Cross Validation and Data Ingestion in TMVA

Kim Albertsson for the ROOT/TMVA team

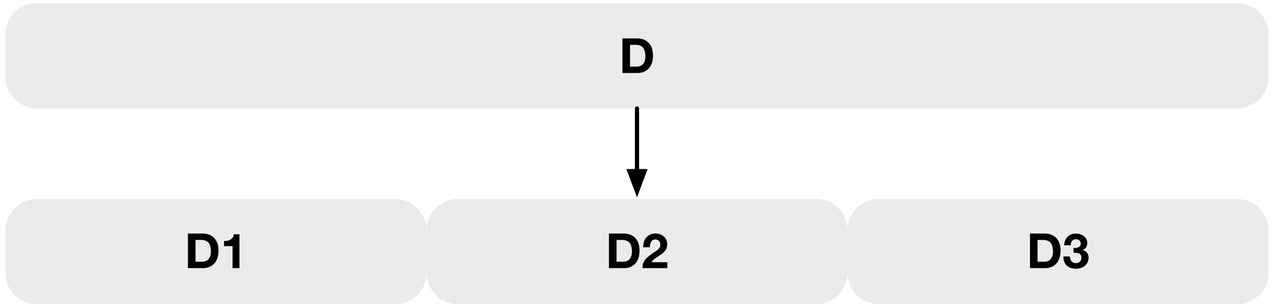
2018-09-13

Outline

- Cross validation improvements in ROOT 6.12 and 6.14
- Data ingestion with RDataFrame (in development)
- See Stefan's talk for long term evolution

Cross validation

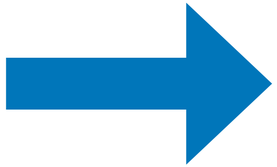
Overview



CV in TMVA

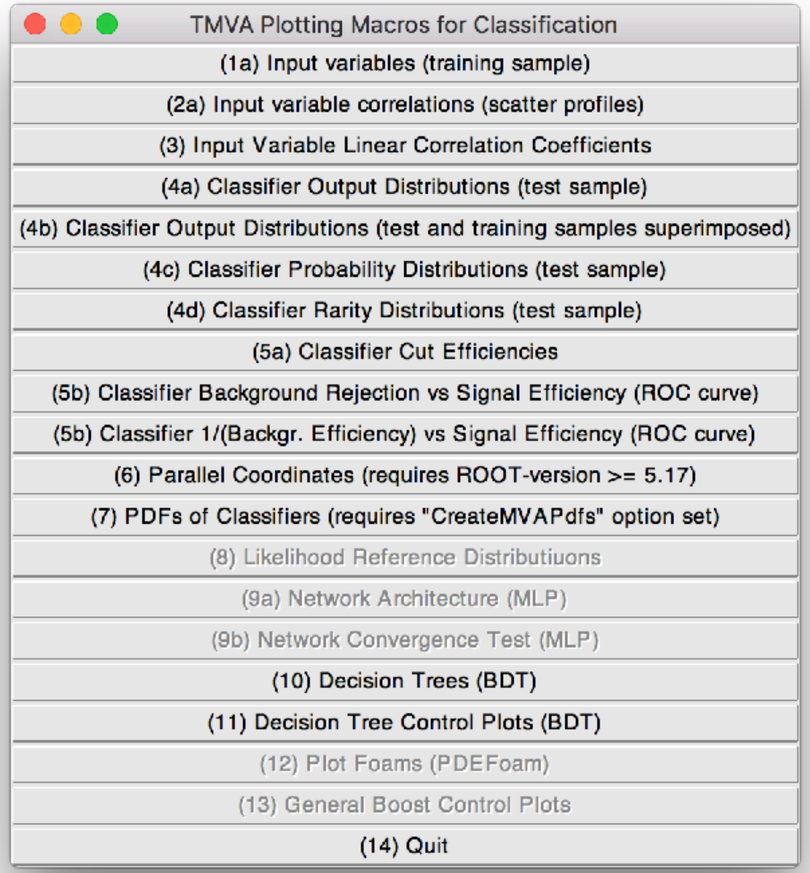
- Interface similar to Factory
- Per-fold information + models
- Integrates with TMVA analysis workflow

