results/hands-on-session/ml/models/nu_vs_mu_xgb",
  "split_manifest": "results/hands-on-session/ml/models/nu_vs_mu_xgb/split_manifest.json",
  "splits_evaluated": [
    "val",
    "test"
  ],
  "n_events": {
    "val": 224174,
    "test": 213809
  },
  "metrics": {
    "val.accuracy": 0.9328557281397486,
    "val.roc_auc": 0.947216047490649,
    "val.log_loss": 0.3737554297303193,
    "test.accuracy": 0.9297597388323223,
    "test.roc_auc": 0.941207980099978,
    "test.log_loss": 0.37822044287189355
  },
  "objective": "binary:logistic",
  "class_mapping": {
    "0": 1,
    "1": 2
  }
}
```

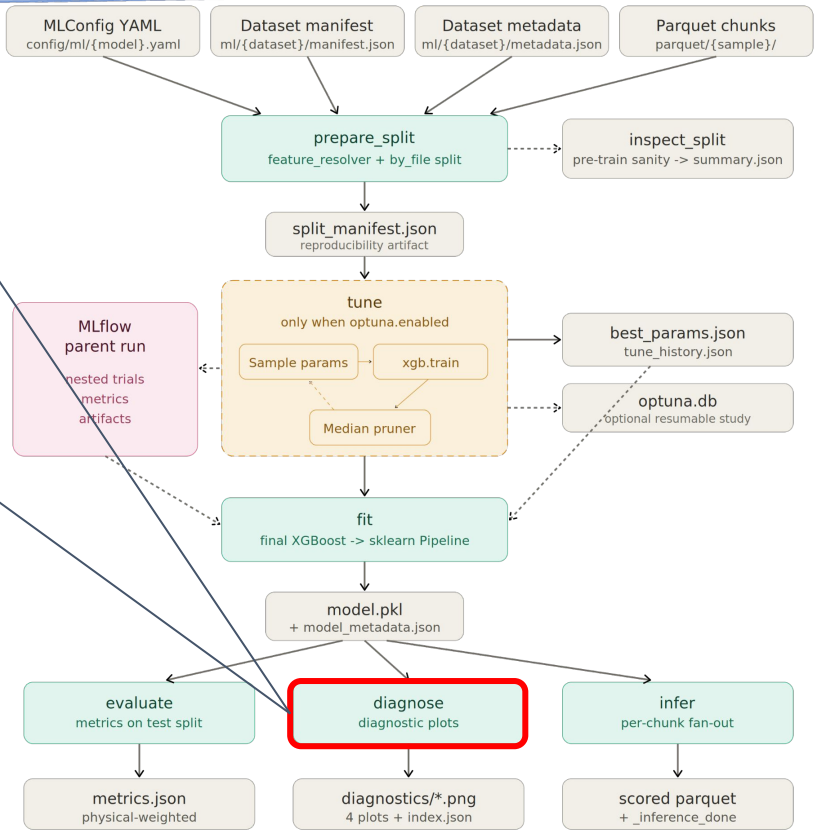



Evaluate the test and validation dataset on a set of chosen metics

KM3TPI - Machine Learning


```
{
  "model_dir": "results/hands-on-session/ml/models/nu_vs_mu_xgb",
  "split": "test",
  "n_events": 213809,
  "plots": {
    "score_histogram": "results/hands-on-session/ml/models/nu_vs_mu_xgb/diagnostics/score_histogram.png",
    "roc_curve": "results/hands-on-session/ml/models/nu_vs_mu_xgb/diagnostics/roc_curve.png",
    "confusion_matrix": "results/hands-on-session/ml/models/nu_vs_mu_xgb/diagnostics/confusion_matrix.png",
    "feature_importance": "results/hands-on-session/ml/models/nu_vs_mu_xgb/diagnostics/feature_importance.png"
  },
  "created_at": "2026-05-29T11:14:35+00:00"
}
```


Plots for a “visual” inspection of the performance for a requested split dataset



ML Snakemake Pipeline - Check

```yaml
dataset_ref: "nu_vs_mu"

dataset:
  features:
    - gandalf_Qup
    - gandalf_Qdn
    - gandalf_nHits
    - QupOvernHits
    - trks.E_0
    - trks.E_1
    - trks.dir.z_0
    - trks.dir.z_1
  target: "sample_type"
  weight_column: ""
  exclude_samples: ["data"]
  cuts: []
  cuts_strict: false
  fill_value: null

