Development of Machine Learning Tools in ROOT

S. V. Gleyzer[1], **L. Moneta**[2], **Omar A. Zapata**[3]
[1] University of Florida
[2] CERN
[3] University of Antioquia and Metropolitan Institute of Technology

E-mail: `Sergei.Gleyzer@cern.ch, Lorenzo.Moneta@cern.ch, Omar.Zapata@cern.ch`

Abstract. ROOT is a framework for large-scale data analysis that provides basic and advanced statistical methods used by the LHC experiments. These include machine learning algorithms from the ROOT-integrated Toolkit for Multivariate Analysis (TMVA). We present several recent developments in TMVA, including a new modular design, new algorithms for variable importance and cross-validation , interfaces to other machine-learning software packages and integration of TMVA with Jupyter, making it accessible with a browser.

1. Introduction
ROOT is an object-oriented data analysis framework that provides statistical methods, visualization and storage libraries for data analysis of high-energy physics (HEP) experiments such as the Large Hadron Collider (LHC) in Geneva, Switzerland [1]. Although originally designed for HEP applications, ROOT is also widely used in other scientific fields outside of particle physics.

Machine learning algorithms have become integral to modern HEP analyses. The ROOT framework provides TMVA, the Toolkit for Multivariate Analysis [2],that contains widely-used machine learning algorithms in HEP such as:

- Fisher and Linear Discriminants (LD)
- Boosted Decision Trees (BDT)
- Decision Rule Ensembles
- k-Nearest Neighbor Classifier (KNN)
- Shallow Artificial Neural Networks (ANN)
- Deep Learning Neural Networks (DNN)
- Support Vector Machines (SVM)

Of these, most popular methods in HEP are boosted decision trees, neural networks and support vector machines. TMVA provides implementations of these popular methods applicable for machine-learning classification and regression. TMVA provides a way to compare the performance of these algorithms on the same dataset, allowing for an apples-to-apples comparison useful in choosing the optimal algorithm for a particular analysis task.

Recently, TMVA has been undergoing a significant makeover targeting greater flexibility, modular design and some new features and interfaces. As machine learning algorithms continue to evolve, the design of machine learning frameworks needs to be as flexible as possible.

2. New Algorithms and Features

Algorithms with useful information for the user have been added, such as cross-validation and variable importance. Also TMVA was integrated with Jupyter, allowing execution in a browser. In the next sections we describe some of the new functionality and features of TMVA.

2.1. Data Loader

One new design development in TMVA is the DataLoader class. This class creates greater flexibility and modularity in training different combinations of classifiers and variables. Previously, the choice of variables was defined once and could not be changed later. The new DataLoader class allows the user to combine different choices of variables and methods with the data. This design flexibility allows other useful algorithms, such as cross-validation and variable importance, described in the following sections, to be implemented.

Figures 1, 2 and 3 show the structure of the DataLoader class. New algorithm features require several DataLoaders at the same time. Various file formats, for example .root and comma-separated value (.csv) files are supported in the DataLoader class.

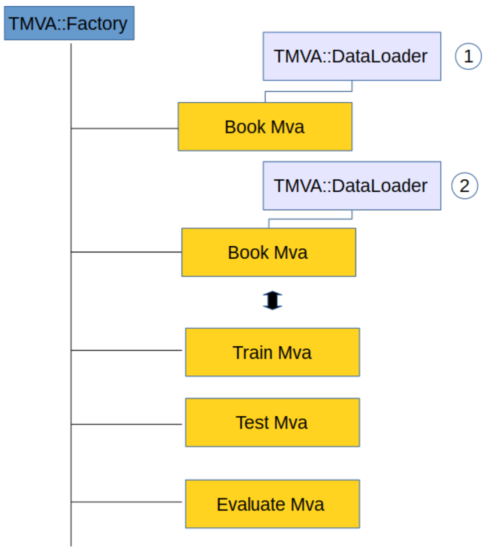

Figure 1. Booking methods with different dataloaders.