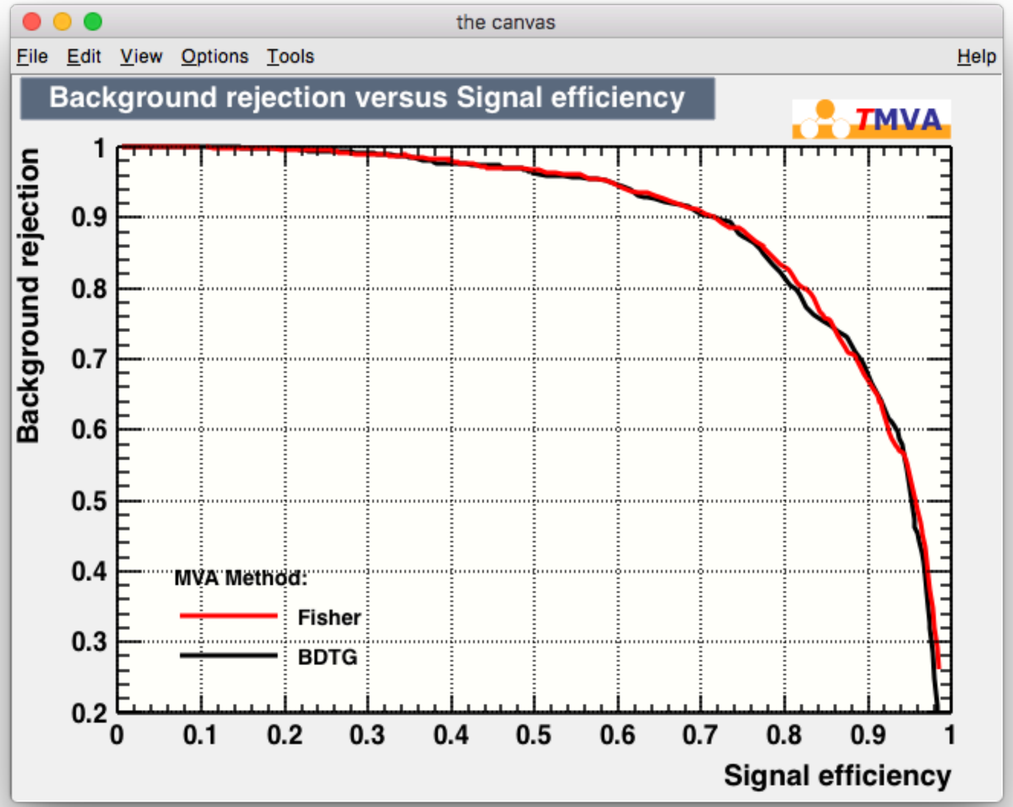
```
// ... snip ...

const char * opts =
“AnalysisType=Classification”;


TMVA::Factory factory{“<jobname>”,
    dataloader, outputFile, opts};

// ... snip ...
```



```
// ... snip ...

const char * opts =
“AnalysisType=Classification:”
“NumFolds=3”;


TMVA::CrossValidation cv{“<jobname>”,
    dataloader, outputFile, opts};

// ... snip ...
```