split:
  strategy: "by_file"
  test_size: 0.2
  val_size: 0.2
  seed: 42

model:
  name: "xgboost"
  objective: "binary:logistic"
  class_weight: "balanced"
  use_imputer: true
  imputer_strategy: "median"
  early_stopping_rounds: 30
  xgb_params:
    tree_method: "hist"
    n_estimators: 20
    max_depth: 6
    learning_rate: 0.05
    subsample: 0.8
    colsample_bytree: 0.8
    min_child_weight: 1.0
    reg_alpha: 0.0
    reg_lambda: 1.0
    random_state: 42
```

Initial pass through checks with hands-on-session dataset

Statistics (trigger level):

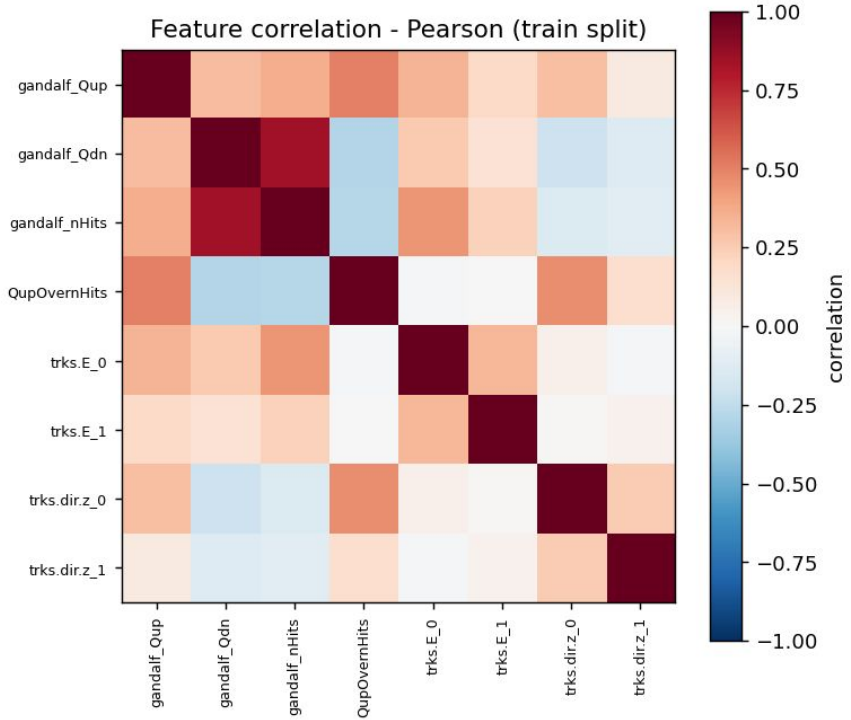
- Neutrinos: 18k
- Muons: 1M

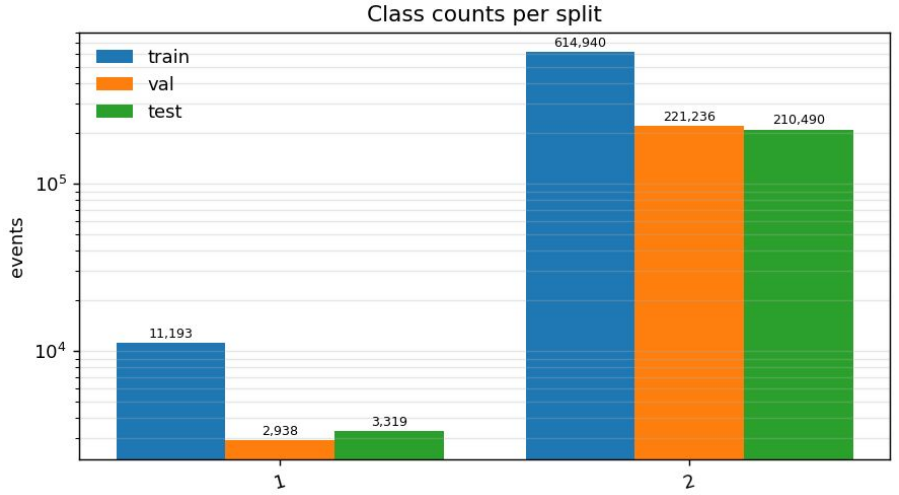
Sample balance ratio: ~ 1:60

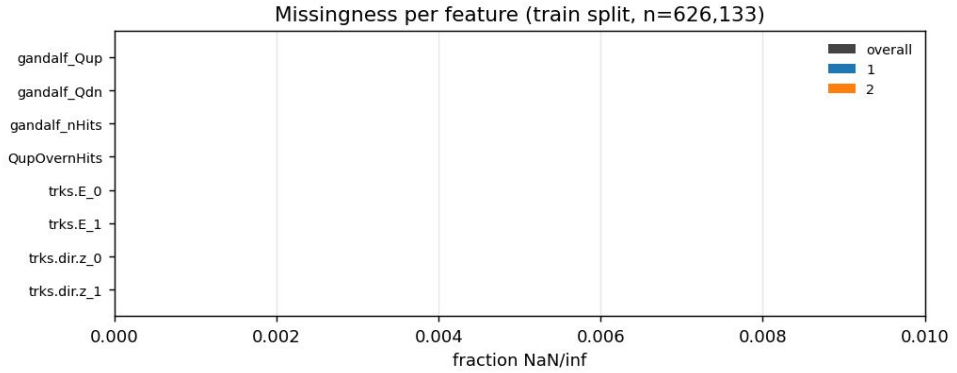
- Just a sanity check!


