DAME Web Application REsource Plugin Creator User Manual

DAMEWARE-MAN-NA-0016

Issue:	2.1
Date:	March 20, 2014
Authors:	S. Cavuoti, A. Nocella, S. Riccardi, M. Brescia

Doc. : ModelPlugin_UserManual_DAMEWARE-MAN-NA-0016-Rel2.1

Index

TABLE INDEX

FIGURE INDEX

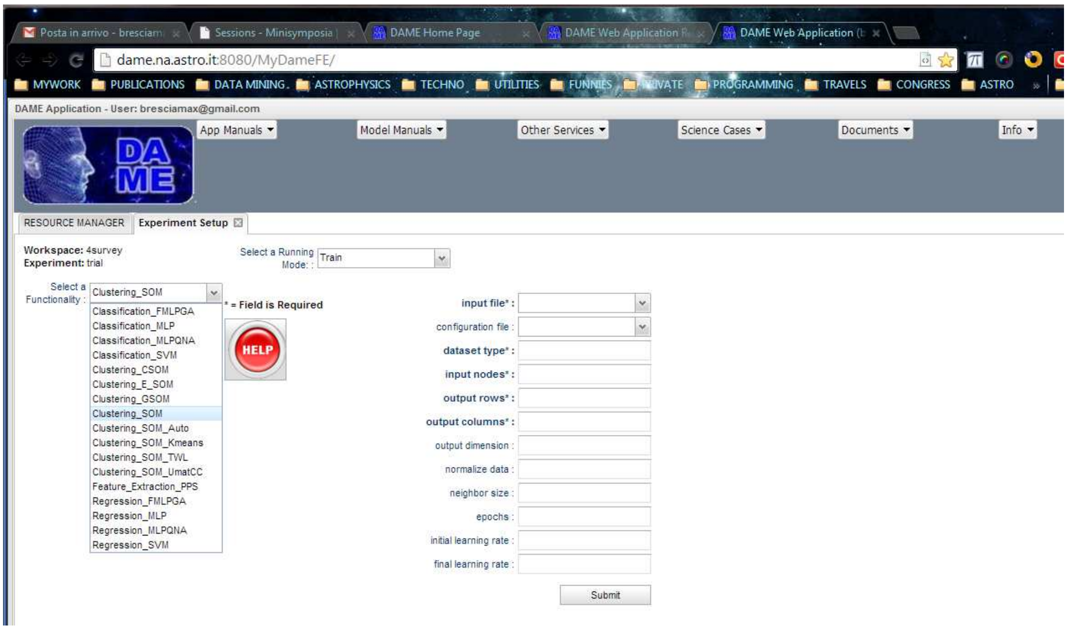
1 Introduction

The present document is the user guide of the DAMEWARE plugin creator, designed to make able end users of the data mining web app to easily integrate new machine learning models within the DAMEWARE infrastructure without any specific knowledge about its internal architecture.

This manual, together with the plugin Java program DAMEWAREPluginWizard.jar, is part of the web app documentation package, and it is available at the address:

http://dame.dsf.unina.it/dameware.html#plugin .

The plugin creator, through a guided procedure, allows user to configure all machine learning model parameters, use cases and I/O information setup. Then it generates source code to be physically included into the infrastructure and to be finally exposed as new machine learning resource available to the community (as shown in Fig. 1Fig. 2).



dame.na.astro.it:8080/MyDameFE/

DAME Application - User: bresciamax@gmail.com

Workspace: 4survey
Experiment: trial

Select a Running Mode : Train

Select a Functionality : Clustering_SOM

* = Field is Required

Fig. 1 – The list (left popup menu) of machine learning models and related functionalities available in the DAMEWARE web application

The plugin creator implicitly assumes that the user model to be plugged in is available in the form of an executable, provided with a command line parameter list or with external configuration files (max 2 external files).

2 System Requirements and Installation

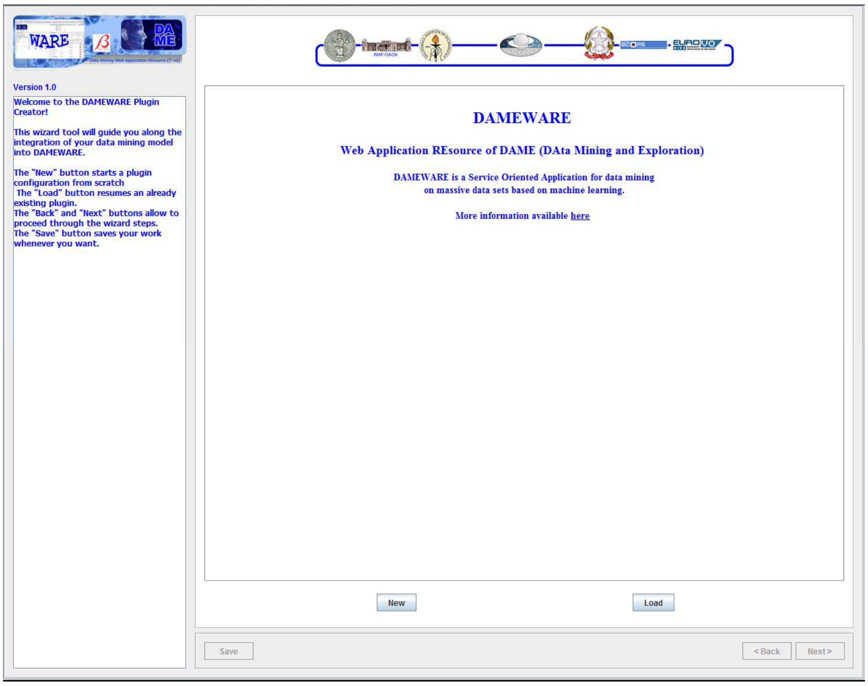
The only basic requirement to download and execute the Java plugin creator on local user machine is that the Java Runtime Environment (JRE) is properly installed. If not, please download and install the JRE from

http://www.oracle.com/technetwork/java/javase/downloads/index.html

After JRE installation, download the plugin application from the web address specified in the introduction above. Then simply double click on it or execute “java –jar DAMEWAREPluginWizard.jar” from a OS command window.