CV in TMVA

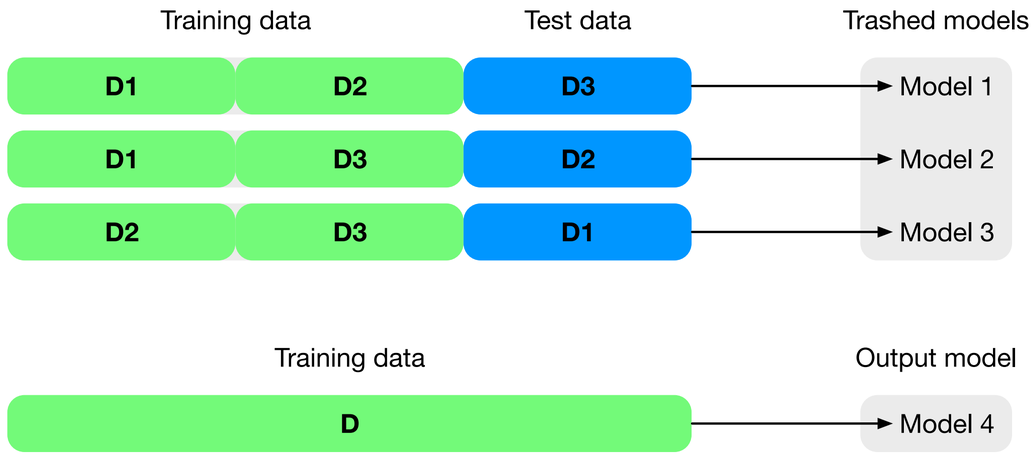

TMVA Plotting Macros for Classification
(1a) Input variables (training sample)
(2a) Input variable correlations (scatter profiles)
(3) Input Variable Linear Correlation Coefficients
(4a) Classifier Output Distributions (test sample)
(4b) Classifier Output Distributions (test and training samples superimposed)
(4c) Classifier Probability Distributions (test sample)
(4d) Classifier Rarity Distributions (test sample)
(5a) Classifier Cut Efficiencies
(5b) Classifier Background Rejection vs Signal Efficiency (ROC curve)
(5b) Classifier 1/(Backgr. Efficiency) vs Signal Efficiency (ROC curve)
(6) Parallel Coordinates (requires ROOT-version >= 5.17)
(7) PDFs of Classifiers (requires "CreateMVAPdfs" option set)
(8) Likelihood Reference Distributiuons
(9a) Network Architecture (MLP)
(9b) Network Convergence Test (MLP)
(10) Decision Trees (BDT)
(11) Decision Tree Control Plots (BDT)
(12) Plot Foams (PDEFoam)
(13) General Boost Control Plots
(14) Quit


the canvas
File Edit View Options Tools Help
Background rejection versus Signal efficiency
TMVA
Background rejection
Signal efficiency
MVA Method:
Fisher
BDTG
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

CV Workflow

Standard workflow

- Train K models
- Estimate performance (loss, distribution)
- Train new model using all data

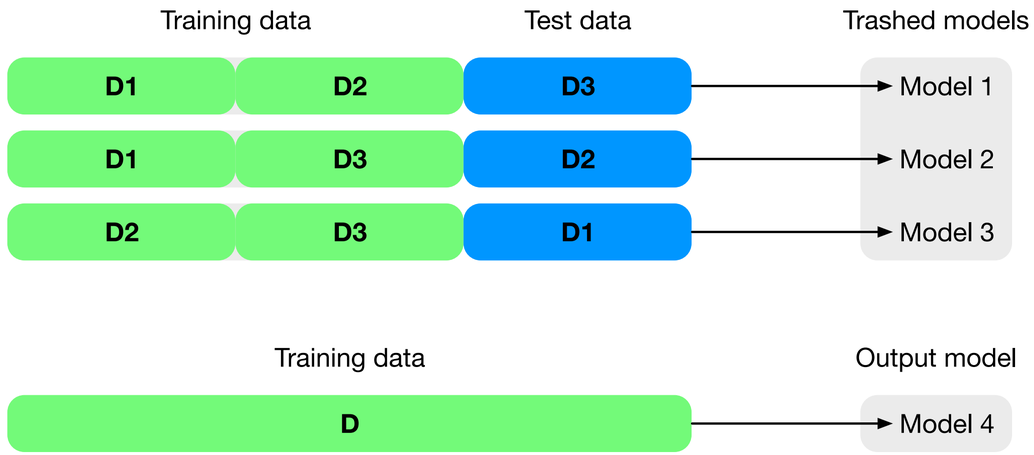


$$\textbf{Err} = \sum_{k=1}^{K} \ell\left(\hat{f}_k, \textbf{target}\right)$$

CV Workflow

Standard workflow

- Train K models
- Estimate performance (loss, distribution)
- Train new model using all data



Problem

- Estimates hold on *average*

$$\textbf{Err} = \mathrm{E}\left[\sum_{k=1}^{K} \ell\left(\hat{f}_k, \textbf{target}\right)\right]$$

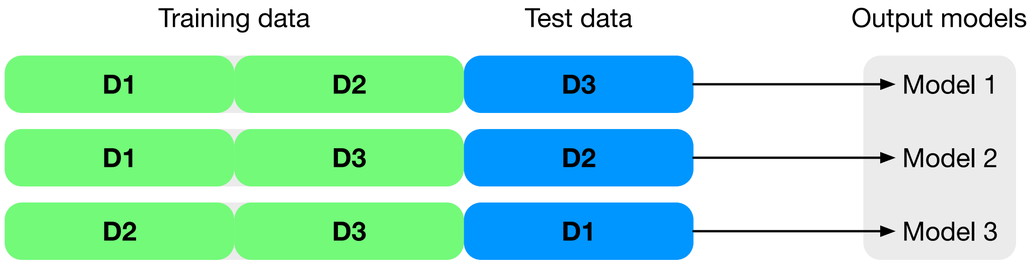
CV in Application

CV in Application

- Train K models
- Estimate performance (loss, distribution)

Caveat

- Requires “deterministic splitting”



$$\textbf{Err} = \sum_{k=1}^{K} \ell\left(\hat{f}_k, \textbf{target}\right)$$

CV in Application

- Random number at event generation
  - Should be uncorrelated
  - Event number for HEP
- Calculate fold from this random number