```json
"assignment": {
  "mu_mc": {
    "test": [
      "chunk_4.parquet"
    ],
    "val": [
      "chunk_2.parquet"
    ],
    "train": [
      "chunk_3.parquet",
      "chunk_1.parquet",
      "chunk_0.parquet"
    ]
  },
  "nu_mc": {
    "test": [
      "chunk_3.parquet"
    ],
    "val": [
      "chunk_0.parquet"
    ],
    "train": [
      "chunk_1.parquet",
      "chunk_2.parquet",
      "chunk_4.parquet"
    ]
```

ML Snakemake Pipeline - Check Training Sample

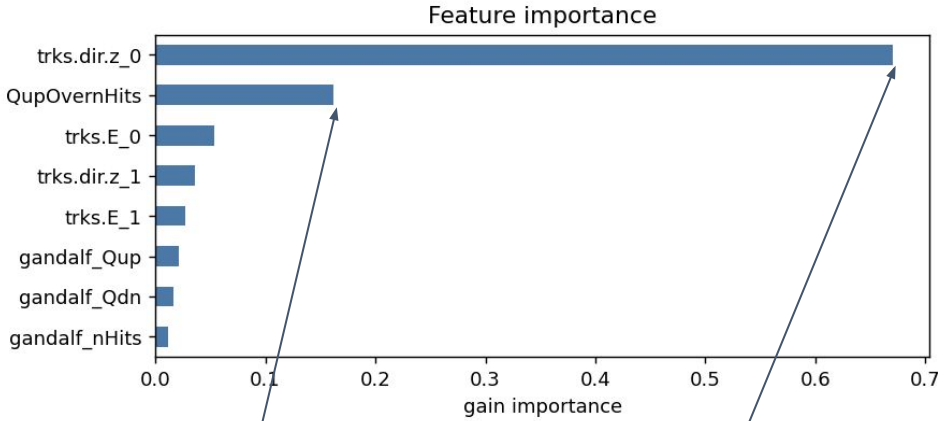






ML Snakemake Pipeline - Evaluation

YJ: TPR-FPR



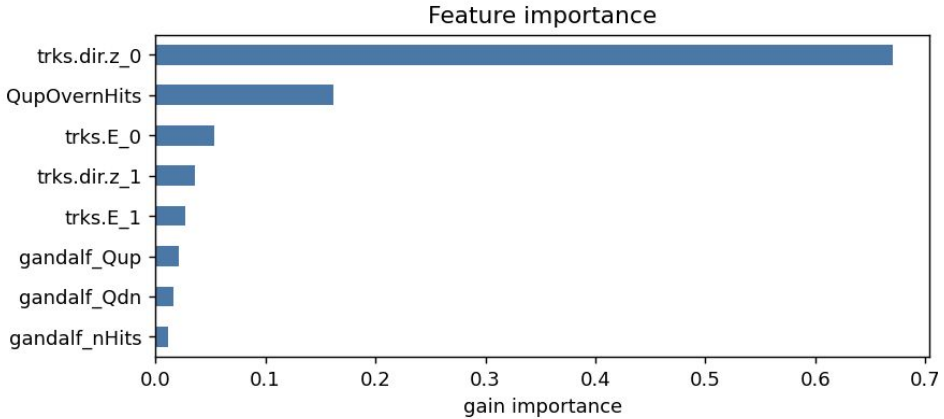
Direction related feature as well

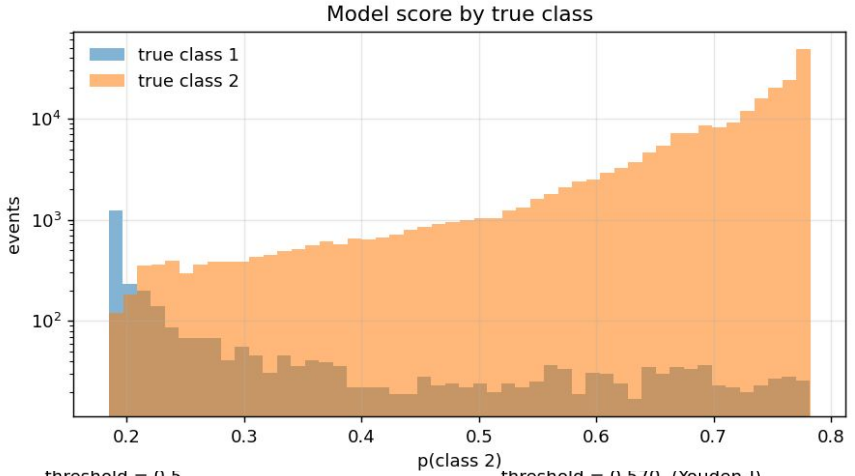
Since no cut in direction, classifier learns from zenith angle under track hypothesis

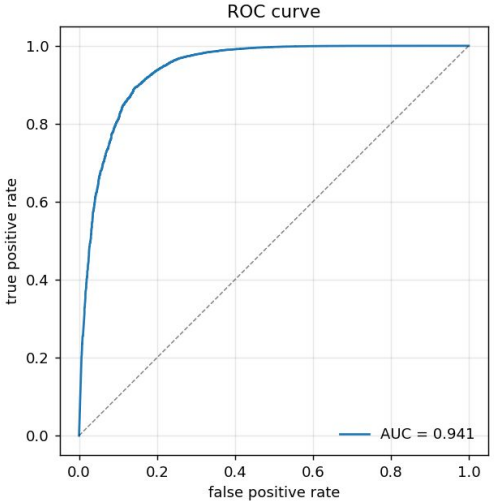
Should implemented both weighted & the PDF here

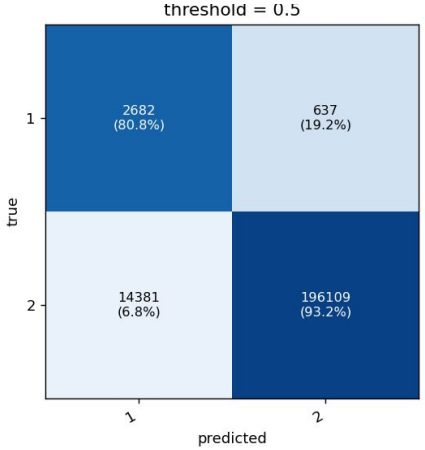
ML Snakemake Pipeline - Evaluation

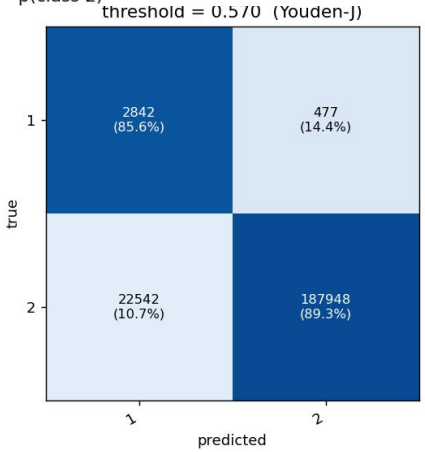
YJ: TPR-FPR











Summary & Next Steps

Current status:

- Preprocessing Module v1 ✅
- Preprocessing Pipeline Rules ✅
- ML Module v1 ✅
- ML Pipeline Rules ✅
- Documentation for both ✅
