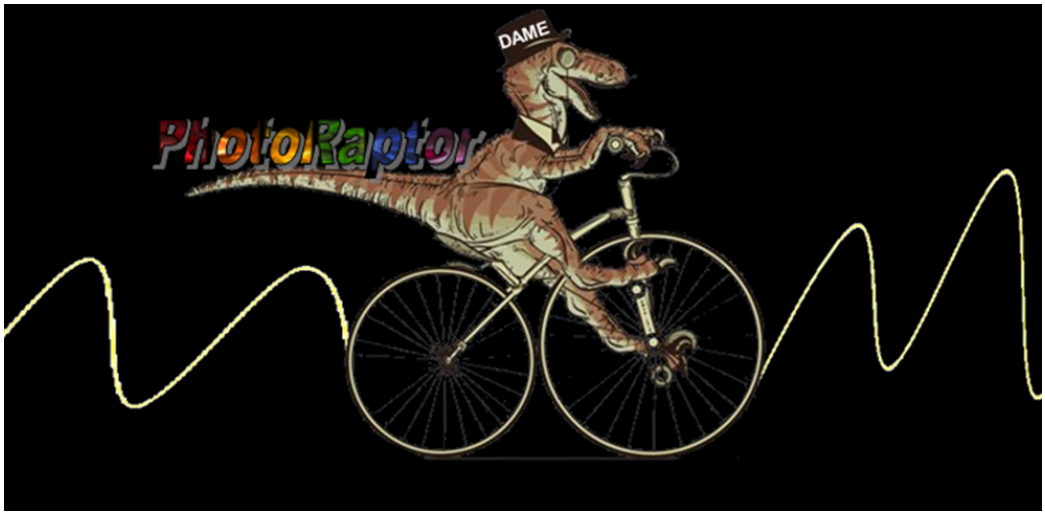
DAta Mining & Exploration

we make science discovery happen

PhotoRaptor

Photometric Research Application To Redshifts

User Manual
Release 1.1



INAF-OACN

IVOA

Contents

Chapter 1

Introduction

Measurements for fainter objects are nowadays accessible by photometry thanks to the improving telescope technology. This makes photometric redshift (photo-z) extremely attractive for observing programmes depending on redshift especially with the advent of modern panchromatic digital surveys. In fact, future large-field public imaging projects, such as KiDS[1] (Kilo-Degree Survey), DES (Dark Energy Survey, The Dark Energy Survey Collaboration 2005), LSST (Large Synoptic Survey Telescope, Ivezic et al. 2009) and Euclid (Euclid Red Book, 2011) require extremely accurate photo-z to obtain accurate measurements that do not compromise the survey's scientific goals.

Due to the necessity to evaluate photo-z for a variety of huge sky survey data sets, it seemed important to provide the astronomical community with an instrument able to fill this gap. Besides the problem of moving massive data sets over the network, another critical point is that a great part of astronomical data is stored in private archives that are not fully accessible on line. So, in order to evaluate photo-z it is needed a desktop application that can be downloaded and used by everyone locally, i.e. on his own personal computer or more in general within the local intranet hosted by a data center.

The name chosen for the application is **PhotoRApToR**, i.e. **Photo**metric **R**esearch **Ap**plication **To R**edshift.
It embeds a machine learning algorithm and special tools dedicated to pre- and post-processing data. The ML model is the MLPQNA (Multi Layer Perceptron trained by the Quasi Newton Algorithm), which has been revealed particularly powerful for the photo-z calculation on the base of a spectroscopic sample (Cavuoti et al. 2012; Brescia et al. 2013a,b; Biviano et al.

[1] `http://www.astro-wise.org/projects/KIDS/`

2013).

In order to favor the portability of this tool, the Graphical User Interface was developed in Java[2] language and runs on top of a standard Java Virtual Machine (JVM), thus permitting its execution in a platform-independent way.

Main features of the presented application can be summarized as follows:

- *Data table manipulation*: in order to navigate throughout user's data sets and related *metadata*, as well as to prepare data tables to be submitted for experiments, there are several options to perform the editing, ordering, splitting and shuffling table rows and columns. A special set of options is dedicated to the missing data retrieval and handling, for instance Not-a-Number (NaN) or not calculated/observed parameters in some data samples;

- *Classification experiments*: the user can perform general multi-class classification problems, i.e. automatic separation of an ensemble of data by assigning a common label to an arbitrary number of their subsets, each of them grouped on the base of a hidden similarity. The classification here is intended as *supervised*, in the sense that there must be given a subsample of data for which the right label has been previously assigned, based on the *a priori* knowledge about the treated problem. The application will learn on this known sample to classify all new unknown instances of the problem;

- *Regression experiments*: the user can perform general regression problems, i.e. automatic learning to find out an embedded and unknown analytical law governing an ensemble of problem data instances (patterns), by correlating the information carried by each element (features or attributes) of the given patterns. Also the regression is here intended in a *supervised* way, i.e. there must be given a subsample of patterns for which the right output is *a priori* known. After training on such KB, the program will be able to apply the hidden law to any new pattern of the same problem in the proper way;

- *Photo-z estimation*: within the *supervised* regression functionality, the application offers a specialized toolset, specific for photometric redshift estimation, able to learn the hidden correlation between photometric and spectroscopic information on a subset of sky objects (patterns of

[2] `http://www.oracle.com/technetwork/java/index.html`

the KB), for which the spectroscopic redshift is available. After training, the system will be able to predict the right photo-z value for any new sky object belonging to the same type of KB;

- *Data visualization*: the application includes some $2D$ and $3D$ graphics tools, for instance multiple histograms, multiple $2D/3D$ scatter and line plots. Such tools are often required within astrophysical problems to visually analyze and explore data distributions and trends, as well as resulting from data mining experiments;

- *Data statistics*: all classification and regression experiments provide a statistical report about their output. In the first case, the typical confusion matrix (Stehman, 1997) is given, including related statistical indicators such as *classification efficiency*, *completeness*, *purity* and *contamination* for each of the classes defined by the specific problem (see Sect. 7.1.1 for details). For what the regression is concerned, the application offers a dedicated tool, able to provide several statistical relations between two arbitrary data vectors (usually two columns of a table), such as average, standard deviation (σ), Root Mean Square (RMS), Median Absolute Deviation (MAD) and the *Normalized* MAD (NMAD, Hoaglin et al. 1983), the latter specific for the photo-z quality estimation, together with percentages of *outliers* at different multiples of σ (Brescia et al., 2013b; Ilbert et al., 2009).

In Fig. 1.1 the layout of a general PhotoRApToR experiment workflow is shown. It is valid for either regression and classification cases.

