



# Google Summer of Code Project

## Integrating Machine Learning in Jupyter Notebooks

**Attila Bagoly** (Eötvös Loránd University, Hungary)

Mentors:

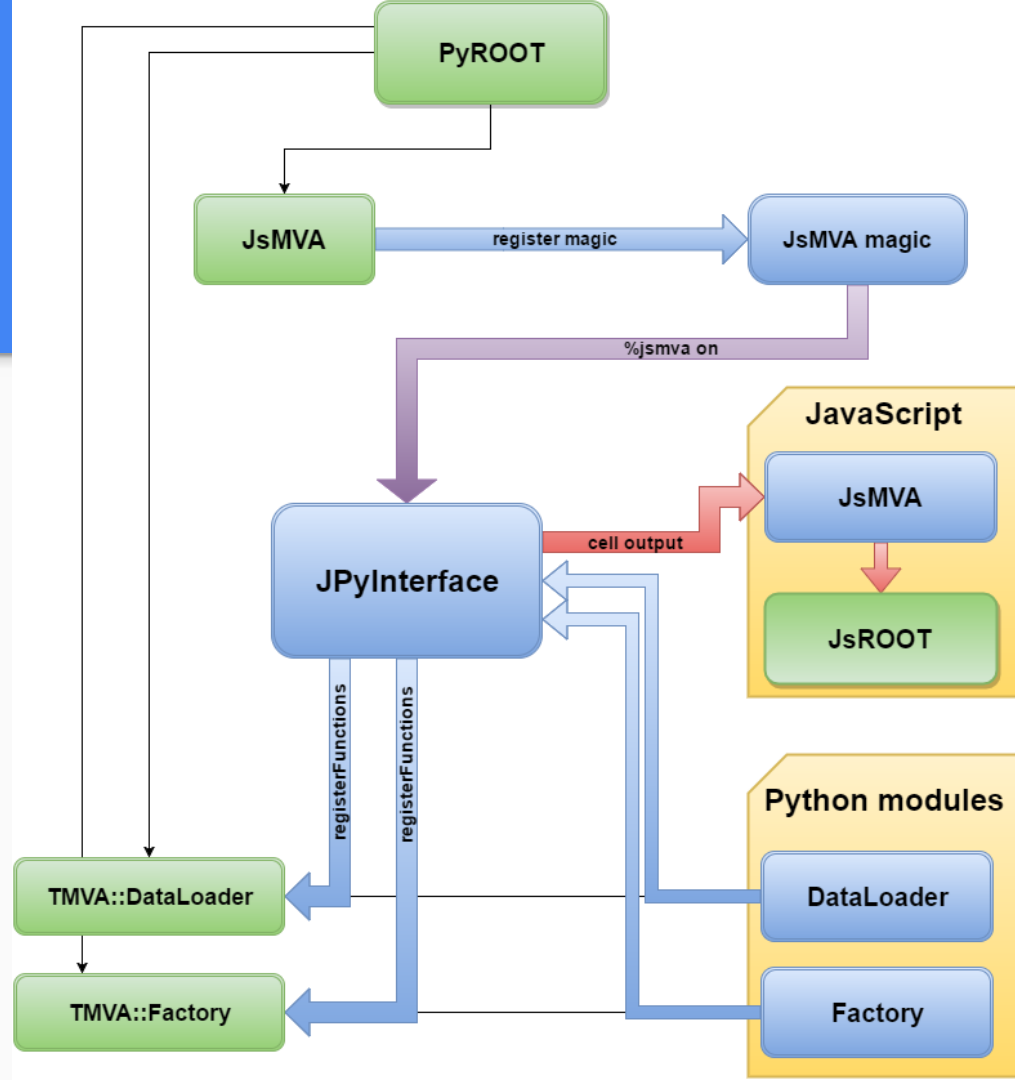
- **Sergei V. Gleyzer**
- **Enric Tejedor Saavedra**