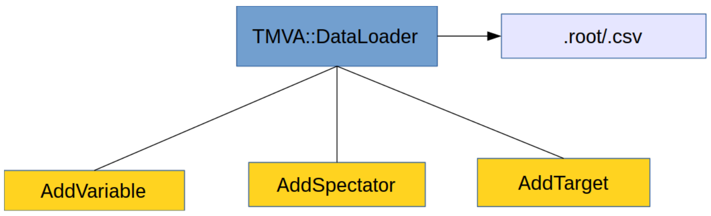

Figure 2. Loading data from files.

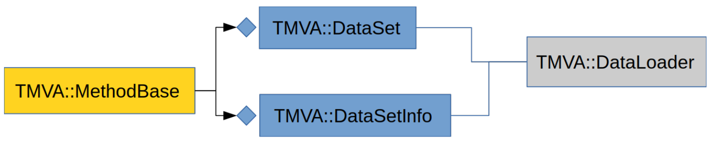

Figure 3. Storing data in objects in the class MethodBase.

2.2. Cross Validation

Cross-validation is a machine-learning model evaluation technique, important for model generalization to unseen data. During k-fold cross-validation, the dataset is partitioned into k folds or partitions. During one cross-validation round, k-1 folds are used for model training, and the remaining one for model testing. Several rounds of cross-validation are performed with different partitions and model performance results are averaged. One advantage of cross-validation over a simple split into training and testing set, is the use of the full dataset to validate the model. Performing cross-validation is known to reduce over-fitting of the data, leading to a more accurate estimate of the performance of the machine-learning model on unseen data [3].

Cross-validation in TMVA is done with a standalone CrossValidation class. Figure 5 illustrates five receiver-operating characteristic (ROC) curves for each cross-validation fold for a basic TMVA example. This example has four random variables with gaussian distributions plus several derived variables after applying basic mathematical operations to these variables.

2.3. Variable Importance

Currently, TMVA provides a number of method-specific variable importance algorithms. Each one is relevant only for the method chosen and is computed during construction. For example, for decision trees variable importance is derived by counting the number of splits for each variable weighted by the square of the information gained from the split, or for neural networks, as the sum of weights between the inputs and the hidden layer [2].

In addition to these, a new method-independent variable importance algorithm was added. This algorithm, described in [4], computes variable importance in the context of classifier performance. A number of seeds are randomly generated, each corresponding to a variable subspace. For each seed, individual variable contributions to classifier performance are measured as loss of classifier performance due to the removal of a variable. Figure 4 shows a sample variable importance plot for a basic example in TMVA.

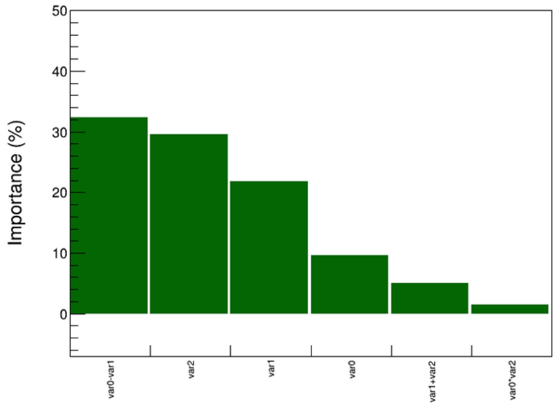

Figure 4. Histogram ranking the variables.

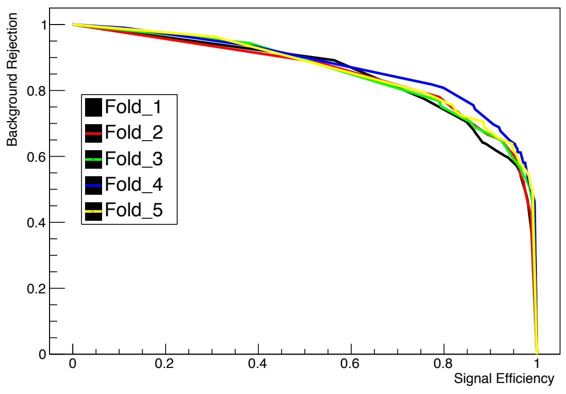

Figure 5. Cross validation ROC curves.