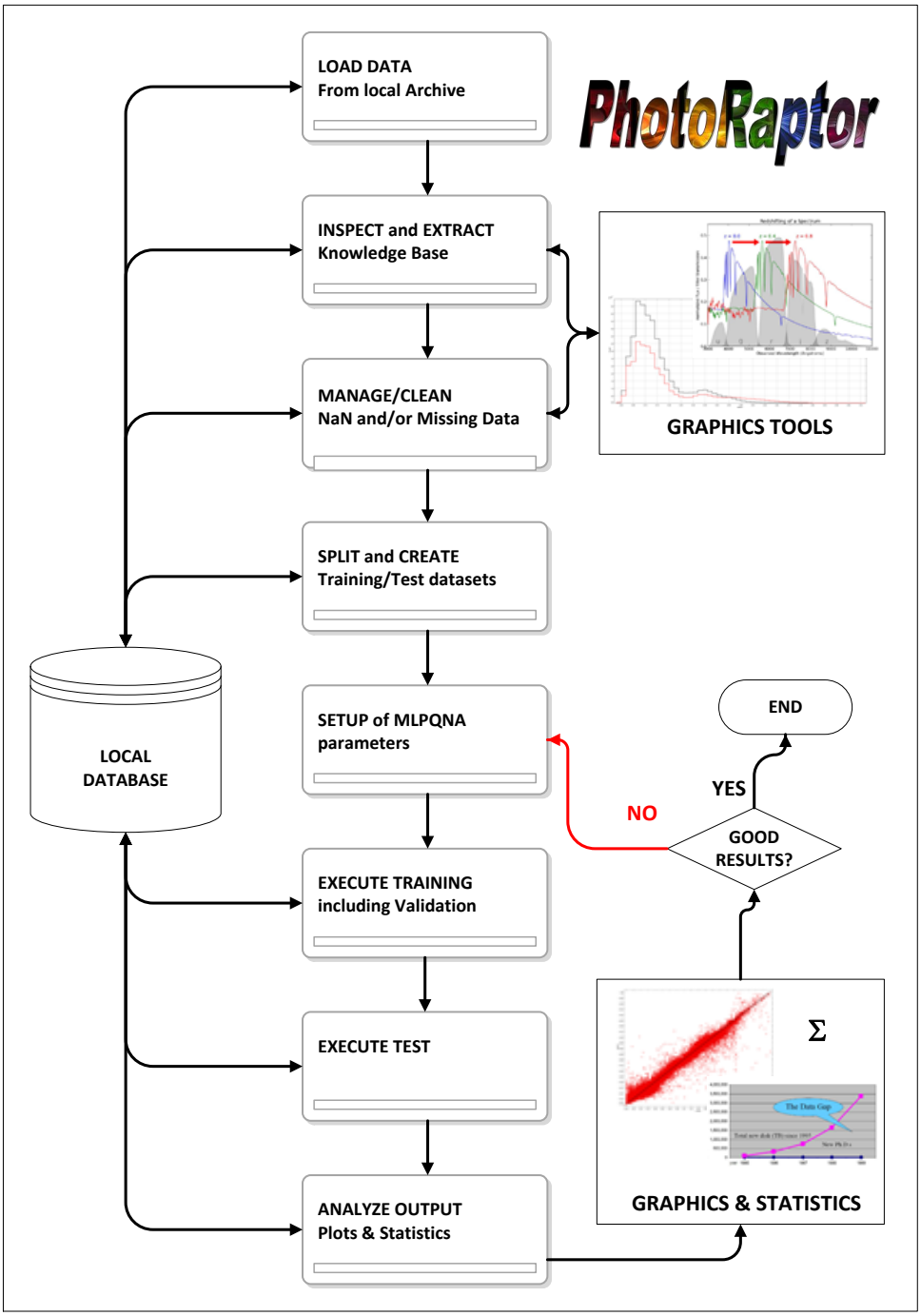


Figure 1.1: The workflow of a generic experiment with PhotoRApToR.

Chapter 2

Installation

2.1 Download and System Requirements

The PhotoRApToR program package is available at the official website (`http://dame.dsf.unina.it/dame_photoz.html#photoraptor`). Here there are different downloadable versions (zipped files) for different OS (Operative System) platforms:

- $WIN7$: (archive name $PhotoRApToR\_win7.zip$), package for MS Windows 7, generic platform type;
- $WIN8$: (archive name $PhotoRApToR\_win8\_64.zip$), package for MS Windows 8, $64-bit$ platform;
- $UBUNTU64$: (archive name $PhotoRApToR\_Ubuntu\_64.zip$), package for Linux Ubuntu v12.04, $64-bit$ platform;
- $SL64$: (archive name $PhotoRApToR\_SL6\_64.zip$), package for Scientific Linux 6, $64-bit$ platform;
- $MacOSX10.7.5(Lion)$: (archive name $PhotoRApToR\_MacLion.zip$), package for Mac OS X Lion;
- $MacOSX10.9.2(Mavericks)$: (archive name $PhotoRApToR\_MacMavericks.zip$), package for Mac OS X Mavericks;

Several other versions could be made available on request.

In terms of system requirements, the user must have previously installed and verified the JVM, available at the official site `http://www.oracle.com/technetwork/java/index.html`. No other specific requirements are needed.

2.2 Installation Procedure

After having chosen and downloaded the package, the following are the main steps to install it.

1. unzip the package compressed archive in a user selected relative path, hereinafter called as $< user\_path >$.

2. **IMPORTANT NOTE**: For the $WIN7$ and $WIN8$ versions we inform that when the program is launched for the first time, there will be automatically created a working directory, named *PhotoRApToR*, in the path *C:*. Here all further experiment executions and results will be stored.

3. Launch the Java executable file *PhotoRApToR* (for its location see the Sec. 2.3).

2.3 Verification

After having downloaded and installed the software package on your machine, in the chosen relative path $< user\_path >$ you would have a directory called *PhotoRApToR*. This is the parent directory hosting all files as needed to run the program. In particular under the parent directory you would be able to find the following directory tree and contents:

PhotoRApToR jar file the main program Java executable. This is the file to be launched to run the program.

lib the directory containing all libraries and executables. **Don't change or remove its contents**. The included files should be:

- **.Jar files:** a series of Java executable files. They may change according the specific version installed, depending on your local OS;
- **.Exe files:** a series of executable files. They may change according the specific version installed, depending on your local OS;
- **.Dll files:** a series of dynamic library files. They may change according the specific version installed, depending on your local OS;

resources the directory containing some package internal utilities. **Don't change or remove its contents**. The included files should be:

.Png files: a series of png type image files and icons internally used by the program. They may change according the specific version installed, depending on your local OS;

.Pdf files: at least the user manual of the program (this document), should be present;

TEST the directory containing some additional files. The included files should be:

.Csv files: a series of CSV type files, useful to be used as examples of data files for experiments during the initial exploration of the program facilities. In particular at least one file useful for regression and one for classification exercises should be present.

In the case of $WIN7$ and $WIN8$ releases, after first launch of the program, under the path $C:$ you will find a directory $PhotoRApToR$, containing a subdir:

MyExp the directory which can be used as destination path for user experiments. At the beginning this directory is empty. Whenever the user runs an experiment, this directory will be populated by sub-directories with name based on date and time of the experiment execution. At the user convenience, all experiment sub-directories can be removed and/or edited without problems.

For the $Ubuntu64$ release this subdir will be located in the same relative destination path of the installation package.

In order to verify the correct execution of the program, we suggest the user to inspect the program parent directory tree to check the presence of all sub-directories and files as described above. Then the user can launch the PhotoRApToR Java executable to see if the program correctly runs. As first action, we suggest the user to try all menu options, by previously loading at least one data table file as first example.

In case of any wrong behavior or failure, please contact us through the e-mail *helpdame@gmail.com*, by specifying details of the problem and current installed version.

Chapter 3

The program menu options

The main window of the PhotoRApToR application (see Fig. 3.1) is divided in three parts.
The first one is the **Menu Bar** with a **Button Bar** below, the second one is the **Table List** on the left and the third one is the panel on the right with **Table Properties**, **Table Editor** and the **Split Panel** below.