3 The Java plugin wizard GUI

The following sections will drive users through the plugin integration procedure.
In Fig. 2 the intro panel is shown. It appears after the execution of the wizard application.



Version 1.0

Welcome to the DAMEWARE Plugin Creator!

This wizard tool will guide you along the integration of your data mining model into DAMEWARE.

The "New" button starts a plugin configuration from scratch
The "Load" button resumes an already existing plugin.
The "Back" and "Next" buttons allow to proceed through the wizard steps.
The "Save" button saves your work whenever you want.

DAMEWARE

Web Application REsource of DAME (DAta Mining and Exploration)

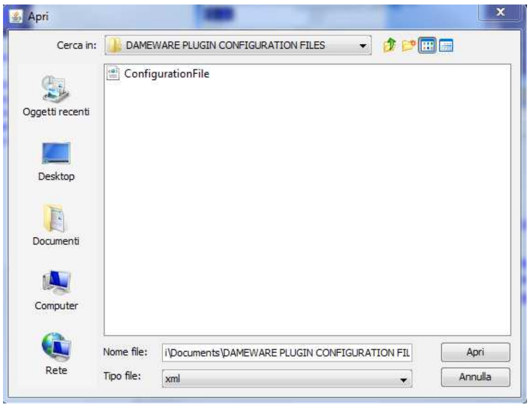
DAMEWARE is a Service Oriented Application for data mining
on massive data sets based on machine learning.

More information available here

Fig. 2 – introduction panel of the plugin wizard tool after launch

At the bottom of the intro panel there are two buttons “Back” and “Next”, initially disabled, useful to proceed (forward and backward respectively) through the procedure steps, described in the next sections.

The button “New” is used if user wants to start a plugin configuration from scratch. Otherwise it can be selected the “Load” button to load an already configured (partially or completely) and saved plugin.
In this last case, after selecting the “Load” option a new window will let you choose any of the XML configuration files available (Fig. 3).



Cerca in: DAMEWARE PLUGIN CONFIGURATION FILES

Nome file: i\Documents\DAMEWARE PLUGIN CONFIGURATION FIL

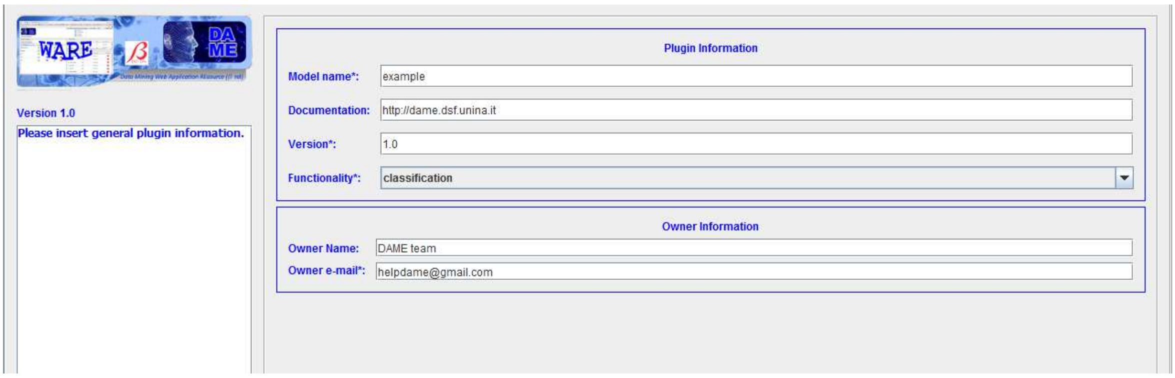
Tipo file: xml

Fig. 3 – loading XML plugin configuration file (already created) panel

In the following we will focus on the “New” option, starting the new plugin configuration procedure.

3.1 General Plugin Information

The Fig. 4 shows the general plugin information to be compiled.



Version 1.0

Please insert general plugin information.

Plugin Information

Model name*: example

Documentation: http://dame.dsf.unina.it

Version*: 1.0

Functionality*: classification

Owner Information

Owner Name: DAME team

Owner e-mail*: helpdame@gmail.com

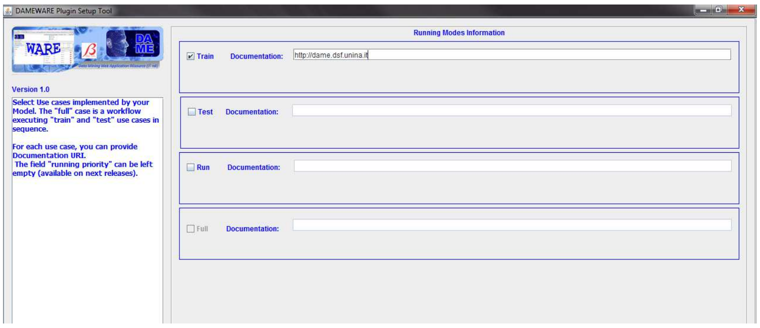
Fig. 4 – General model information panel

- **Model name***[1]: the name of the model to be plugged in. It is strongly suggested to use an acronym of the model (for example MLP for Multi Layer Perceptron). Spaces are not allowed here, use underscore instead, if required;
- **Documentation**: the URI address where to find model related documentation (if absent, any reference can be specified as well);
- **Version***: current model version or release;
- **Functionality***: the functional domain of experiments to be associated to the plugged model (for instance classification or regression for supervised models, clustering or feature_extraction for unsupervised). The user should be aware about the feasible domain for the model;
- **Owner Name**: the user name;
- **Owner e-mail***: a correct e-mail address of the user, to be contacted by DAMEWARE administrator;

At the end, click “Next” button to proceed.