2.4. TMVA and Jupyter Notebooks

Another new feature in TMVA is the integration of TMVA and Jupyter Notebooks [5]. Jupyter is a web application that combines live code, rich text, links and formula in a user-friendly format. With the creation of a ROOT Jupyter kernel, additional integration of machine-learning tools in ROOT with Jupyter notebooks became possible. Currently, all the functionality of TMVA is available in Jupyter notebooks, requiring only access to a browser to run and execute TMVA.

3. Interfaces to external machine learning tools

Another useful functionality added to TMVA is the interface to external machine-learning tools written in R and Python languages. Figure 6 shows the relationship between ROOT, TMVA and other statistical packages.

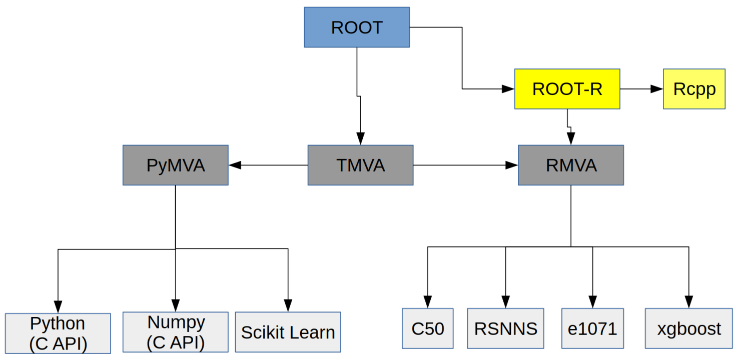

Figure 6. Interplay of Machine Learning Tools in ROOT.

3.1. ROOT-R Interface

R is a free software framework for statistical computing[6]. ROOT-R interface was developed to use R functions directly in ROOT. It opens a large set of statistical tools available in R, including machine-learning packages, for use within ROOT. The ROOT-R interface design is shown in Figure 7.

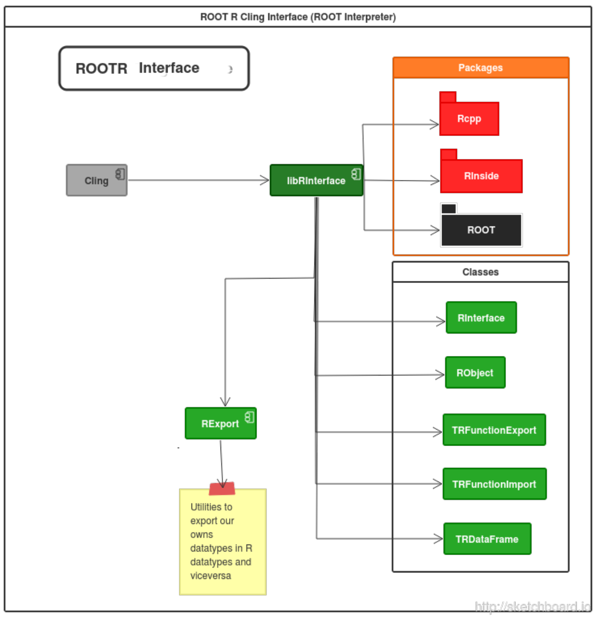

Figure 7. ROOT-R Interface design.

3.2. RMVA Interface

RMVA is a set of TMVA plugins based on the ROOT-R interface. It allows the use of machine-learning methods available in R directly from TMVA. The goal behind RMVA is not to replace the R package itself but to allow its direct comparison with existing tools in TMVA for a given problem, while adding more methods to TMVA users toolbox.

The RMethodBase class in TMVA starts the R environment using ROOT-R, imports the required modules and maps the DataLoader events in R data frame objects using the helper class ROOT::R::TRDataFrame. Each of the methods inherits from the base class RMethodBase as shown in Figure 8. Currently, the following machine-learning packages in R are supported:

- Decision trees and rule-based models (C50) [7].
- Stuttgart Neural Networks in R (SNNS)[8].
- Support Vector Machines in R (e1071)[9].
- eXtreme Gradient Boost (xgboost) An optimized general purpose gradient boosting library[10].

As shown in Figure 10 the above machine-learning methods in R can be executed within the TMVA framework for the basic example.