Beginnig from the *Menu Bar*, it is possible to decribe all the commands:

File is the menu from which to launch standard commands to open or save files. The following options are shown:

- **Load Table:** opens a new dialog where it is possible to select table format and file;
- **Discard Table:** allows to erase a table item from the *Table List*;
- **Save Table:** saves the selected table using a Browse Dialog;
- **Exit**

Table is a menu containing the commands that allow to see and modify the table properties:

- **Table Data:** it opens the selected table in a new window;
- **Table Metadata:** shows only the column's metadata for the selected table;
- **Row Shuffle:** the selected table rows are shuffled and the new table is opened in a new window;
- **Not a Number:** it opens a new window for managing the table data in order to remove the Not a Number elements (e.g. values like -9999) from the dataset;

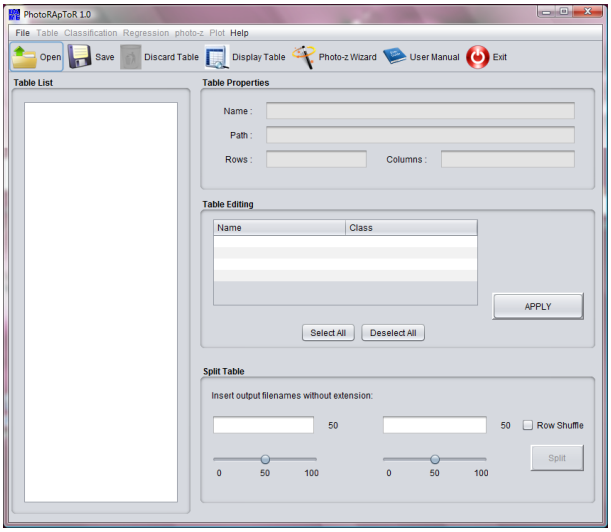


Figure 3.1: The PhotoRApToR main window.

Classification is a menu where are selectable different options that allow to execute different experiments:

each one of the options **Train, Test** or **Run** opens a new window where it is possible to configure and execute experiments using the model MLPQNA as engine;

Regression is a menu with other experimental options:

the options **Train, Test** and **Run** are the same of those present in the Classification menu;

Statistics: in a new window the user can select the Target column and the Output column to generate related statistics;

Outliers: it opens a new window where the user can analyze outliers of a loaded data table and generate a dataset without outliers by setting statistical parameters;

photo-z performs experiments dedicated to the evaluation of photometric redshift in a new window. It sets MLPQNA parameters and generates an output table and the related statistics;

Plot is the menu that shows three different ways to generate data plots:

Histo Plot: it opens a new window where to create single or multiple histograms;

Scatter Plot: it opens a new window where to create single or multiple scatter plots. It contains several parameters to set up, like for instance the type of line plot or the marker for the data points;

3D Plot: after the selection of three table columns and the setup of parameters, a cube plot is generated with also the possibility to change the angle of view;

Help is the menu with manuals and program's credits;

User's Manual: it opens this document;

Open Wizard: it runs the Primer Wizard guided procedure;

Website: the program info site (`http://dame.dsf.unina.it/dame_photoz.html#photoraptor`);

About: the application information;

The *Quick Button Menu bar*, located under the main menu bar, allows a fast access to the main functions of the application.

Open is a button that opens a dialog for the selection of table format and the browsing of files (see Fig. 4.1);

Save opens a dialog to save the tables;

Discard Table removes selected data tables from the application table list (not removing the physical table file);

Display Table shows the whole table dataset in a new window;

Photo-z Wizard starts the Primer Wizard window;

User Manual opens this document;

EXIT closes the application and exits.

Chapter 4

Pre-processing

The evaluation of photo-z is made possible by the existence of a rather complex and not analytically known correlation existing among the fluxes, as measured in broad band photometry, the morphological types of the galaxies, and their distance. The search for such a correlation (a nonlinear mapping between the photometric parameter space and the redshift values) is particularly suited to data mining methods. For data sets in which accurate and multi-band photometry for a large number of objects is complemented by spectroscopic redshifts, and for a statistically significant sub-sample of the same objects, the *empirical methods* offer greater accuracy. These methods use the sub-sample of the photometric survey with spectroscopically-measured redshifts as a training set to constrain the parameters of a fit mapping the photometric data as redshift estimators.

The fundamental premise to use this tool is that the user must preliminarily know how to represent its data. As trivial as it might seem, it is worth to explicitly state that depending on the ML method employed, the user must: *(i)* be conscious of the target of his experiment, such as for instance a regression or classification; and *(ii)* possess a deep knowledge of the characteristics and of the meaning of his data.

The first step is to open a table by selecting the file format and by browsing it through the Load Dialog (Fig. 4.1).

In order to reach an intelligible and homogeneous representation of data sets, it is mandatory to preliminarily take care of their internal format to transform pattern features and force them to assume a uniform representation before submitting them to the training process. In this respect real working cases might be quite different among themselves.

PhotoRApToR can ingest and/or produce data in any of the following

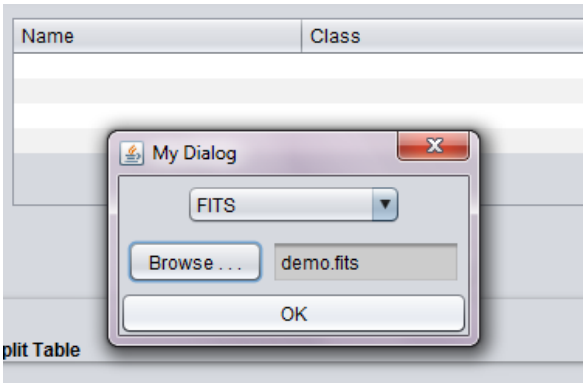


Figure 4.1: The Load Table dialog.

supported formats:

- FITS (Wells et al., 1981): tabular/image;
- ASCII (ANSI et al., 1977): ordinary text, i.e. space separated values;
- VOTable[1]: VO compliant XML-based documents;
- CSV (Repici, 2010): Comma Separated Values;

In the Load Dialog a drop down menu allows to select the file format. By clicking on the **Browse** button, it is possible to search the local dataset.

For this description a file named "**demo.fits**" was choosen: every time a new table is loaded, a new item, with its table name, is added to the *Table List*.

4.1 User data handling

The PhotoRApToR core engine is the MLPQNA neural network. In this respect, before launching any experiment, it may be necessary to manipulate data in order to fulfill the requirements in terms of training and test patterns (data set rows) and features (data set columns) representation as well as contents: *(i)* either training and test data files must contain the same number of input and target columns, in the same order; *(ii)* the target columns must always be the last columns of the data file; *(iii)* the input columns (features) must be limited to the physical parameters, without any other type of additional columns (like column identifiers, object coordinates etc.); *(iv)* all input data must be numerical values (no categorical entries are allowed).

[1]http://www.ivoa.net/documents/VOTable/

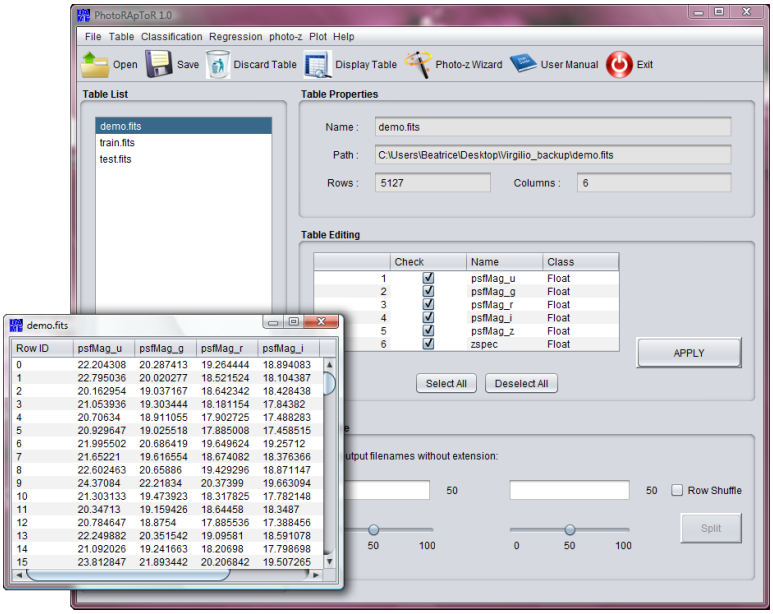


Figure 4.2: Every dataset in the table list can be displayed in a new Table Window.