3.2 Use Case Specifications

As common practice in the machine learning domain, any experiment requires a series of phases (use cases) to be performed in sequence. For instance, at the beginning of any experiment, a training phase is required to train the model (either in supervised and unsupervised cases). The test phase is usually the second step (sometimes also named as validation & test phase). The last step is the “run” phase, in which the trained and tested model is used as a generic function, applied to new data of the focused problem. There is also the “full” step, automatically performing both “train” and “test” phases, considered as a unique workflow.



DAMEWARE Plugin Setup Tool

Version 1.0

Select Use cases implemented by your Model. The "full" case is a workflow executing "train" and "test" use cases in sequence.

For each use case, you can provide Documentation URI.
The field "running priority" can be left empty (available on next releases).

Running Modes Information

☑ Train Documentation: http://dame.dsf.unina.it

☐ Test Documentation:

☐ Run Documentation:

☐ Full Documentation:

Fig. 5 – use case setup panel

[1] the asterisk means mandatory option

Fig. 5 shows the use case setup panel, in which the user can choose the steps available for the plugged model (usually a canonical machine learning includes all of them), by also specifying referenced documentation (if any).

If train and test cases checkboxes are selected, the "full" case (starting disabled) will become available. The "Next" button becomes available when at least one running mode is selected.
In the figure above, only the "Train" running mode is selected as an example.

3.3 Input parameter Setup

In Fig. 6 the main input data panel is shown, appearing after pressing "Next" button from the previous panel. It allows user to enter information about the parameters required by the model for each of the previously selected use cases. The figure reports an example for the Train use case (it is the same in other use cases).

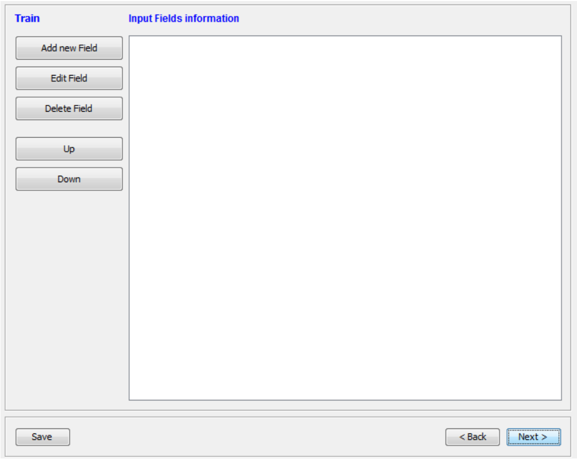


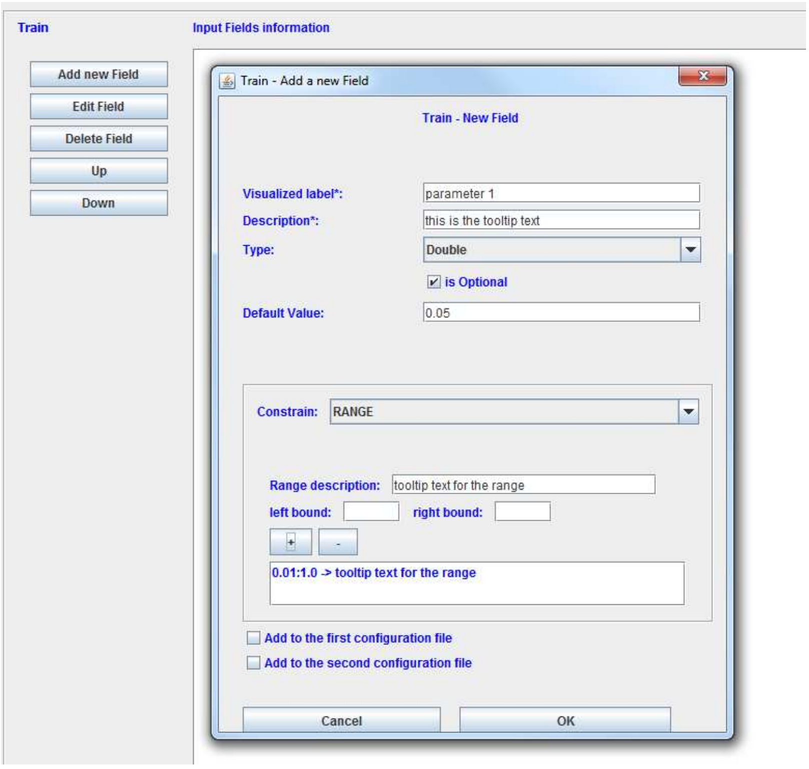
Fig. 6 – General panel to build new parameter fields (Train use case example)

3.3.1 New parameter setup

By selecting "Add new field" in the panel of Fig. 6, the panel in Fig. 7 allows to configure the input parameter by specifying:

- **Visualized label***: the label of the parameter as it will appear in the DAMEWARE web app experiment setup tab (for example "input layer neurons" or "error threshold");
- **Description***: a brief description of the parameter, appearing as tooltip;
- **Type**: the internal value allowed for the parameter (Integer, Double, Boolean, String). In the Boolean case the "true/false" value can be chosen;
- **Is Optional**: checkbox to be selected if the parameter will be not required (for an optional parameter, default value will be assigned if left empty);
- **Default Value**: if the "is optional" checkbox is selected, the default value field will become enabled and the user will have to enter the desired value (according to the type specified in the field "Type"); If desired value is not according to the type specified in the field "Type" an Error message will be shown;
- **Constraint**: if the parameter has a limited range of allowed values, the user can choose one of the three options "NO_CONSTRAINTS, RANGE, VALUES".
  - **IMPORTANT**: in the case of RANGE selected, the user must take care to insert the correct types of values as required by that field. For example, if the required range is between 0 and 1 for a double value, the user MUST insert 0.0 and 1.0 respectively;
- **Add to the first configuration file**: if this option is selected, the value of the parameter will be write in a configuration file, considered as the first one in case of the presence of two configuration files.
- **Add to the second configuration file**: if this option is selected, the value of the parameter will be write in a second of two foreseen configuration files.