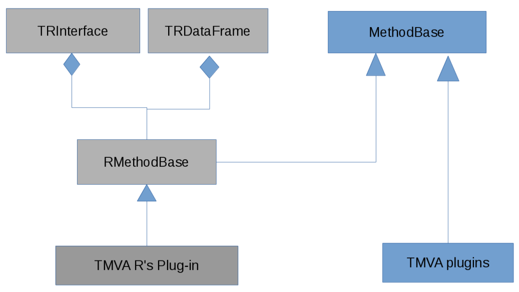

Figure 8. RMVA plugins system.

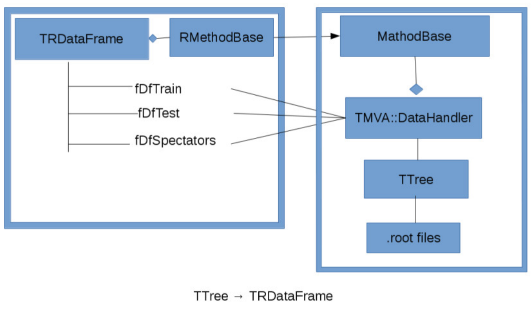

Figure 9. ROOT-R and TMVA data flow.

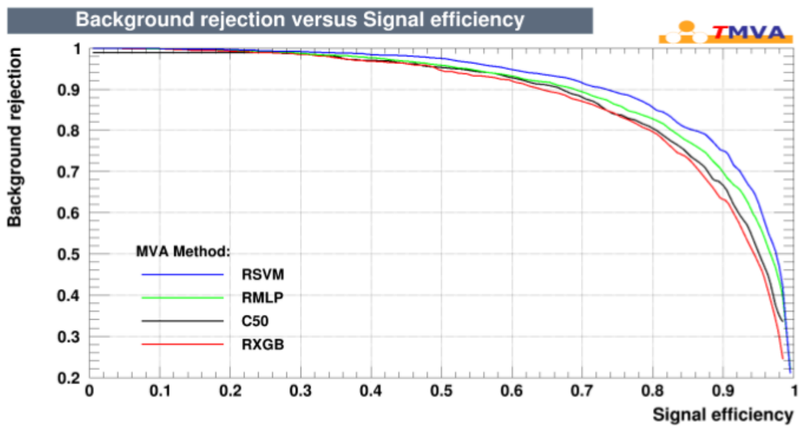

Figure 10. ROC Curves for RMVA methods.

3.3. Python with TMVA (PyMVA)

Similarly to RMVA, PyMVA is a set of TMVA plugins based on Python API that allows direct use of machine-learning methods written in Python from within TMVA. The goal, similarly to RMVA, is not to replace the original method, but to allow the comparison of the method with the other methods in TMVA using the same dataset, selections and performance metrics.
The PyMethodBase class in PyMVA initializes the Python environment, imports the required modules and maps the DataLoader events in numpy arrays using C-API. Each PyMVA method inherits from the base class PyMethodBase, as illustrated in Figure 11. Figure 12 shows how the dataset is mapped from ROOT trees to numpy arrays. The following Python based methods from the Scikit-learn software package [11] are currently available in TMVA:

- Random Forest (PyRandomForest)
- Gradient Boosted Regression Trees (PyGTB)
- Adaptive Boosting (PyAdaBoost)

Figure 13 shows the ROC curves of various PyMVA methods for a basic example.

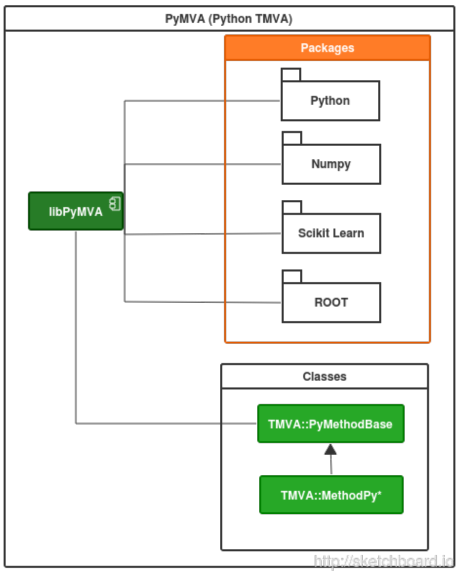

Figure 11. PyMVA plugins system.