The application makes available a set of specific options to inspect and modify data file entries. Selecting one item from the *Table List*, all the table properties are displayed inside the right panel: in particular the table name, its complete path and the number of columns and rows. With a double click on the table name in the Table List or by clicking the **Display Table** button it is possible to open a new window showing the complete dataset.

4.1.1 Data Feature Selection

A fundamental step for any machine learning experiment is to decide which features to use as input attributes for patterns to be learned. In the specific case of photo-z estimation from a spectroscopic KB, from the available data it is thus necessary to inspect and check which types of flux combinations would be more effective, in terms of available photometry (number and type of fluxes, bands, magnitudes or related colors).

In practical sense, one has to try to maximize the information carried by hidden correlations among different bands, magnitudes and zspec available. In spite of what can be thought, not always the maximum number of available parameters should be used to train a machine learning model. The experience has demonstrated that it is more the quality of data, than the quantity of features and patterns, the crucial key to obtain best predic-

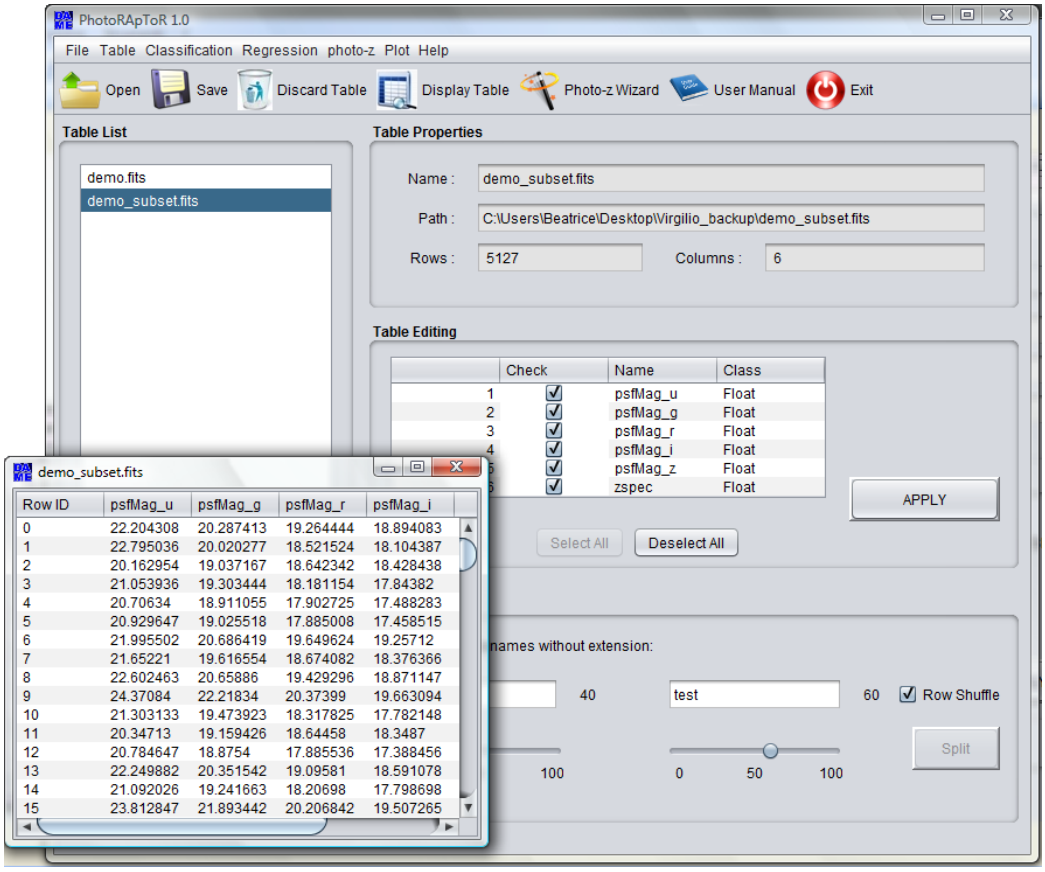


Figure 4.3: Table subset creation. With the click of the **Apply** button a new dataset has been generated, with only the original dataset features in the columns selected in the Edit Table panel.

tion results (Brescia et al., 2013a). Of course, it depends on how wide is the variety of photometric bands and magnitudes for which a high quality of zspec entries is available in the KB. As usual the cross-matching among different surveys makes available a wider number of photometric bands, but sometimes drastically reducing the number of objects available for training (i.e. the KB). But if the related photometry quality is sufficiently high, this is the best way to obtain good performances.

In the *Edit Table* panel all column *meta-data* for the selected table are displayed and with them it is possible to generate a subset of the table containing the needed columns only. After the selection of desired columns using the related checkboxes, a table subset is created by clicking the **Apply** button.

4.1.2 Data Editing

The random shuffling operation is useful to avoid systematic trends during training and to ensure homogeneity in the distribution of training and test patterns. This last property is, in fact, directly connected to the necessity to split initial data into separated sets, respectively for training and for testing phases. This is a simple action made possible by the *Split* option. When the table is selected in the *Table List*, we must choose two different names for the split files (in this case *train* and *test*) and two different percentages of the original data set[2].
By clicking on **Split** button, the two split datasets are generated. If the selectable checkbox **Row Shuffle** is selected, the two datasets are also randomly shuffled by rows before to split them, and added to the *Table List* (Fig. 4.4), ready for the next phase.

There is no any analytical rule to decide the percentages of the splitting operation. According the direct experience, an empirical rule of thumb suggests to use 80% and 20% for training and test respectively (Kearns, 1996). But certainly it depends on the initial amount of available KB. For example also 60% vs 40% and 70% vs 30% could be in principle used in case of large datasets (over ten thousand patterns). It depends also on the quality of available KB. When both photometry and spectroscopy are particularly clean and precise (i.e. with a high S/N), there could also be possible to obtain high performances by training on the half of the KB. The more patterns are available for test, the more precise is the statistical knowledge about experiment performance. But this straightforward rule is valid only in case of a sufficiently large KB, i.e. without affecting the number of patterns necessary to train the network.

[2]It is important to observe that for machine learning supervised methods three different subsets for every experiment would be generally required from the available KB: *(i)* (*training set*) to train the method in order to acquire the hidden correlation among the input features; *(ii)* the (*validation set*), used to check and validate the training in particular against the loss of generalization capabilities (a phenomenon also known as overfitting); and *(iii)* the (*test set*), used to evaluate the overall performances of the model (Brescia et al. 2013a). Within the implemented version of MLPQNA model in the PhotoRApToR application, the validation is embedded into the training phase, by means of the standard leave-one-out k-fold cross validation mechanism (Geisser 1975).