If none of the two configuration file adding has been selected, the created parameter is by default considered in the executable command line list. In this case after pressing “OK” button a popup disclaimer confirming this will appear.



Train - Add a new Field

Train - New Field

Visualized label*: parameter 1

Description*: this is the tooltip text

Type: Double

☑ is Optional

Default Value: 0.05

Constrain: RANGE

Range description: tooltip text for the range

left bound: right bound:

0.01:1.0 -> tooltip text for the range

☐ Add to the first configuration file

☐ Add to the second configuration file

Fig. 7 – Example (Train use case) panel for new parameter (field) editing

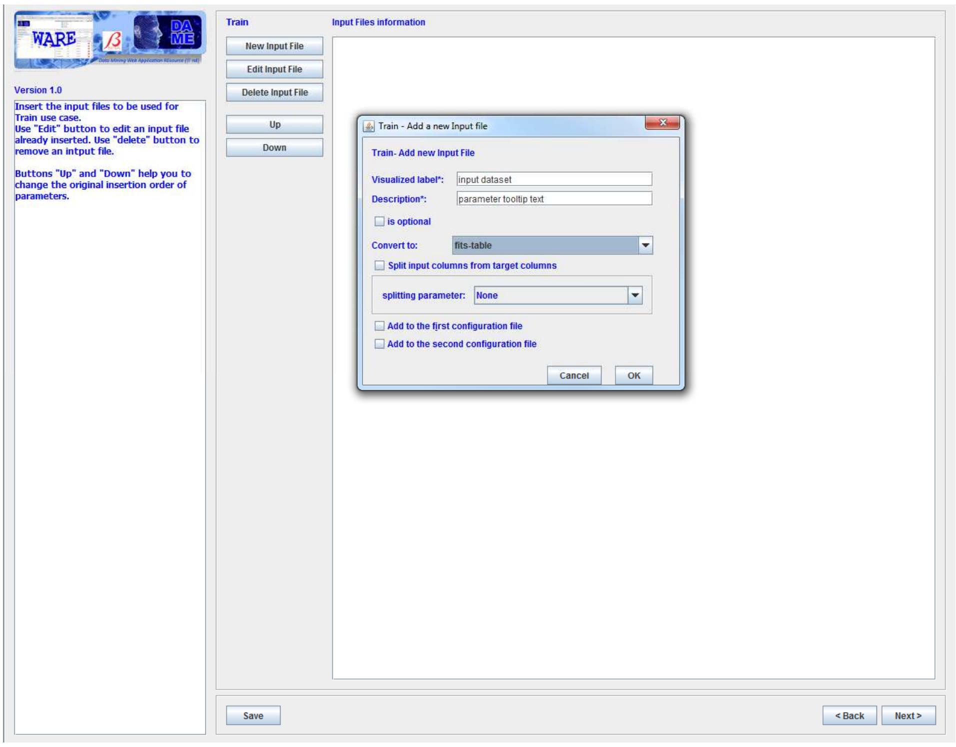
At the end, press “OK” button to save the parameter configuration.

3.3.2 Edit, sort and remove input parameter setup

By selecting “Edit Field” button in the panel of Fig. 6, it is possible to modify the saved parameter configuration.
By selecting “Delete Field” button in the panel of Fig. 6, it is possible to delete the saved configuration for a parameter.
Finally, the buttons “Up” and “Down” in the panel of Fig. 6 can be used to sort listed parameters (already configured) as they will appear in the sequence shown in the DAMEWARE web app model configuration browser tab during an experiment setup process.

3.4 Input file Setup

After pressing “Next” button, the input file choice panel is shown (Fig. 8). Any use case of a model foresees of course to specify input files to be used by the model (such as dataset, trained weight matrix for a neural network, external configuration files etc.). The panel shown reports an example of input file setup pop-up window (for instance in the Train use case, the same for the other use cases).



Version 1.0

Insert the input files to be used for Train use case.
Use "Edit" button to edit an input file already inserted. Use "delete" button to remove an intput file.

Buttons "Up" and "Down" help you to change the original insertion order of parameters.

Train- Add new Input File

Visualized label*: input dataset

Description*: parameter tooltip text

□ is optional

Convert to: fits-table

□ Split input columns from target columns

splitting parameter: None

□ Add to the first configuration file

□ Add to the second configuration file

Fig. 8 – Example (Train use case) of input file setup panel

3.4.1 Add new input file

By selecting “New Input File”, it is allowed to enter the input file information:

- **Visualized label***: the name of the input file type label (as it will appear in the DAMEWARE web app setup tab). For example: “dataset” or “trained weight file
- **Description***: a brief description of the file content;
- **Is optional**: if selected this input file field is considered not required (if optional it is empty, no default values will be used in this case);
- **Convert to**: if the model requires a particular format for this file (for example, the dataset should be given as a CSV file), the user can choose the right format by selecting one of the allowed formats listed in the pop-up menu (for instance, ASCII, CSV, CSV-no_header, FITS-TABLE and VOTABLE). **NOTE**: the DAMEWARE web app allows to arbitrarily submit input files in a quite wide range of supported formats. But the model usually allows only a particular format as input. So far, the web app will perform an embedded conversion to the right format, as specified by the user in this field.
- **Split input columns from target columns:** if the model requires to separate the rows of the input file (For example separate an input dataset file in Inputs and Targets using the number of input neurons));

At the end, click “OK” button to save the input file configuration.