- Apply final model to data

```
TMVA::DataLoader dl{“dataset”};
dl.AddVariable(“x");
dl.AddSpectator(“eventId");

//  ... snip ...

const char * opts =
    "AnalysisType=Classification:"
    "NumFolds=2:SplitExpr="
    "int([eventId])%int([NumFolds])";

TMVA::CrossValidation cv{"<jobname>",
    &dataloader, outputFile, opts};
```

Conclusion — CV

- Cross validation introduced in 6.10
- Continued improvement in 6.12 / 6.14
- Works like factory
- Integrates with TMVA workflow
- Supports “CV in application”

Data ingestion

Scope

- Integrate RDataFrame into TMVA for data loading
  - Update recommended workflow
  - Support with interface changes

Quick Intro to RDataFrame

ROOT 6.14 introduced RDataFrame

- High level interface for data transformations
- Efficient implementation (implicit multithreading)
- See talk of Enrico!

- How does it integrate with TMVA?

```
RDataFrame rdf{"Tree", "data.root"};

// x and y are branches already in tree
auto df = df.Define("z", "sin(x + y)");
auto df = df.Define("q", □(){
  Double_t val = do_some_processing();
  return val;
});

// Save to a new file
df.Snapshot("NewTree", "NewFile",
            {"x", "y", "z", "q"});
```

Current Situation

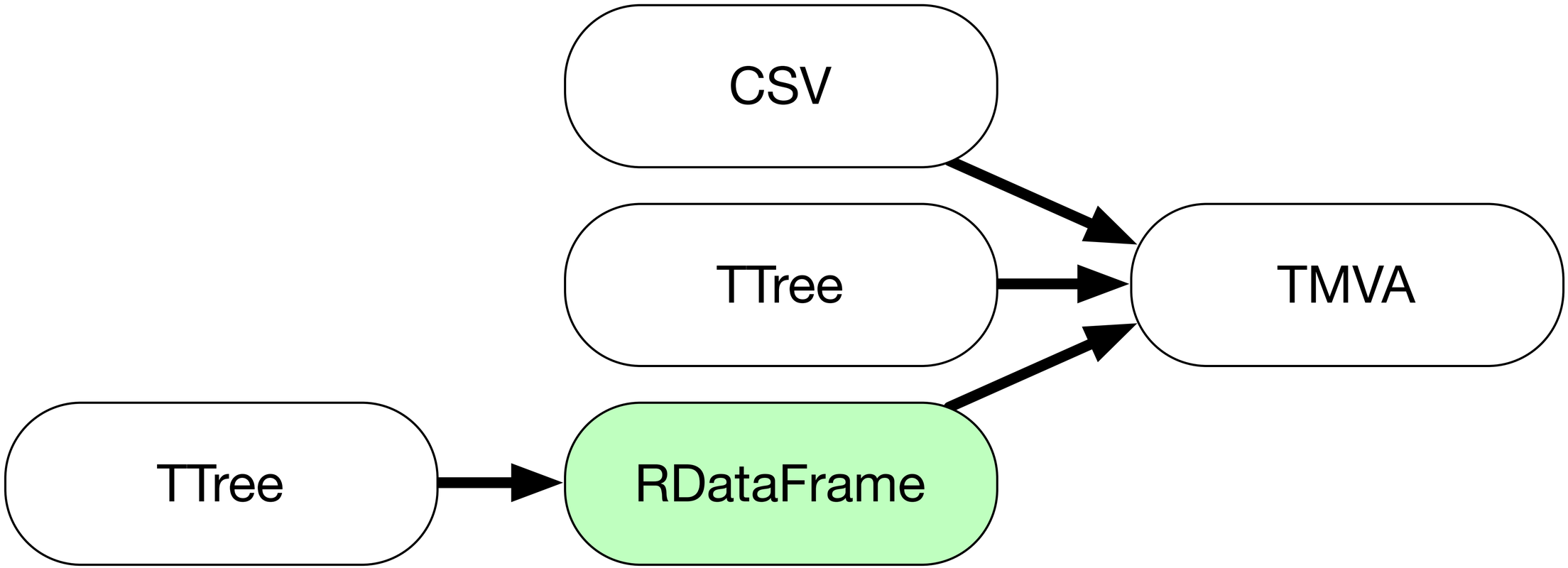
Current issues

- Verbose
- Transformation mini-language
  - Limited
  - What if you want to cache to disk?