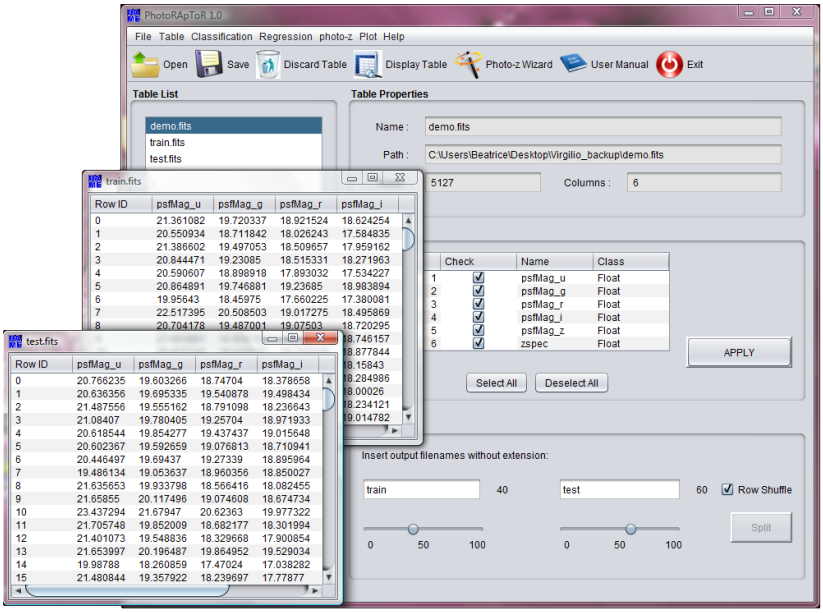


Figure 4.4: Use of the Split tool. After selecting the table to be split, two different names are choosen for the files and the sliders are dragged to select a different percentage. By clicking on Split button the two split datasets are generated.

4.2 Not a Number

It may be frequent that a data table may have empty entries (sparse matrix) or missing data (lack of observed values for some features in some patterns). Missing values (Marlin, 2008) are frequently identified by special entries in the patterns, like Not-A-Number, out-of-range, negative values in a defined positive numeric field, etc. Missing data is among the most frequent sources of perturbation in the learning process, causing confusion in classification experiments or mismatch in regression problems. This is especially true for astronomy where inaccurate or missing data are not only frequent, but very often cannot be simply neglected, since they carry useful information. To be more specific, missing data in astronomical databases can be of two types:

Type I: true missing data which were not collected. For instance a given region of the sky or object was not observed in a given photometric band thus leading to a missing information. These missing data may arise also from the simple fact that data, coming from any source and related to a generic experiment, are in most case not expressly collected for DM purposes and, when originally gathered, some features were not considered relevant and thus left unchecked.

Type II: upper limits or non-detections (i.e. object too faint to be detected

in a given band). In this case the missing datum conveys very useful information which needs to be taken into account into the further analysis. It needs to be noticed, however that, often upper limits are not measured in absence of a detection and therefore this makes these missing data undistinguishable from Type I.

In some cases, inaccurate values can be related to systematic data errors, for example due to an hardware setup condition in the data collecting instrument. However, this is usually trivial to be recovered, if the user has knowledge about the collecting device conditions, but in any case a deep care about the presence of inaccurate values should be required whenever a DM process is approached, and a close interaction with a domain expert helps in preventing wrong results in the experiment.

In other words, missing data in a data set might arise from unknown reasons during data collecting process (Type I), but sometimes there are very good reasons for their presence in the data since they result from a particular decision or as specific information about an instance for a subset of patterns (Type II). This fact implies that a special care needs to be put in the analysis of the possible presence (and related causes) of missing values, together with the decision on how to submit these missing data to the ML method, in order to take into account such special cases and to prevent wrong behaviors in the learning process.

In principle, data entries affected by missing attributes, i.e. patterns having fake values for some features, may be used within the KB for a photo-z experiment. In particular they can be used to differentiate the data sets with an incremental quantity of affected patterns, useful to evaluate their noise contribution to the performance of the photo-z estimation after training. Theoretically it should be expected that a greater amount of missing data, evenly distributed in both training and test sets, induces a greater deterioration in the quality of the results. This precious information may be indeed used to assign different indices of quality to the produced photo-z catalogue.

The organization of data sets with different rates of missing data can be performed through PhotoRApToR by means of the following options:

A click on **Table>Not a Number** menu item opens a new window. In the upper left panel a drop down menu allows to select the dataset to check and in a text field the user must define the symbol by which missing data are represented in the dataset (i.e. symbols like -9999, -99.0, NaN and so on). Two checkboxes select which columns will be checked: the input features columns (in this demonstration case the photometric input data),

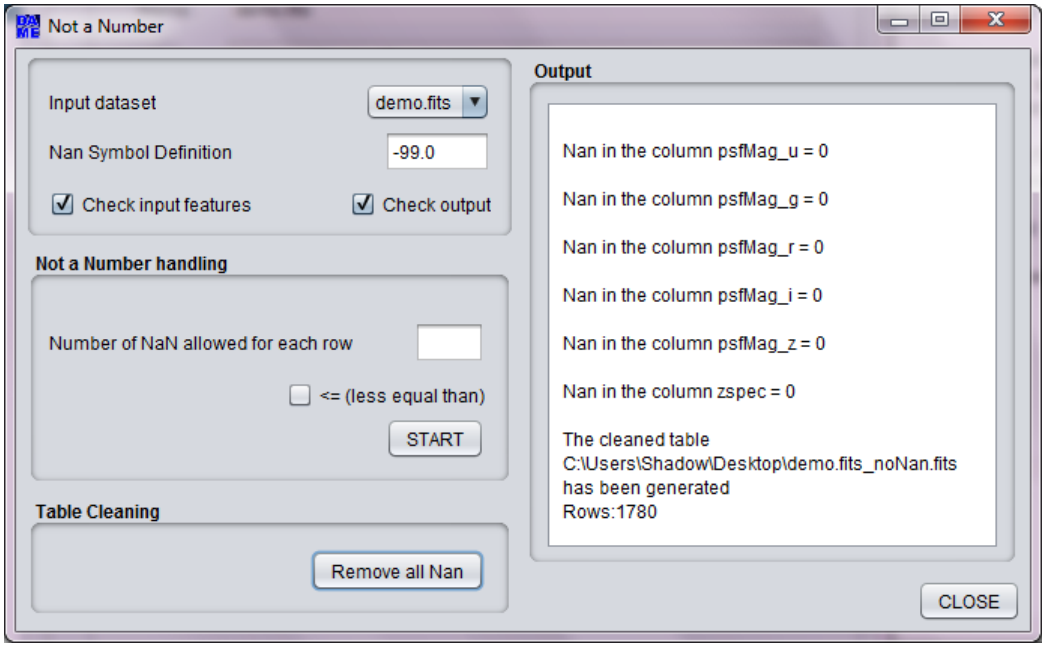


Figure 4.5: Use of the NaN tool. After the definition of NaN symbol in the input dataset, the user can generate a new dataset only with rows containing NaN elements or one cleaned by NaN.

or the output column (in this case, the zspec column).

In the panel below it is possible to set the number of NaN for each row, and by clicking on **START** button a new dataset is generated, builded only with rows containing the chosen number of the NaN. If the $\leq$*(less equal than)* checkbox is selected, the dataset will be generated with all the rows containing less NaN elements than the user has selected.
If the user clicks the **Remove all NaN** button, the tool generates a dataset without all the rows containing NaN elements.
In the Output panel on the right side of Fig. 4.5 the number of NaN for each column and for each row of the input dataset and the path of the output file are reported.

Chapter 5

Photometric redshift estimation

A click on the **photo-z** button opens a new window (Fig. 5.1).
After having prepared the KB (see Chap. 4), the user would have two subset tables ready to be submitted for a photo-z experiment. By looking at the Fig. 1.1 the experiment consists of a pre-determined sequence of steps, for instance *(i)* Training and validation of the model network; *(ii)* blind Test of the trained model network; *(iii)* statistical and visual inspection of results; and *(iv)* Run, i.e. the execution of a well trained, validated and tested network on new data samples.