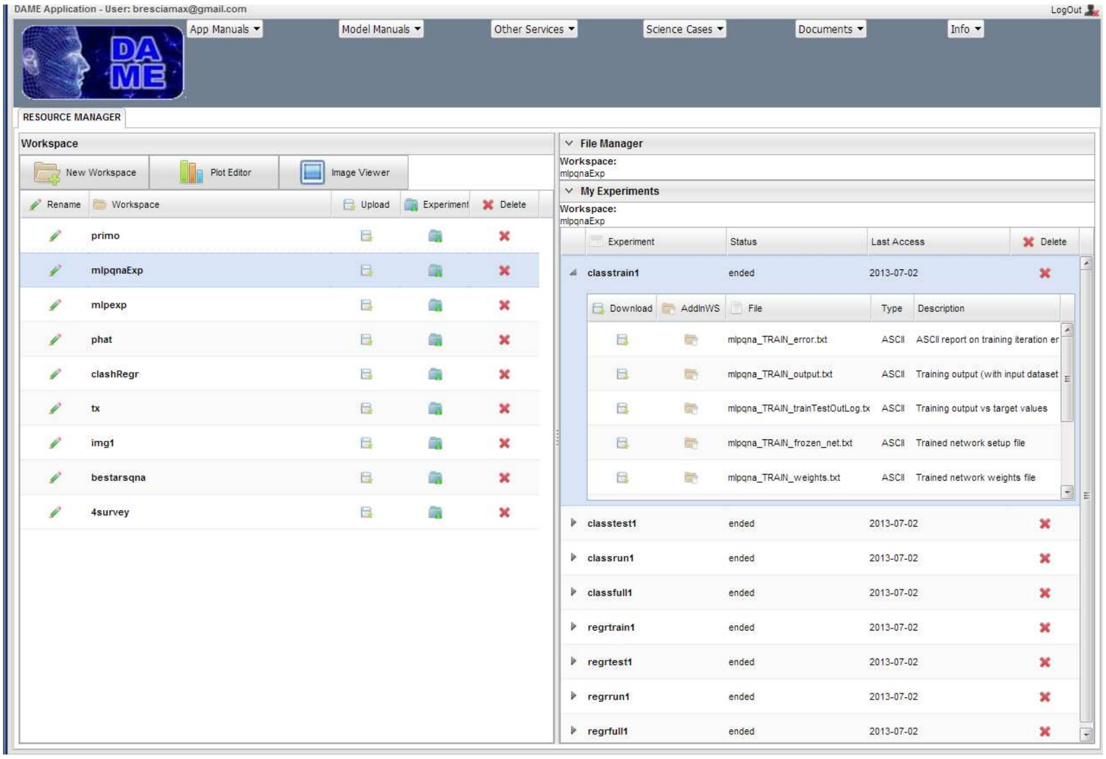
3.4.2 Edit, sort and remove an input file

By selecting “Edit Input File” button in the panel of Fig. 8, it is possible to modify the saved file configuration.
By selecting “Delete Input File” button in the panel of Fig. 8, it is possible to delete the saved configuration for an input file.
Finally, the buttons “Up” and “Down” in the panel of Fig. 8 can be used to sort listed input files (already configured) as they will appear in the sequence shown in the DAMEWARE web app model configuration tab during an experiment setup process.

3.5 Output File Setup

Each time an experiment is completed, involving one or more use cases, its outcome consists in a series of output files, images, plots, diagrams, tables, trained weights, logs etc.

Through the panel shown in Fig. 10, the user can configure and insert the list of output files as foreseen to be produced for each model use case. The figure reports an example in the Train use case (the same for the others). Only the files specified here will be exposed as output available to be downloaded in the DAMEWARE web app experiment status window (see Fig. 9).



Rename	Workspace	Upload	Experiment	Delete
	primo			✖
	mlpqnaExp			✖
	mlpexp			✖
	phat			✖
	clashRegr			✖
	tx			✖
	img1			✖
	bestarsqna			✖
	4survey			✖