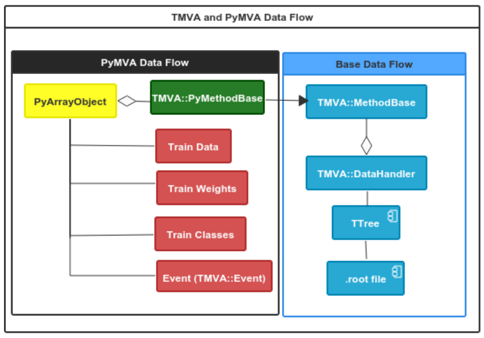

Figure 12. PyMVA data flow.

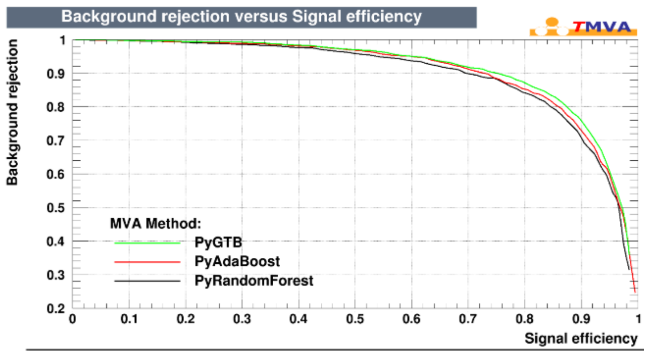

Figure 13. ROC Curves for PyMVA.

3.4. Conclusions

Machine learning tools in ROOT have undergone a significant makeover and upgrade. In particular, TMVA has a new design targeting greater flexibility and modularity, new features such as cross-validation, variable importance and interfaces to R and python-based machine-learning tools. In addition, TMVA is available in Jupyter notebooks, making it accessible in a web browser.

Acknowledgments

The work of O. A. Zapata was partially supported by Sostenibilidad-UdeA, UdeA/CODI grant IN361CE and COLCIENCIAS grant 111-556-934918.

References

[1] Antcheva I, Ballintijn M, Bellenot B, Biskup M, Brun R, Buncic N, Canal P, Casadei D, Couet O, Fine V, Franco L, Ganis G, Gheata A, Maline D G, Goto M, Iwaszkiewicz J, Kreshuk A, Segura D M, Maunder R, Moneta L, Naumann A, Offermann E, Onuchin V, Panacek S, Rademakers F, Russo P and Tadel M 2009 *Computer Physics Communications* **180** 2499 – 2512 ISSN 0010-4655 40 {YEARS} {OF} CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures URL `http://www.sciencedirect.com/science/article/pii/S0010465509002550`

[2] Hoecker A, Speckmayer P, Stelzer J, Therhaag J, von Toerne E and Voss H 2007 (*Preprint* `physics/0703039`)

[3] Arlot S, Celisse A *et al.* 2010 *Statistics surveys* **4** 40–79

[4] Gleyzer S V and Prosper H B 2008 *PoS* 067

[5] Pérez F and Granger B E 2007 *Computing in Science and Engineering* **9** 21–29 ISSN 1521-9615 URL `http://ipython.org`

[6] R Core Team 2016 *R: A Language and Environment for Statistical Computing* R Foundation for Statistical Computing Vienna, Austria URL `https://www.R-project.org/`

[7] Kuhn M, Weston S, Coulter N and code for C50 by R Quinlan M C C 2015 *C50: C5.0 Decision Trees and Rule-Based Models* r package version 0.1.0-24 URL `https://CRAN.R-project.org/package=C50`

[8] Bergmeir C and Benítez J M 2012 *Journal of Statistical Software* **46** 1–26 URL `http://www.jstatsoft.org/v46/i07/`

[9] Meyer D, Dimitriadou E, Hornik K, Weingessel A and Leisch F 2015 *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien* r package version 1.6-7 URL `https://CRAN.R-project.org/package=e1071`

[10] Chen T and He T 2015 *R package version 0.4-2*

[11] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V *et al.* 2011 *Journal of Machine Learning Research* **12** 2825–2830