- Environment versioning (conda, pip) ✅
- CI/CD & tests (cov: 71%) ✅
- Demo notebooks ✅

Next Steps with high priority:

- Singularity container for full compatibility ❌
- Continue pipeline testing ⏳
- Feature engineering & cut selection ⏳

✅ Finalized
❌ Not started
⏳ Active

Summary & Next Steps

Long term objectives:

- Baseline classifier: track vs shower on νμ CC / νe CC
- First end-to-end run on full ORCA MC production
- Check extension to ARCA
- Integrate it to existing collaboration workflows
- Integrate dedicated monitoring with MLflow
- First official checks with ORCA23/24 mass production
- Extensive documentation for task delegation and usage for members not too familiar with Snakemake

Check out git repo & hands-on session…

Back-Up Slides

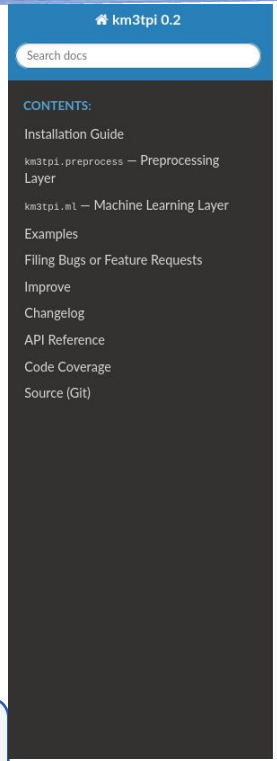
Documentation

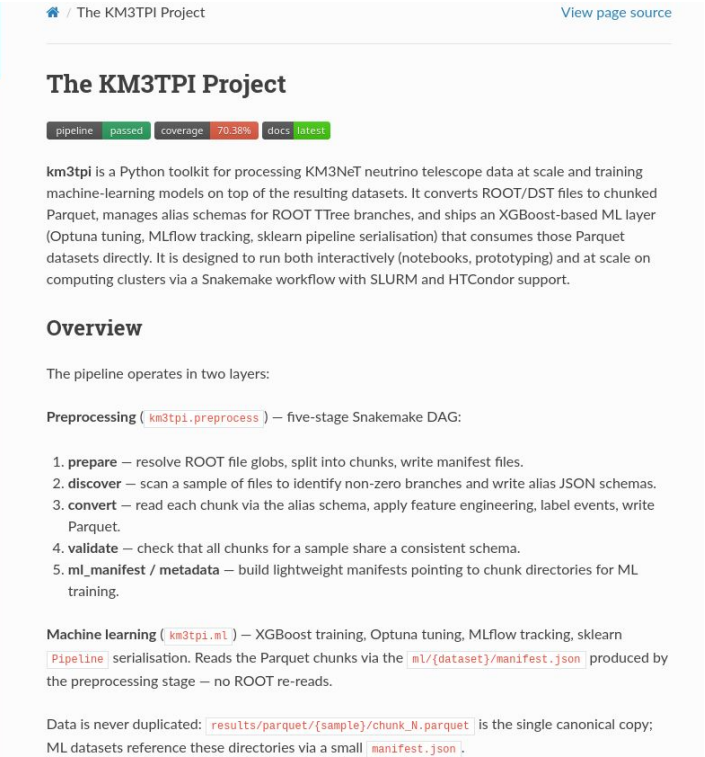
Check latest docs here

Chapters:

- Installation
- Preprocessing Layer
- ML Layer
- Examples
- API ref

To be continually updated with the latest tagged version pipeline job





Developer note

If the doc link is not working, check it via the project because it might have been transferred.

User Configuration - Preprocessing