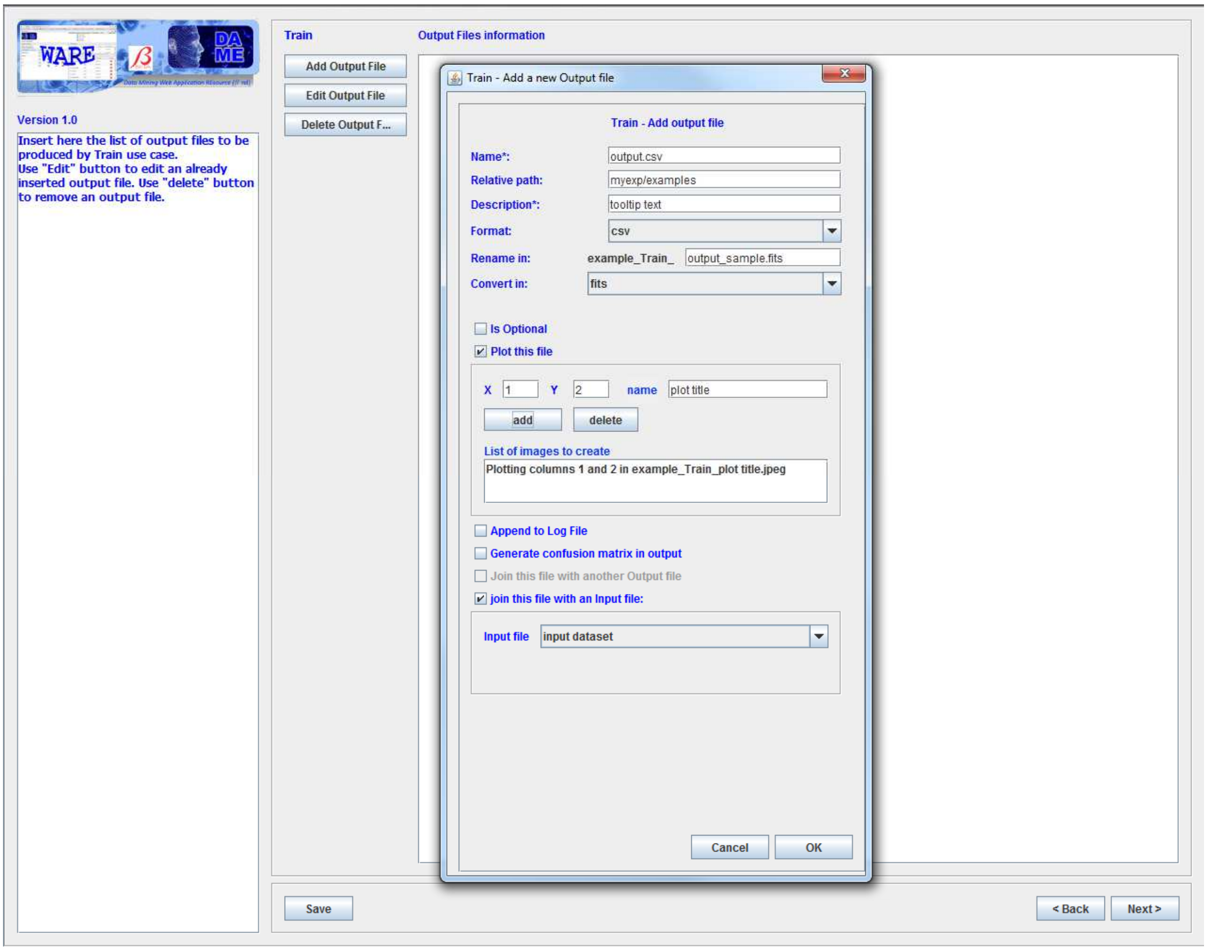
Workspace:
mlpqnaExp

Workspace:
mlpqnaExp

Experiment	Status	Last Access	Delete
classtrain1	ended	2013-07-02	✖
classtest1	ended	2013-07-02	✖
classrun1	ended	2013-07-02	✖
classfull1	ended	2013-07-02	✖
regrtrain1	ended	2013-07-02	✖
regrtest1	ended	2013-07-02	✖
regrrun1	ended	2013-07-02	✖
regrfull1	ended	2013-07-02	✖

Download	AddInWS	File	Type	Description
		mlpqna_TRAIN_error.txt	ASCII	ASCII report on training iteration er
		mlpqna_TRAIN_output.txt	ASCII	Training output (with input dataset
		mlpqna_TRAIN_trainTestOutLog.tx	ASCII	Training output vs target values
		mlpqna_TRAIN_frozen_net.txt	ASCII	Trained network setup file
		mlpqna_TRAIN_weights.txt	ASCII	Trained network weights file

Fig. 9 – The DAMEWARE web app experiment status window, containing list of outputs



Version 1.0

Insert here the list of output files to be produced by Train use case. Use "Edit" button to edit an already inserted output file. Use "delete" button to remove an output file.

Train - Add a new Output file

Train - Add output file

Name*: output.csv
Relative path: myexp/examples
Description*: tooltip text
Format: csv
Rename in: example_Train_ output_sample.fits
Convert in: fits

☐ Is Optional
☑ Plot this file

X 1 Y 2 name plot title

add delete

List of images to create
Plotting columns 1 and 2 in example_Train_plot title.jpeg

☐ Append to Log File
☐ Generate confusion matrix in output
☐ Join this file with another Output file
☑ join this file with an Input file:

Input file input dataset

Cancel OK

Fig. 10 – Example (Train use case) of the output file setup panel

3.5.1 Add new output files

By selecting “Add Output File” in the panel of Fig. 10, the new panel shown allows the user to describe the output file information.

- **Name***: the name of output file as generated by your own model code;
- **Relative path**: optional relative path (sub-directory) where output file will be created (basically the experiment location used internally by the DAMEWARE web app);
- **Description**: parameter tooltip text;
- **Format**: the internal representation of the file (it is basically to assign a right extension at the creation time for further use in the web app). It must match the real type as used by your model code;
- **Rename in**: All the files produced by the model will be renamed in <plugin>_<running mode>_<new name>. In this text area you can choose the final name of output file as it will appear in the web app experiment status panel. If this element is empty, the output file name will be <plugin>_<running mode>_<original name>;
- **Convert in**: the type to be converted the output file. Unselect it if none conversion is required;
- **Plot this file**: select this checkbox only if it is required a plot for the file contents. If selected a pop-up tab will appear asking to configure plot features (Fig. 10):
  - **X**: file column index[2] to be used as the abscissa;
  - **Y**: file column index to be used as the ordinate;

[2] Here index is the ordinal number referred to the column position in the file, not the meta-datum.

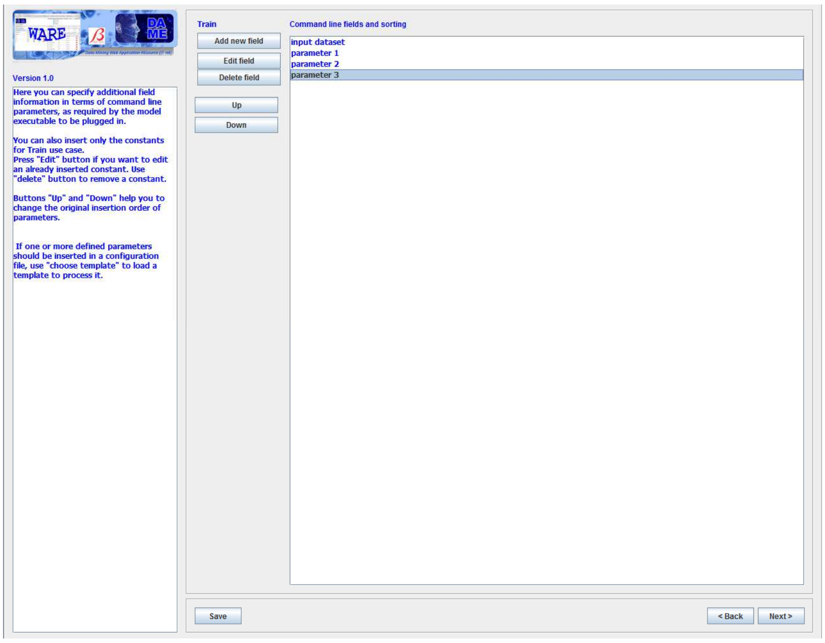
  - **Name**: this field specifies the name of the parameter used as identifier for the produced image. (Example: X = 1, Y = 2, name = plotting1); Then the internal identifier of the image will be named plottingImage1.
  - **List of images to create**: a field where to insert the list of all the images to be generated from a specified output file. describing the images;
- **Append to Log File**: it specify if the file content should also be included in the log file;
- **Generate confusion matrix in output**: the output values of the experiment (table of data) will be also saved as a confusion matrix (<u>only in case of classification type experiments</u>);
- **Join this file with another output file**: Join it with columns from another output file. If this checkbox is selected, you can choose the output file to join;
- **Join this file with an input file**: if selected the user must enter the type of the input file that will be merged with the output file. The resulting output file will contain joined input and output columns.