- Jupyter notebook:
  - Interactive coding
  - Document: HTML, [Markdown](#) support
  - Shareable: SWAN, nbviewer, binder
- Current status of TMVA:
  - We can't use TMVAGui
  - We can read back the classifier outputs and we can make visualizations.
  - **BUT users don't want to spend time with making visualizations**
- Integrating TMVA in Jupyter:
  - Support for TMVAGui in notebooks
  - New visualizations
  - Pythonic interface for a bunch of functions
  - Interaction: changes modify the state of TMVA
  - HTML formatted output

# Code structure

- Importing ROOT will import **JsMVA**, this will register jsmva magic
- `%jsmva on`: JPyInterface inserts new methods to `TMVA.DataLoader` and `TMVA.Factory`, overloads some functions with a wrapper, register HTML transformer function
- New methods: inserting HTML to cell output, with JavaScript call for `JsMVA.js`
- `JsMVA.js` using **JsROOT** to create JavaScript plots

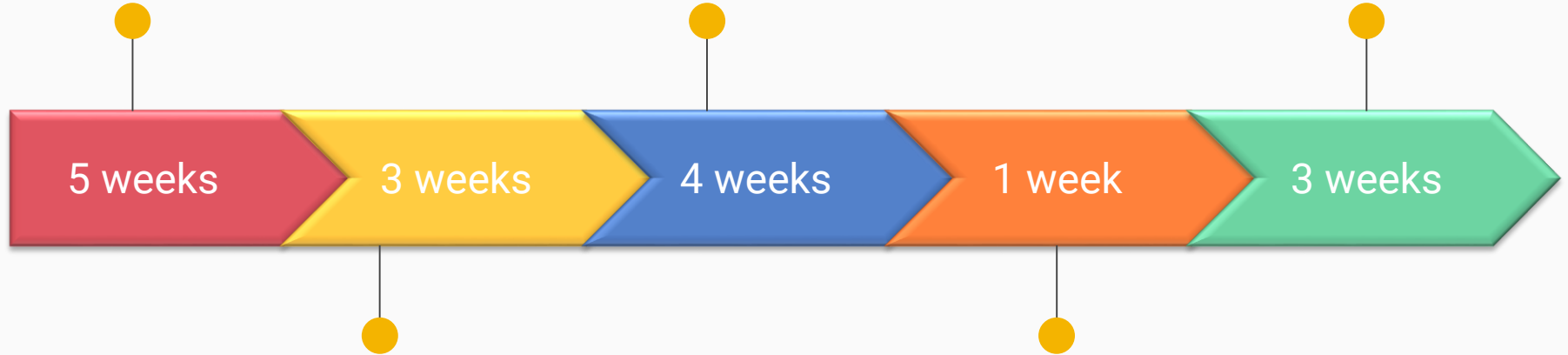


# Timeline

TMVAGui  
visualizations

Interactive training  
mode

Deep neural network  
builder, HTML  
formatted output



Neural network, deep  
network, decision tree  
visualizations

New user interface,  
documentation,  
tutorials

# TMVAGui visualizations

```
loader.DrawCorrelationMatrix("Signal")
```

Correlation Matrix (Signal)



## Visualizations related to input variables

- Correlation matrix
- Input variables
- Transform input variables & show

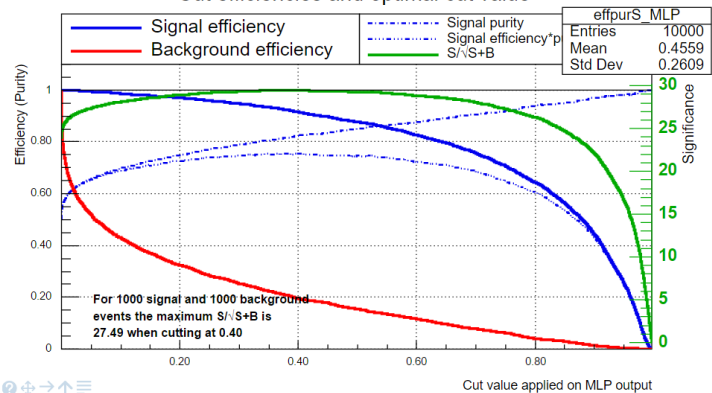
## Visualizations related to classifier outputs