No glaring issues, but room for improvement

```
// Prepare data sources
auto datafile =
    TFile::Open("data.root");
TTree * tree = datafile→Get(“Signal");

// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddVariable("x");
dl.AddVariable("y");
dl.AddVariable("z := x + y");
dl.AddTree(tree, “Signal");

// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

RDF Data Source



RDF Data Source

Move data processing to RDataFrame!

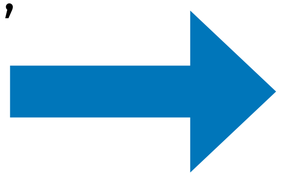
Benefits

- Expressive transformations (only c++)
- Possibly improved performance
- Leads to further simplifications

```
// Prepare data sources
RDataFrame rdf{"Signal", "data.root"};
auto df = df.Define("z", "x + y”);


// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddVariable("x");
dl.AddVariable("y");
dl.AddVariable("z");
dl.AddDataFrame(df, "Signal");

// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

Easy Caching

```
// Prepare data sources
RDataFrame rdf{"Signal", "data.root"};
auto df = df.Define("z", "x + y”);

// Cache transformations
df.Snapshot(“Signal”,
    “data_preprocessed.root”,
    {“x”, “y”, “z”});
```



```
// Prepare data sources
RDataFrame rdf{"Signal",
   "data_preprocessed.root"};


// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddVariable("x");
dl.AddVariable("y");
dl.AddVariable("z");
dl.AddDataFrame(df, "Signal");

// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

- Easy to move between on-the-fly and cached transformations
- Removes boilerplate in training and application

Interface Simplification

- Repeated use of dl.AddVariable("x");
- Information already in RDataFrame

```
// Prepare data sources
RDataFrame rdf{"Signal",
   "data_preprocessed.root"};


// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddVariable("x");
dl.AddVariable("y");
dl.AddVariable("z");
dl.AddDataFrame(df, "Signal");

// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

Interface Simplification

- Repeated use of dl.AddVariable("x");
- Information already in RDataFrame
- Default initialisation of Variables
  - Unless overridden by you!

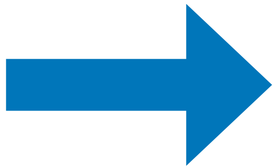
```
// Prepare data sources
RDataFrame rdf{"Signal",
   "data_preprocessed.root"};
```

```
// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddDataFrame(df, “Signal",
   {“x", "y", “z"});
```

```
// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

Conclusion — Ingestion

```
// Prepare data sources
auto datafile =
    TFile::Open("data.root");
TTree * tree = datafile→Get(“Signal");

// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddVariable("x");
dl.AddVariable("y");
dl.AddVariable("z := x + y");
dl.AddTree(tree, “Signal");

// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```



```
// Prepare data sources
RDataFrame rdf{"Signal",
   "data_preprocessed.root"};

// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddDataFrame(df, “Signal",
   {“x", "y", “z"});

// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

Conclusion — Ingestion

- Simpler code in training and application
- Same functionality
- Potentially better speed
- You can already use the ideas since 6.14
- Nicer interface with 6.16 (?)

```
// Prepare data sources
RDataFrame rdf{"Signal",
   "data_preprocessed.root"};


// Declare data format
TMVA::DataLoader dl{"dataset"};
dl.AddDataFrame(df, “Signal",
   {“x", "y", “z"});


// Sythesise dataset
dl.PrepareTrainingAndTestTree(
    "<cut>", "<splitopt>");
```

Thanks

Get in touch!

https://root.cern/tmva

https://root.cern/forum

Supplementary slides

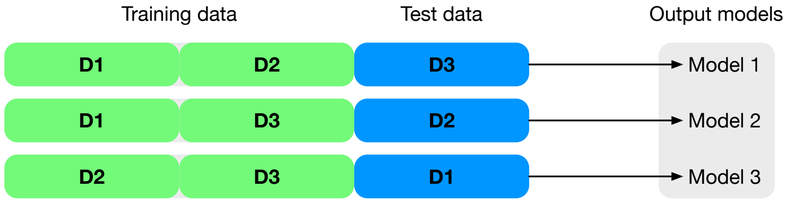
Cross Validation in Application

“CV in application”

- Workflow in HEP analysis
- Deterministic assignment of events to folds + save all trained models
- Performance estimation holds for collection of models

Standard approach

Cross validation in application



Used in e.g.

- Evidence for the H → bb¯ decay with the ATLAS detector (2017) — arxiv:1708.03299
- Search for the bb decay of the Standard Model Higgs boson in associated (W/Z)H production with the ATLAS detector (2015) — arxiv:1409.6212