We outline that for the first two steps, the basic rule is to use different data subsets. In general all empirical photo-z methods may suffer of a poor capability to extrapolate outside the range of distributions imposed by the parameter space and photometric flux limits used for the training. In other words, outside the limits of magnitudes and spectroscopic redshift (zspec) used in the training set, these methods do not ensure optimal performances. But, within the ranges of the training parameter space, the empirical models are able to overtake fitting models, essentially because they do not make any a priori assumption on the physical properties of objects[1]. In order to remain in a safe condition, the user must perform a selection of test data according to the training sample photometric and spectroscopic limits.

Therefore, none of the objects included in the training sample should be included also in the test sample. Moreover only the data set used for the test has to be used to generate performance statistics. In other words the test must be blind, i.e. containing only objects never submitted to the network before.

For what the training is concerned, this phase embeds two processing steps, for instance training of the MLPQNA model network and training

[1] priors like SFR, IMF, metallicity, age etc.

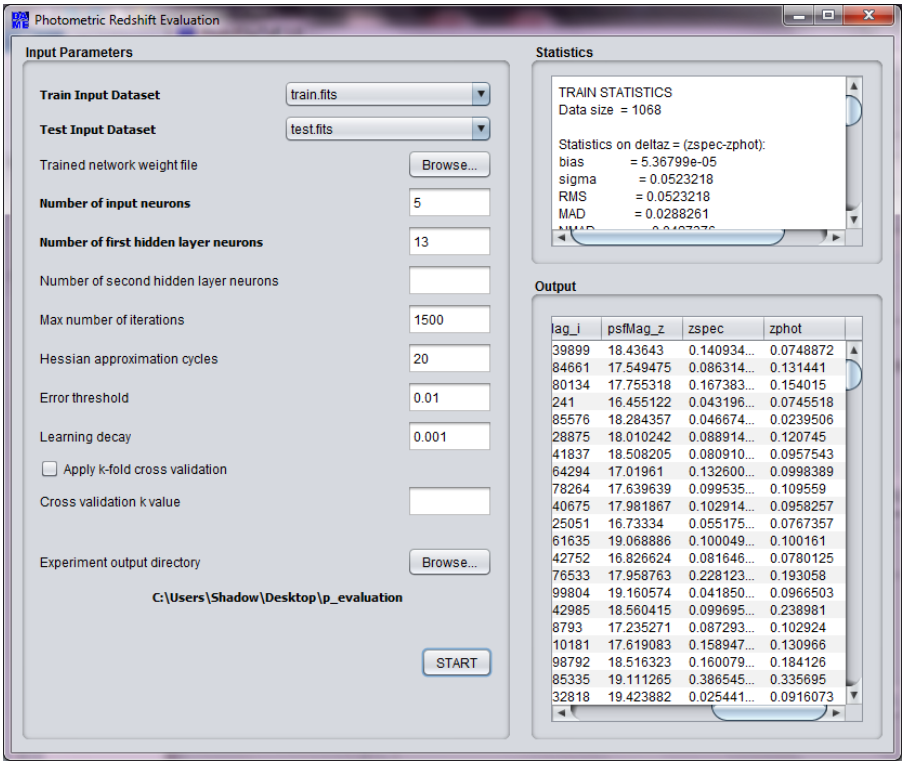


Figure 5.1: Photometric redshift evaluation window. On the left there are the fields for setting parameters. On the right there are two panels. When the experiment is complete, in the upper panel the regression train and test statistics is displayed. In the lower panel the final table with the photo-z column is reported.

validation. It is in fact quite frequent for machine learning models to suffer of an *overfitting* on training data, affecting and badly conditioning the training performances. The problem arises from the paradigm of supervised machine learning itself. Any ML model is trained on a set of training data in order to become able to predict new data points. Therefore its goal is not just to maximize its accuracy on training data, but mainly its predictive accuracy on new data instances. Indeed, the more hard and computationally stiff is the model setup during training, the higher would be the risk to fit the noise and other peculiarities of the training sample in the new data (Dietterich, 1995). The technique known as *cross validation* does not suffer of such drawback; it can avoid overfitting on data and is able to improve the generalization performance of the ML model.

Therefore in the PhotoRApToR application, the validation can be implicitly performed during training, by enabling at the setup step the standard leave-one-out k-fold cross validation mechanism (Geisser, 1975). The au-

tomatized process of the cross-validation is done by performing k different training runs with the following procedure: (i) splitting of the training set into k random subsets, each one composed by a percentage of the data set (depending on the k choice); (ii) at each run we applied the rest of the data set for training and the excluded percentage for validation. The k-fold cross validation is able to avoid overfitting on the training set, although with an increase of the execution time estimable around $k-1$ times the total number of runs.

The photo-z experiment setup sets the MLPQNA input parameters necessary to run a regression train + test experiment, so that it generates an output table which last column is the estimated photometric redshift. Two drop-down menu allow to select the TRAIN dataset and the TEST one (**this parameter is a field required**).
The other parameters are involved in the regression training only, because for the Test phase only the test input dataset is required and remaining parameters are derived by the internal model configuration as frozen at the end of training.

We can group the MLPQNA model training parameters into three subsets:

- **network topology**: all parameters related to the MLP network architecture;
  - *input neurons*: the number of neurons at input layer. In terms of input data set it corresponds to the number of columns of the data table, (also named as input features of the data sample, i.e. magnitudes/colors composing the photometric information of each object in the data), except for the target column (i.e. the spectroscopic redshift), which is related to the single output neuron of the regression network.(**this parameter is a field required**);
  - *first hidden layer neurons*: the number of neurons composing the first layer after the input. As a rule of thumb, it is reasonable to set this number to $2N+1$, where N is the number of input neurons (**this parameter is a field required**);
  - *second hidden layer neurons*: this is an optional parameter. Although not required in normal conditions, as stated by the known universal approximation theorem (Cybenko, 1989), problems having a very high complexity of its parameter space, i.e. with a large amount of distribution irregularities, are better treated by what was defined as *deep* networks, i.e. networks with more than one computational (hidden) layer (Bengio & LeCun, 2007). As a rule

of thumb, it is reasonable to set this number to $N - 1$, where N is the number of input neurons;

- *trained network weights*: this parameter is related to the file containing the matrix of weights (internal connections among neurons). A weight matrix exists only after having performed one training session at least. So far this parameter is left empty at the beginning of any experiment. But, for all other use cases (Test, Run) it is required to load an already trained network. However this parameter could also be used to perform further training cycles for an already trained network (i.e. enhanced training);

- **learning rule setup**: all parameters related to the QNA learning rule;

  - *Max number of iterations*: the maximum number of iterations at each Hessian approximation cycle. Typical range for such value is $[1000, 30000]$, depending on the requested precision. It can affect the computing time of the training;
  - *Hessian approximation cycles*: number of approximation cycles searching for the best value close to the Hessian of the error. If set to zero, the max number of iterations will be used for a single cycle. At each cycle the algorithm performs a series of iterations along the direction of the minimum error gradient, trying to approximate the Hessian value. A reasonable range could be $[20, 60]$, depending on the final precision required. If set to a high value, it is recommended to enable the cross validation option (see below), to prevent overfitting occurrence;
  - *Training error threshold*: This is the stopping criteria of the algorithm. It is the training error threshold (a value of 0.01 is typical for photo-z experiments);
  - *Learning decay*: this value determines the analytical *stiffness* of the approximation process. It affects the expression of weight updating law, by adding the term $decay * ||networkweights||^2$. Its range goes from a minimum value of 0.001 (very low stiffness) up to 100 (very high stiffness). If set to a high value, it is recommended to enable the cross validation option (see below), to prevent overfitting occurrence;

- **validation setup**: all parameters related to the optional training validation process;

– *Cross validation k value*: when the cross validation is enabled, this value is related to the automatic procedure that splits in different subsets the training data set, by applying a k-step cycle in which the training error is evaluated and its performances are validated. A reasonable value could be 5 or 10, depending on the amount of training data used. We remind that this value may highly affect the computing time of the experiment.