- ROC curve
- Output distributions
- Cut efficiencies

[Go back](#)

```
factory.DrawCutEfficiencies(dataset, "MLP")
```

Cut efficiencies and optimal cut value



# Classifier output: Neural networks, decision trees

## Simple neural network

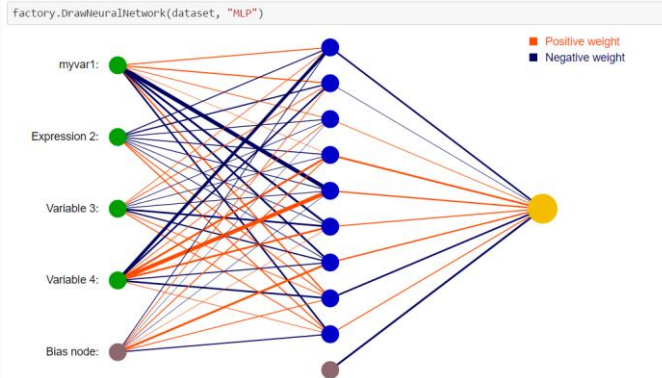
- Python function reads the network, converts to JSON; JS with d3js make the visualization from JSON
- Interactive: focusing connections, zooming, moving

## Deep neural network

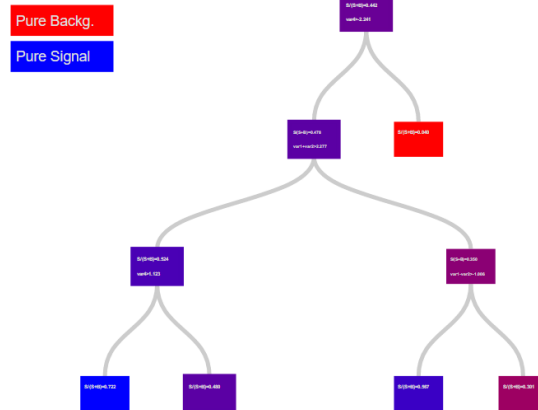
- HTML5 Canvas visualization (speed)
- Less interactive: zooming, moving

## Decision trees

- Ipywidgets: input field for selecting the tree
- Visualization from JSON with D3js
- Interactive: closing subtree, showing the path, focusing, moving, zooming, reset



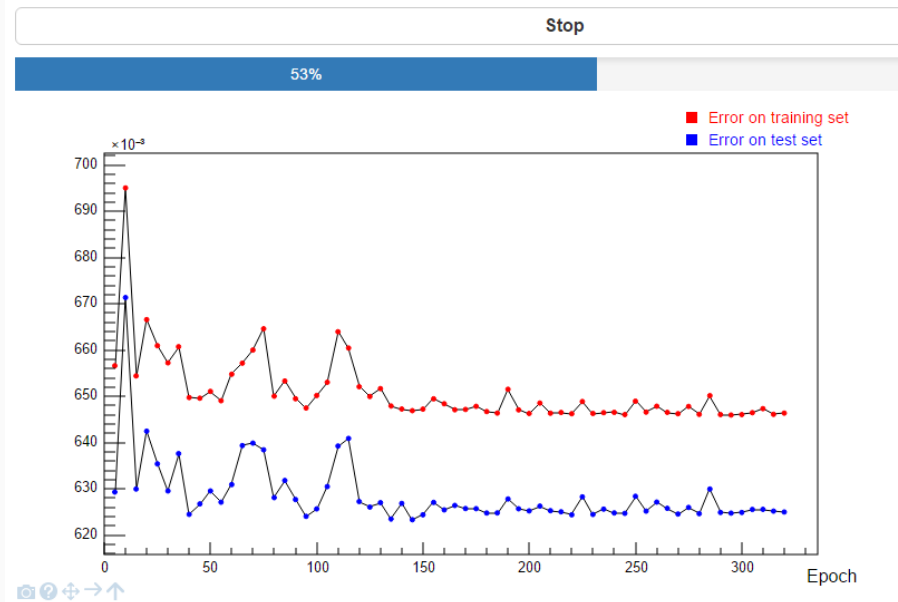
factory.DrawDecisionTree(dataset, "BDT") #11