```yaml
detector_id: "hands-on-session"
base_dir: "/mnt/SteamML/Datasets/KM3NeT/ml_training"
trees: ["E", "T"]

samples:
  data:
    root_files: "{base_dir}/{detector_id}/*.root"
    identifiers: ["data"]
    skip_fields: ["hits", "mc_trks"]
    sample_type: "data"
    n_jobs: 5

  nu_mc:
  mu_mc:
  noise_mc:

discovery:
  n_preview_files: 4
  sampling_strategy: "spread" # "first" | "spread" | "random"
  seed: 42

features:
  E: []
  T: []
```

```yaml
dask:
  scheduler: "synchronous" # "synchronous" | "threads" | "processes"
  n_workers: 1
  threads_per_worker: 1

writer:
  row_group_size: 250000
  extensionarray: false
  log_memory: true
```

- how to export works?
- define folder path & classes
- which trees to map & how?
- do I want to add new features?

User Configuration - Machine Learning

Dataset & split strategy

```yaml
dataset_ref: "nu_vs_mu"

dataset:
  features:
    - gandalf_Qup
    - gandalf_Qdn
    - gandalf_nHits
    - QupOvernHits
    - trks.E_0
    - trks.E_1
    - trks.dir.z_0
    - trks.dir.z_1
  target: "sample_type"
  weight_column: ""
  exclude_samples: ["data"]
  cuts: []
  cuts_strict: false
  fill_value: null
  metadata_columns:
    - mc_trks.E_0 #
    - mc_trks.dir.z_0
    - sample_type

inspect:
  primary_split: "train"
  max_features: 0 # 0 = plot all 8 features
  energy_key: "mc_trks.E_0"
  zenith_key: "mc_trks.dir.z_0"
  sample_id_key: "sample_type"

split:
  strategy: "by_file"
  test_size: 0.2
  val_size: 0.2
  seed: 42
```

ML Dataset & Models

```yaml
ml:
  datasets:
    nu_vs_mu:
      samples: ["nu_mc", "mu_mc"]
      columns: "common"
      description: "Neutrino vs atmospheric muon classifier"

  models:
    nu_vs_mu_xgb: "config/ml/nu_vs_mu_xgb.yaml"
```

Model Parameters

```yaml
model:
  name: "xgboost"
  objective: "binary:logistic"
  class_weight: "balanced"
  use_imputer: true
  imputer_strategy: "median"
  early_stopping_rounds: 30
  xgb_params:
    tree_method: "hist"
    n_estimators: 20
    max_depth: 6
    learning_rate: 0.05
    subsample: 0.8
    colsample_bytree: 0.8
    min_child_weight: 1.0
    reg_alpha: 0.0
    reg_lambda: 1.0
    random_state: 42
```

Preprocessing benchmarks on whole ORCA13

```
[INFO]  Parsing samples from: config/pipeline.yaml
[INFO]  Scanning directory: /run/media/mchadolias/SteamML/Datasets/KM3NeT/ml_training/KM3NeT_00000117
[INFO]  Found 553 file(s) matching pattern '*'.

km3tipi — file count per sample
config : config/pipeline.yaml
dir    : /run/media/mchadolias/SteamML/Datasets/KM3NeT/ml_training/KM3NeT_00000117
pattern: *
```

SAMPLE	COUNT	SIZE	AVG/FILE	IDENTIFIERS
data	139	12.20 GB	89.90 MB	data
nu_mc	137	282.36 MB	2.06 MB	neutrinos,gsg_neutrinos
mu_mc	139	13.52 GB	99.65 MB	mupage,mupage_default
noise_mc	138	253.10 MB	1.83 MB	pure_noise
unclassified	0	0 B	—	(no matching identifier)
TOTAL	**553**	**26.25 GB**	**48.61 MB**	

```
run summary
  distinct runs             : 139
  full runs (all 4 samples) : 136  (97.8%)

  per-sample run coverage:
    √ data          :  139 covered,    0 missing
    ~ nu_mc         :  137 covered,    2 missing
    √ mu_mc         :  139 covered,    0 missing
    ~ noise_mc      :  138 covered,    1 missing
```

Successful execution the pipeline in work laptop in less than 5 minutes!

```
km3tipi monitor  — 2026-05-03 22:06:25
 config:  config/pipeline.yaml
 results: results/KM3NeT_00000117

pipeline  (done/total — = all done · * running · X has failed · . pending)
                        prepar  discov  conver  valida  ml_man  metada
 data                   1/1 =   2/2 =   7/8 .   1/1 =   -       -
 nu_mc                  1/1 =   2/2 =   5/5 =   1/1 =   -       -
 mu_mc                  1/1 =   2/2 =   8/8 =   1/1 =   -       -
 noise_mc               1/1 =   2/2 =   5/5 =   1/1 =   -       -

 nu_vs_mu               -       -       -       -       1/1 =   1/1 =
 signal_vs_background   -       -       -       -       1/1 =   1/1 =

scheduler  (none, user=mchadolias)
 no scheduler available — pipeline view only
```

Snakemake Guide

- **Workflow management system**

- Snakemake is designed such that:
  - **Reproducibility** is easily implemented
  - Scalable from laptop to **HPC cluster**
  - Good practices with little effort (logging, benchmarks)

- Quick installation with conda/mamba
- Check more here for documentation & tutorials [1,2,3]
- Most important file in a Snakemake workflow is the **Snakefile**
- A rule describes how to get one or more output files from input files