Finally **Experiment output directory** is the parent directory of the output for the experiments and it was called **p_evaluation** in the example shown in Fig. 5.1.

5.1 The training error and decay factor

The error calculated during training by the MLPQNA is evaluated for all the presented input patterns between their known target and the calculated output of the model. The error function in the regression case is based on the Least Mean Square (LSE) + Tychonov regularization (Groetsch, 1984). This function is defined as follows:

$$E = \frac{\sum_{i=1}^{N}(y_i - t_i)^2}{2} + \frac{d||W||^2}{2} \tag{5.1}$$

where N is the number of input patterns, y and t are the network output and the pattern target respectively, d is the decay input parameter and W the network weight matrix.

Regularization of the weight decay is the most important issue within the model mechanisms. When the regularization factor is accurately chosen, then generalization error of the trained neural network can be improved, and training can be accelerated. If the best decay regularization parameter d is unknown, it could be experimented the values within the range from 0.001 (weak regularization) up to 100 (very strong regularization). In order to achieve the weight decay rule, we minimize more complex merit function:

$$f = E + \frac{dS}{2} \tag{5.2}$$

Here E is the training set error, S is the sum of squares of network weights, and decay coefficient d controls the amount of smoothing applied to the network. Optimization is performed from the initial point and until the successful stopping of the optimizer has been reached.

Searching for the best decay value is a typical trial-and-error procedure. It is usually performed by training the network with different values of the decay parameter d, from the lower value (no regularization) to the infinite value (strongest regularization). By inspecting statistical results at each stage of the procedure (optimal decay can be selected by using test set or cross-validation, and in the latter case all dataset can be used for training), it can be seen the control tendency to overfit by continuously changing the decay factor. A zero decay corresponds to overfitted network. Infinitely large decay gives us an underfitted network. Between these extreme values there is a range of networks which reproduce dataset with different degrees of precision and smoothness.

5.2 Primer Wizard

When the program is launched, in addition to the main program window, also a tutorial wizard is started, called *Photo-z Primer Wizard*.

1. The first dialog explains scientifical applications of the program and gives the possibility to skip the tutorial by switching to the application main window.

2. In the second dialog it is possible to open table data (selectable choices: ASCII, FitsTable, CSV, VoTable) (cf. Fig. 5.2).

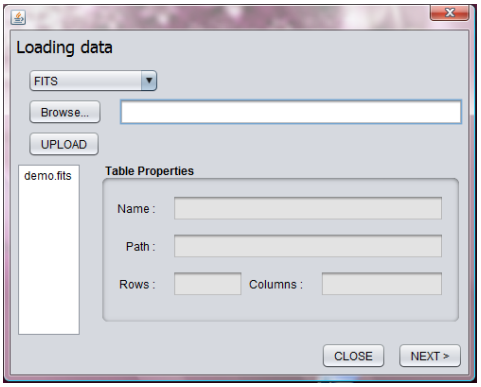


Figure 5.2: Primer Wizard second dialog window.

3. In the third dialog it is possible to manipulate metadata to select only needed columns by a checkbox;

4. In the fourth dialog we can separate our data into two files (*train* and *test*) using the Split function.

5. In the fifth dialog the experiment setup begins (Fig. 5.3). Here it is necessary to insert the parameters to setup the experiment and select the output folder. Then a click on the START button executes the experiment and a status message shows that the experiment is running.

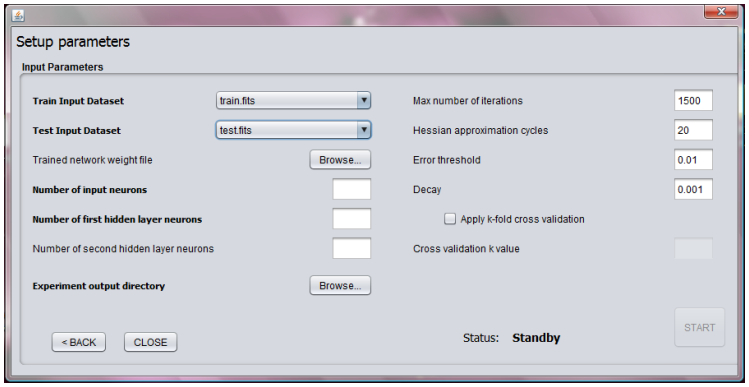


Figure 5.3: Primer Wizard fifth dialog window.

6. At the end of the experiment, the final dialog automatically is opened (Fig. 5.4). Here are displayed the output table on the right and the statistical report on the left. By clicking on the Scatter Plot button, in a different dialog the scatter plot zphot/zspec is shown.

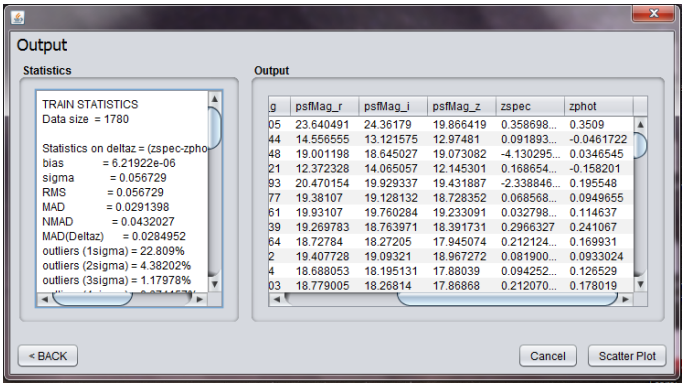


Figure 5.4: Primer Wizard final dialog window.

Chapter 6

Other functionalities

To complete the description of the resources made available by the PhotoRApToR application, we outline that besides photometric redshift estimation, as a specialized kind of regression experiments, the user has the possibility to perform generic regression as well as multi-class classification experiments.

For a generic regression problem, all the above functionalities described in the case of photo-z, remain still valid, with the only straightforward exception for the statistics produced, which is generated for generic quantities formulated below.

$$\begin{aligned} \Delta out &= target - output \\ \Delta out_{norm} &= \frac{target - output}{1 + target} \end{aligned}$$

6.1 Regression

A click on **Regression>Train** menu item opens a new window (Fig. 6.1) where it is necessary to set MLPQNA's input parameters.

- A drop-down menu allows to select the input file; (**this parameter is a field required**)
- if we had already done the training phase, it is possible to use the **trained weight file**;
- the **Number of input neurons** is the number of input dataset columns (except for the target column); (**this parameter is a field required**)

- the **Number of first hidden layer neurons** is the number of neurons of the first hidden layer of the network; (**this parameter is a field required**)

- the **Number of second hidden layer neurons**, as a suggestion this number should be smaller than the previous layer. By default the second hidden layer is empty (not used);

- **Max number of iteration** is one of the internal model parameters. It indicates the number of algorithm iterations and it is one of the stopping criteria. By default this value is set to 1500;

- **Hessian approximation cycles** indicates the number of restarts for each approximation step of the Hessian inverse matrix. By default this value is set to 20;

- **Error threshold** indicates the minimum network error at each iteration step (see Sect. 5.1 for details). Except for problems which are particularly difficult to solve, in which a value of 0.0001 should be used, a value of 0.01 is usually considered sufficient. By default this value is therefore set to 0.01;

- **Decay** indicates the weight regularization decay. If accurately chosen, this parameter leads to an important improvements of the generalization error of the trained neural network and implies an acceleration of training (see Sect. 5.1 for details). By default the value is set to 0.001;

- **Cross validation** is based on an automatic procedure that splits in different subsets the training dataset, by applying a k-step cycle in which the training error is evaluated and its performances are validated. By default the k value is set to 10;

- finally **Experiment output directory** is the parent directory hosting the output for the experiments.