# Interactive training mode

- C++ interface for tracking/stopping the training
- New thread for training
- Main thread periodically refreshes the plot (inserts small JS script, which removes itself)
- Error plots supported for MLP, DNN, BDT methods
- Progress bar for a bunch of methods
- Stop button: by clicking on it the main thread will send stop message for training loop (just the loop, no interfere with saving the net, or other data)

## Train method: MLP



## Train method: Cuts



# Pythonic user interface, tutorial notebooks, documentation

```
factory = TMVA.Factory(JobName="TMVAClassification", TargetFile=outputFile,  
                       V=False, Color=True, DrawProgressBar=True, Transformations=["I", "D", "P", "G", "D"],  
                       AnalysisType="Classification")
```

## New interface

- Option strings not very nice, we can do better in python
- Bunch of functions use option string
- Wrapper functions for them, with jsmva magic these functions are replaced with corresponding wrapper function
- The settings can be passed by named arguments: `V=True, Transformations=["I", "D"]` will be translated to `!V:Transformations=I,D`

Arguments of constructor: The options string can contain the following options:

Keyword	Can be used as positional argument	Default	Predefined values	Description
JobName	yes, 1.	not optional	-	Name of job
TargetFile	yes, 2.	if not passed histograms won't be saved	-	File to write control and performance histograms histograms
V	no	False	-	Verbose flag
Color	no	True	-	Flag for colored output
Transformations	no	""	-	List of transformations to test. For example with "I;D;P;U;G" string identity, decorrelation, PCA, uniform and Gaussian transformations will be applied
Silent	no	False	-	Batch mode: boolean silent flag inhibiting any output from TMVA after the creation of the factory class object
DrawProgressBar	no	True	-	Draw progress bar to display training, testing and evaluation schedule (default: True)
AnalysisType	no	Auto	Classification, Regression, Multiclass, Auto	Set the analysis type

[Tutorial, documentation links on](#)



[GitHub.com/qati/GSOC16](https://github.com/qati/GSOC16)



# Deep neural network builder, HTML formatted output

- Booking DNN confusing: lot of settings, everybody forgets the exact names
- Graphical interface: booking DNN with pleasure
- We can add different types of layers
- Specify the neuron number and training strategy for layer
- Connect the layers: building the network
- Save network: transform the graphical representation to option string and books the method

- jsmva magic register output transformer function, inserts CSS to notebook
- Structures the data to CSS formatted HTML table

The screenshot displays a graphical user interface for building a deep neural network. At the top, the title bar reads "factory.BookDNN(loader)". Below it, a menu bar contains "Global options", "Add layer", "Connect layers", "Scale colors", and "Save network".

The main workspace shows a network diagram with four nodes: a green "Input layer", a purple "Options TANH" node, a red "Options ReLU" node, and a yellow "Output layer Options" node. Curved lines connect the Input layer to the TANH node, the TANH node to the ReLU node, and the ReLU node to the Output layer. A large white arrow points from the diagram towards the data table below.

Below the diagram is a table with two rows of "DataSetInfo":

DataSetInfo	Dataset: tmva_class_example Added class "Signal"
DataSetInfo	Dataset: tmva_class_example Added class "Background"

Below the table, there are two entries for "Add Tree":

- Add Tree TreeS of type Signal with 6000 events
- Add Tree TreeB of type Background with 6000 events

On the right side, there are two floating windows. The "Training Strategy" window shows various parameters: LearningRate (0), Momentum (0), Repetitions (3), ConvergenceSteps (1), BatchSize (3), TestRepetitions (7), WeightDecay (0), Regularization (NONE), DropConfig, DropRepetitions (3), and Multithreading (checked). The "Add neurons" window shows "Number of neurons: 0".

## **Everything on GitHub:**

<https://github.com/qati/GSOC16>

## **Notebooks on nbviewer (static, rendered):**

<http://nbviewer.jupyter.org/github/qati/GSOC16/blob/master/index.ipynb>

## **Notebooks on binder (interactive):**

[www.mybinder.org/repo/qati/GSOC16](http://www.mybinder.org/repo/qati/GSOC16)

## **Or you can download:**

<https://github.com/qati/GSOC16/tree/master/notebooks>

Thank you!