```
rule discover:
    input:
        manifest = str(MANIFEST_DIR / "{sample}" / "discovery.txt"),
    output:
        alias_json = str(ALIAS_DIR / "{sample}" / "{tree}_aliases.json"),
    params:
        n_preview   = config["discovery"]["n_preview_files"],
        sampling    = config["discovery"].get("sampling_strategy", "spread"),
        seed        = config["discovery"].get("seed", 42),
        skip_fields = lambda wc: " ".join(
            config["samples"][wc.sample].get("skip_fields", ["hits"])
        ),
    log:
        str(LOG_DIR / "discover" / "{sample}_{tree}.log"),
    benchmark:
        str(BENCHMARK_DIR / "discover" / "{sample}_{tree}.tsv")
    retries: 2
    shell:
        """
        mkdir -p $(dirname {output.alias_json}) $(dirname {log})

        python scripts/discover.py \
            --manifest    {input.manifest}       \
            --tree        {wildcards.tree}       \
            --sample      {wildcards.sample}     \
            --output      {output.alias_json}    \
            --n-preview   {params.n_preview}     \
            --sampling    {params.sampling}       \
            --seed        {params.seed}           \
            --skip-fields {params.skip_fields}   \
        > {log} 2>&1
        """
```

Snakemake Guide

- The user requests Snakemake to produce files via the command line
- The wildcards “sample” & “tree” has to be filled with a concrete value (in this example a data & T)

```
rule discover:
    input: results/hands-on-session/manifests/data/discovery.txt
    output: results/hands-on-session/aliases/data/T_aliases.json
    log: results/hands-on-session/logs/discover/data_T.log
    jobid: 19
    benchmark: results/hands-on-session/benchmarks/discover/data_T.tsv
    reason: Missing output files: results/hands-on-session/aliases/data/T_aliases.json
    wildcards: sample=data, tree=T
    resources: tmpdir=/tmp, disk_mb=1000, disk=1 GB, disk_mib=954, mem_mb=2000, mem=2 GB, mem_mib=1908, runtime=15
```

- A second request to produce the file is not executed since:

```
rule discover:
    input:
        manifest = str(MANIFEST_DIR / "{sample}" / "discovery.txt"),
    output:
        alias_json = str(ALIAS_DIR / "{sample}" / "{tree}_aliases.json"),
    params:
        n_preview   = config["discovery"]["n_preview_files"],
        sampling    = config["discovery"].get("sampling_strategy", "spread"),
        seed        = config["discovery"].get("seed", 42),
        skip_fields = lambda wc: " ".join(
            config["samples"][wc.sample].get("skip_fields", ["hits"])
        ),
    log:
        str(LOG_DIR / "discover" / "{sample}_{tree}.log"),
    benchmark:
        str(BENCHMARK_DIR / "discover" / "{sample}_{tree}.tsv")
    retries: 2
    shell:
        """
        mkdir -p $(dirname {output.alias_json}) $(dirname {log})

        python scripts/discover.py \
            --manifest    {input.manifest}       \
            --tree        {wildcards.tree}       \
            --sample      {wildcards.sample}     \
            --output      {output.alias_json}    \
            --n-preview   {params.n_preview}     \
            --sampling    {params.sampling}       \
            --seed        {params.seed}           \
            --skip-fields {params.skip_fields}   \
        > {log} 2>&1
        """
```

Run Preprocess Notebook

Check the *demo_processing.ipynb*

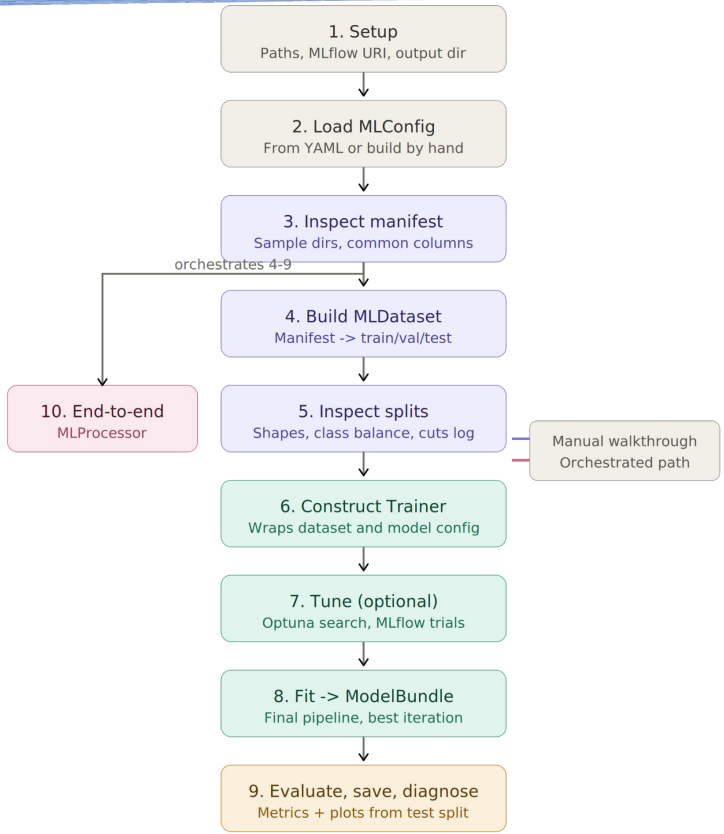
```
1. Setup — paths, working dir, optional pointer to real data
2. Synthesise a tiny ROOT file (skip if you point `DATA_DIR` at real ones)
3. File discovery — `find_files`, `get_files_by_identifier`
4. Alias discovery — `AliasDiscoverer` + `AliasStore`
5. Read with `TreeReader` (lazy dask-awkward)
6. Apply registered features
7. Add the `sample_type` label and write Parquet
8. Validate the chunk
9. Build a minimal ML manifest by hand
10. End-to-end with `DSTProcessor`