3.5.2 Edit and remove an output file

By selecting "Edit Output File" button in the panel of Fig. 10, it is possible to modify the saved file configuration.
By selecting "Delete Output File" button in the panel of Fig. 10, it is possible to delete the saved configuration for an output file.

3.6 Command line constant parameters Setup

The "Additional fields and sorting" panel, shown in Fig. 11 (as example for the Train use case) allows user to specify additional field information in terms of command line constant parameters or configuration files constant parameters (if previously created, see Fig. 12), as requested by the model executable to be plugged in.



Here you can specify additional field information in terms of command line parameters, as required by the model executable to be plugged in.

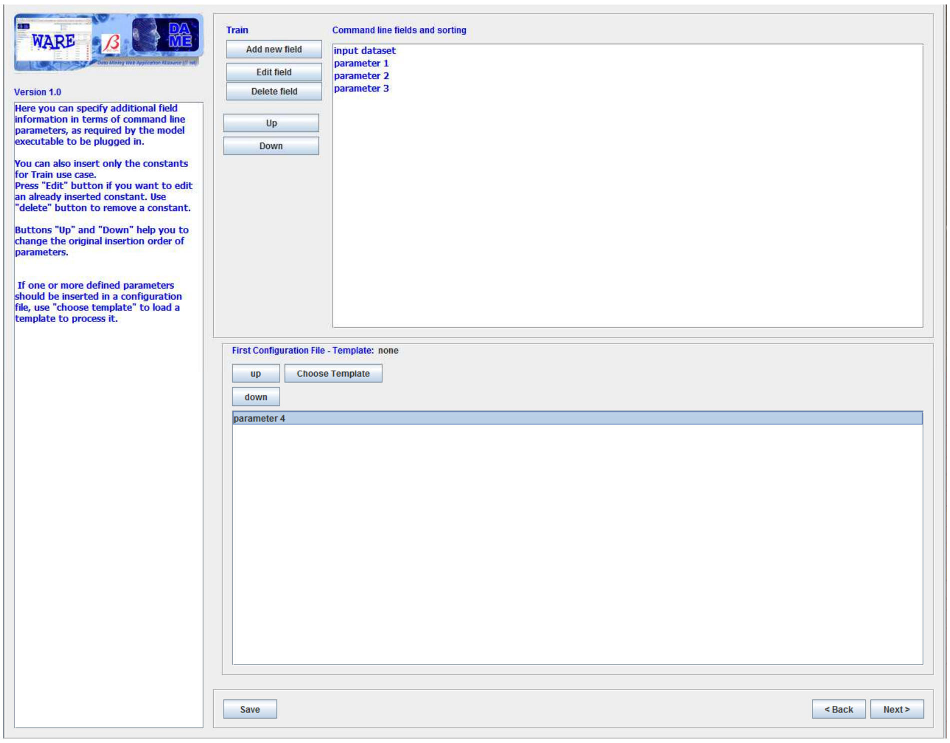
You can also insert only the constants for Train use case.
Press "Edit" button if you want to edit an already inserted constant. Use "delete" button to remove a constant.

Buttons "Up" and "Down" help you to change the original insertion order of parameters.

If one or more defined parameters should be inserted in a configuration file, use "choose template" to load a template to process it.

input dataset
parameter 1
parameter 2
parameter 3

Fig. 11 – Example (Train use case) of "additional fields and sorting" setup panel – Command Line case



Version 1.0

Here you can specify additional field information in terms of command line parameters, as required by the model executable to be plugged in.

You can also insert only the constants for Train use case.
Press "Edit" button if you want to edit an already inserted constant. Use "delete" button to remove a constant.

Buttons "Up" and "Down" help you to change the original insertion order of parameters.

If one or more defined parameters should be inserted in a configuration file, use "choose template" to load a template to process it.

Command line fields and sorting

input dataset
parameter 1
parameter 2
parameter 3

First Configuration File - Template: none

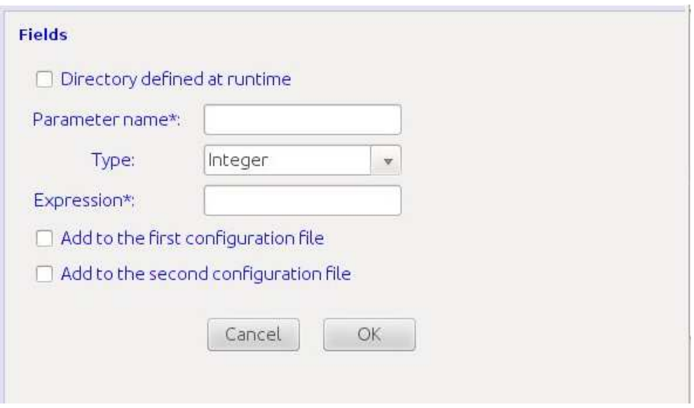
parameter 4

Fig. 12 Example (Train use case) of “additional fields and sorting” setup panel – Configuration Files case

If one or two configuration files, for each of it, you can choose a Template (if available) to create the configuration file, using button “Choose Template”.

3.6.1 Add a constant field

By selecting “Add new field” in the panel of Fig. 11 it is possible to add a new command line constant parameter (Fig. 13). This can be done by describing its name (identifier of the parameter), type and the expression.
If the model provides to pass the name of an output directory (defined at runtime) you can use checkbox “Directory defined at runtime”.