Self-contained: synthesises ROOT files if none are provided.
```

If you want to use a subset of the provided DSTs

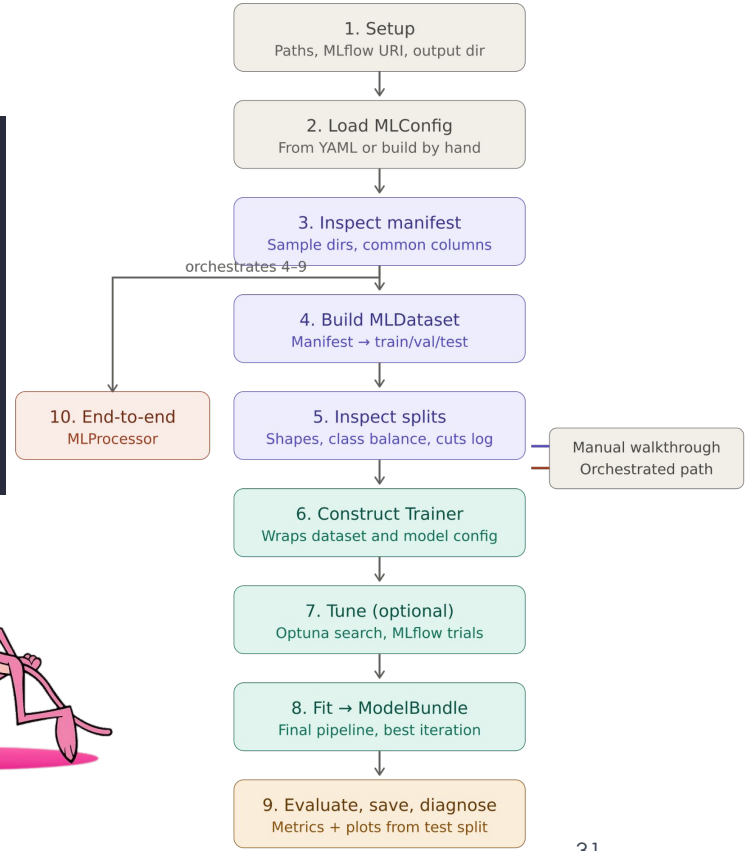
```
WORK = Path("./_preprocess_demo").resolve()
if WORK.exists():
    shutil.rmtree(WORK)
WORK.mkdir(parents=True)

# If you have a real small dataset, set DATA_DIR to its directory and
# the synthesis step below will be skipped.
DATA_DIR: Path | None = None    # e.g. Path("/sps/km3net/.../small_subset")
print("WORK     :", WORK)
print("DATA_DIR :", DATA_DIR)
```



Run Machine Learning Notebook

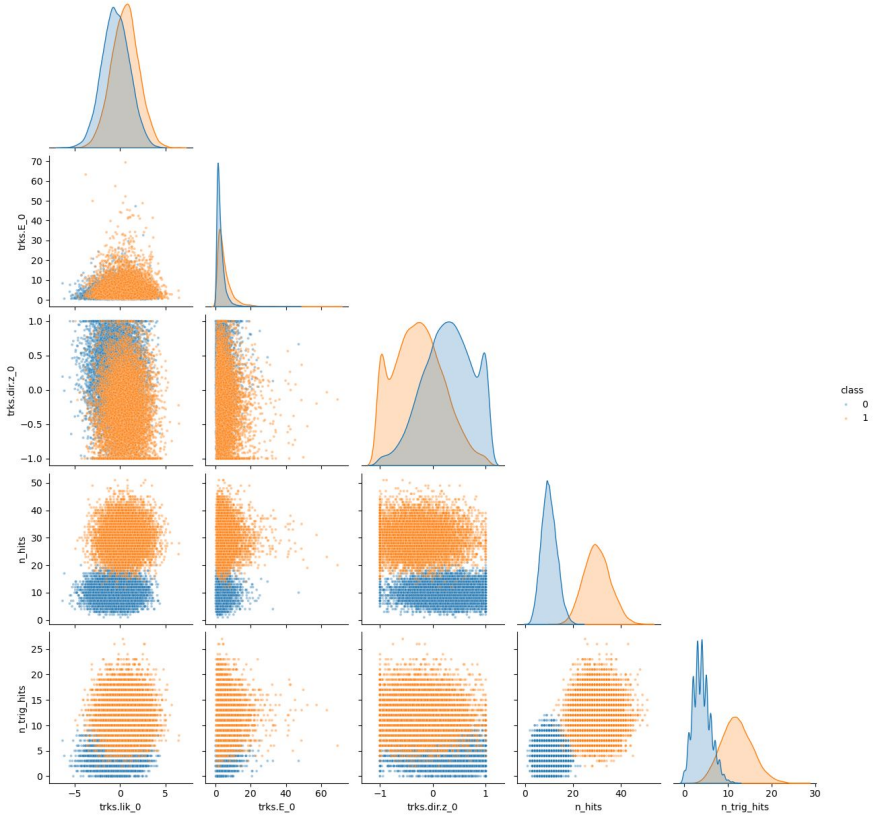
```
dataset_cfg = DatasetConfig(
    manifest_path=manifest_path,
    features=[
        "trks.lik_0", "trks.E_0", "trks.dir.z_0",
        "n_hits", "n_trig_hits",
    ],
    target="sample_type",
    weight_column="weight",          # set None to disable per-event weights
    exclude_samples=["data"],        # default — drops any sample with "data" in name
    cuts=[],                         # registry empty in M1
)
print(json.dumps(dataset_cfg.to_dict(), indent=2, default=str))
```



If you want to use a subset of the files that you produced in the previous, define the ML_MANIFEST

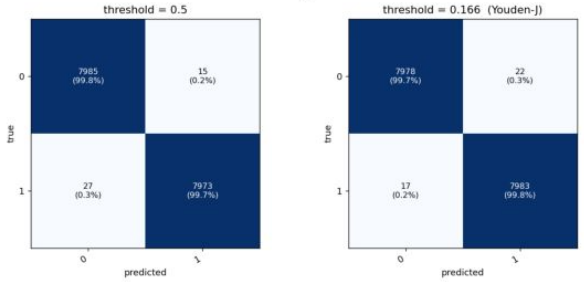
ML Test with synthetic data

- train/val/test datasets
- 16k events per set
- Class balance: 1:1
- 5 “features”:
  - “trks.lik_0"
  - "trks.E_0"
  - "trks.dir.z_0"
  - "n_hits"
  - "n_trig_hits"

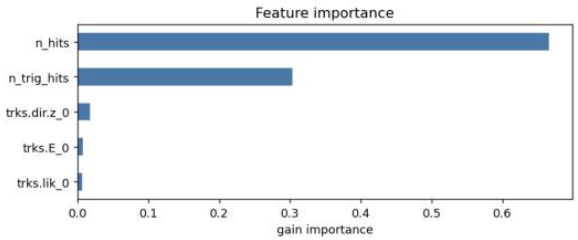


Diagnostics

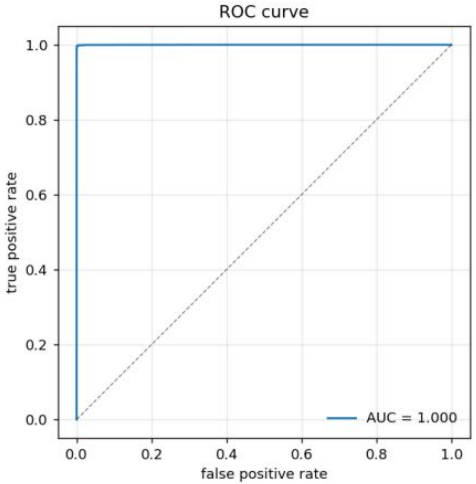
confusion_matrix



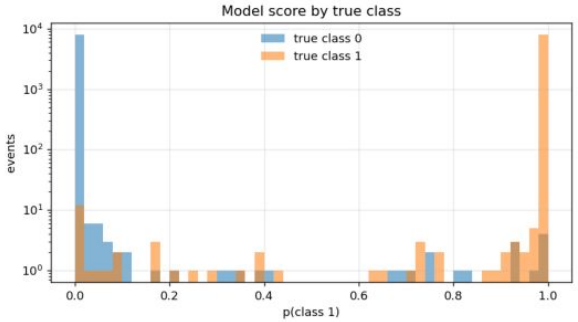
feature_importance



roc_curve



score_histogram




```
{
  "feature_names": [
    "trks.lik_0",
    "trks.E_0",
    "trks.dir.z_0",
    "n_hits",
    "n_trig_hits"
  ],
  "target": "sample_type",
  "objective": "binary:logistic",
  "num_classes": null,
  "class_mapping": {
    "0": 0,
    "1": 1
  },
  "xgb_params": {
    "tree_method": "hist",
    "n_estimators": 117,
    "max_depth": 5,
    "learning_rate": 0.22648248189516848,
    "n_jobs": 1,
    "random_state": 0,
    "subsample": 0.892797576724562,
    "colsample_bytree": 0.8394633936788146,
    "min_child_weight": 0.20513382630874505,
    "reg_alpha": 2.5348407664333426e-07,
    "reg_lambda": 3.3323645788192616e-08
  },
  "best_iteration": 116,
  "metric": "roc_auc",
  "metric_value": 0.9999489324103895,
  "km3tpi_version": null,
  "xgboost_version": "3.2.0",
  "sklearn_version": "1.8.0",
  "created_at": "2026-05-07T19:25:15+00:00",
  "training_manifest": "
/mnt/SteamML/Code/km3tpi/notebooks/jupyter_tests/ml/nu_vs_mu/manifest.js
on
",
  "extra": {}
}
```


Model Metadata