Fields

☐ Directory defined at runtime

Parameter name*:

Type: Integer

Expression*:

☐ Add to the first configuration file

☐ Add to the second configuration file

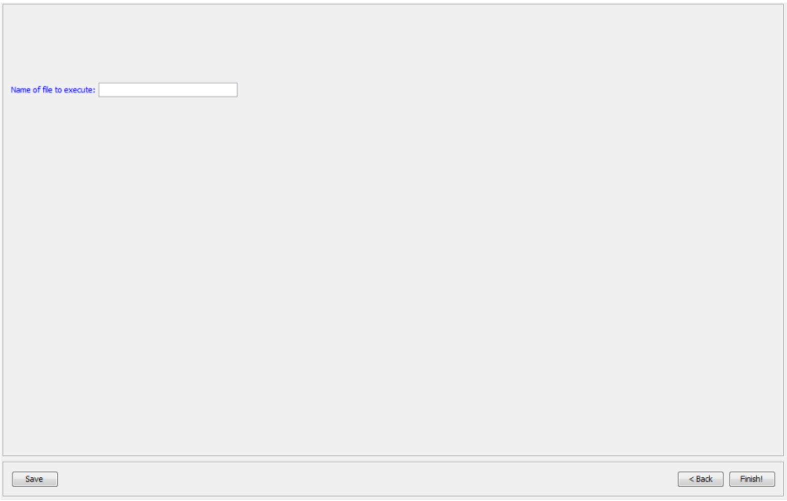
Fig. 13 – window to add a constant field (either as command line parameter or configuration file entry)

3.6.2 Edit, sort and remove a constant field

After creation of a new constant parameter, it can be modified by choosing “Edit field” in the panel of Fig. 11.
Of course, in a command line it is important the order in which the parameters are listed (otherwise the execution will crash). So far, to correctly locate the constant parameter position in the command line, the buttons “Up and “Down” of Fig. 11 can be used.
Finally the option “Delete field” in Fig. 11 allows user to delete a constant parameter.

3.7 Final Steps of the setup procedure

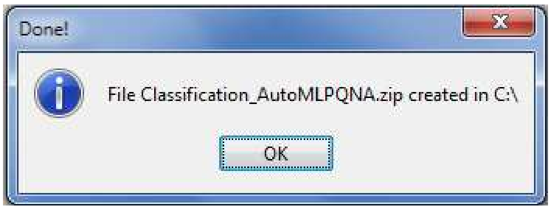
At the end of the setup procedure, by having configured input and output files, constant and parameters of command lines for all the foreseen use cases of the model to be plugged in, the panel shown in Fig. 14 asks for the final name of the executable program to be launched if the plugin will be selected for an experiment through the DAMEWARE web app GUI.



Name of file to execute:

Fig. 14 – final panel

The “Save” button pressed here, will save all the performed steps of the plugin creation procedure.
You can press Finish button only after compiling “Name of file to execute” field, to generate the final plugin code to be integrated within the web app.
After the “Finish” button is clicked the procedure will create all internal features to be physically integrated into the DAMEWARE web app infrastructure.
The “Finish” button will activate a new panel asking for the directory where to store all created files of the plugin package.
The Fig. 15 reports the disclaimer which informs about the compressed archive successful creation.



Done!

File Classification_AutoMLPQNA.zip created in C:\

Fig. 15 – example of message notifying the successful creation of the compressed plugin package

3.8 Delivery of the plugin creator

At the end of the guided procedure, for instance when the user selects the button “Finish” of Fig. 14, the plugin wizard produces a compressed package (.zip) called <functionality>_<model acronym>.zip, containing:

- ✓ A Java file named <functionality>_<model acronym>.java;
- ✓ A Java file named <model acronym>.java;
- ✓ An XML file named <functionality>_<model acronym>.xml;
- ✓ An XML file named ConfigurationFile.xml that tracks all the setup work done.

The user must deliver the .zip file, together with the source code of the model (to be recompiled on the worker nodes of DAME computing platform), to the DAMEWARE administrators (helpdame@gmail.com), asking to complete the model plugin integration task.
Then the user will be informed about the availability of the new model in the new release of DAMEWARE web app.

4 Appendix – References and Acronyms

Abbreviations & Acronyms

A & A	Meaning	A & A	Meaning
AI	Artificial Intelligence	KDD	Knowledge Discovery in Databases
ANN	Artificial Neural Network	IEEE	Institute of Electrical and Electronic Engineers
ARFF	Attribute Relation File Format	INAF	Istituto Nazionale di Astrofisica
ASCII	American Standard Code for Information Interchange	JPEG	Joint Photographic Experts Group
BoK	Base of Knowledge	LAR	Layered Application Architecture
BP	Back Propagation	MDS	Massive Data Sets
BLL	Business Logic Layer	MLC	Multi Layer Clustering
CE	Cross Entropy	MLP	Multi Layer Perceptron
CSOM	Clustering SOM	MSE	Mean Square Error
CSV	Comma Separated Values	NN	Neural Network
DAL	Data Access Layer	OAC	Osservatorio Astronomico di Capodimonte
DAME	DAta Mining & Exploration	PC	Personal Computer
DAMEWARE	DAME Web Application REsource	PI	Principal Investigator
DAPL	Data Access & Process Layer	REDB	Registry & Database
DL	Data Layer	RIA	Rich Internet Application
DM	Data Mining	SDSS	Sloan Digital Sky Survey
DMM	Data Mining Model	SL	Service Layer
DMS	Data Mining Suite	SOFM	Self Organizing Feature Map
FITS	Flexible Image Transport System	SOM	Self Organizing Map
FL	Frontend Layer	SW	Software
FW	FrameWork	UI	User Interface
GRID	Global Resource Information Database	URI	Uniform Resource Indicator
GSOM	Gated SOM	VO	Virtual Observatory
GUI	Graphical User Interface	XML	eXtensible Markup Language
HW	Hardware		

Tab. 1 – Abbreviations and acronyms

__oOo__