After the parameter setup, a click on *START* button executes the MLPQNA regression experiment and the resulting output is displayed in the main panel on the left. After the experiment, also the statistics is generated with a specific algorithm and the result is presented in the text panel on top of the panel (Fig. 6.1).

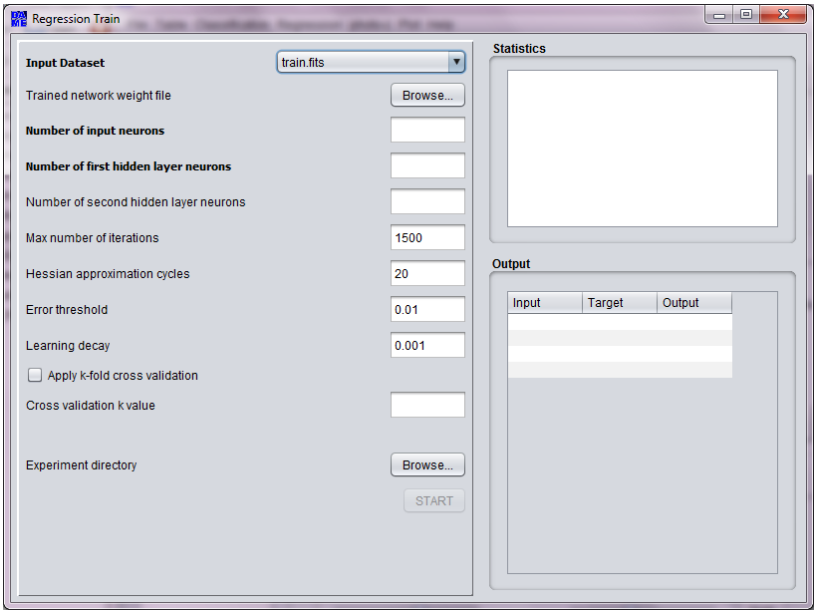


Figure 6.1: Regression Train window. On the left there are the fields for setting parameters. On the right there are two panels. When the experiment is complete, in the upper panel the regression train statistics is displayed. In the lower panel the final table with the photo-z column is reported.

6.2 Classification

In the case of the multi-class classification, the above considerations and options remain still valid with only some differences, described in what follows.

During the training setup, there are two specific parameters, not involved in regression problems:

- *Output neurons*: the number of neurons of the output layer. Forced to 1 in the obvious case of regression experiments, this parameter corresponds here to the number of different classes present in the training sample. It is required that the class identifiers should have a binary coding format for labels (for example a three-class problem is represented by three columns, labeled as, respectively, $1, 0, 0$, $0, 1, 0$ and $0, 0, 1$);

- *Cross entropy*: this optional parameter, if enabled, replaces the standard training error evaluation (for instance the MSE between output and target values). Its meaning is discussed below.

The option **Classification>Train** opens a new window (Fig. 6.2) for the MLPQNA's parameters setting.

- A drop-down menu allows to select the input file; (**this parameter is a field required**);
- Two buttons, one to **Define Classes** and the other to **Skip** this operation and set the other parameters;
- if we had already done the training phase, it is possible to use the **trained weight file**;
- **Number of input neurons:** as specified for regression; (**this parameter is a field required**);
- **Number of first hidden layer neurons:** as specified for regression; (**this parameter is a field required**);
- **Number of second hidden layer neurons:** as specified for regression;
- the **Number of output neurons** is the number of neurons in the output layer of the network. It must correspond to the number of target columns in the input file, represented in binary code; (**this parameter is a field required**);
- **Max number of iteration:** as specified for regression;
- **Hessian approximation cycles:** as specified for regression;
- **Error threshold:** as specified for regression;
- **Decay:** as specified for regression;
- **Cross validation:** as specified for regression;
- finally **Experiment output directory** is the parent directory of the output for the experiments.

The **Define Classes** button open a dialog where the user must set the number of input features and the number of classes that will have as output, codified in a binary representation. By clicking on the **Confirm** button, the tool will check all the occurrences that will be labeled in binary format to become a correct class identifier for the MLPQNA. The user can choose which binary label should be associated to every occurrence or which occurrence should be deleted. By clicking on the **OK** button, it is created the modified table and closed the dialog, coming back to the Classification setup window, showing the field ***Table used*** with the name of the table that will be used

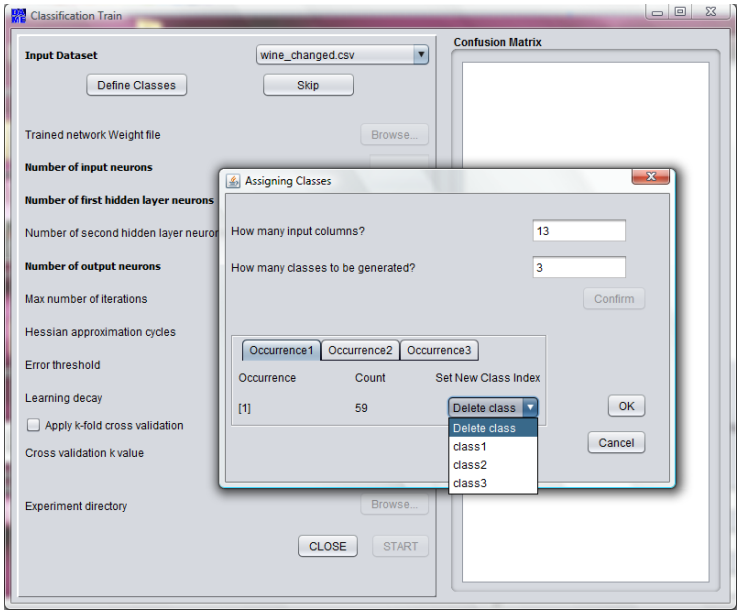


Figure 6.2: Classification Train window. On the left there are the fields for setting parameters. On the right panel the Confusion Matrix is displayed.

during the classification experiment (if the user clicks on the **Skip** button the input table's name will be in this field).

A click on the *START* button executes the MLPQNA classification experiment and the resulting output is displayed in the main panel on the left. The text panel above the *Confusion Matrix* is reported.

By clicking on **Test** or **Run** options of *Regression* and *Classification* menu items, a window similar to those described for the **Train** case is opened.

The Cross Entropy (CE) error function was introduced to address classification problem evaluation in a consistent statistical fashion (Rubinstein et al., 2004). The CE method consists of two phases: *(i)* generate a random data sample (trajectories, vectors, etc.) according to a specified mechanism; *(ii)* update the parameters of the random mechanism based on the data to produce a *better* sample in the next iteration.

In practice a data model is created based on the training set, and its CE is measured on a test set to assess how accurate the model is in predicting the test data. The method compares indeed two probability distributions, p the true distribution of data in any corpus, and q which is the distribution of data as predicted by the model. Since the true distribution is unknown, the CE cannot be directly calculated, while an estimate of CE is obtained using

the following expression:

$$H\left(T,q\right)=-\sum_{i=1}^{N}\frac{1}{N}log_{2}q\left(x_{i}\right)$$

where T is the chosen training set, corresponding to the above mentioned true distribution p, N is the number of objects in the test set, and $q\left(x\right)$ is the probability of the event x estimated from the training set.

Chapter 7

Post-Processing

After having successfully terminated a training session, the model will produce (among several output files) a final network weight matrix (file by default called *trainedWeights.txt*) and the network configuration setup (file by default called *frozen_train_net.txt*), which can be used during next experiment steps (Test and Run use cases), together with their respective input data sets.

7.1 Statistics

As already underlined, concerning the performance evaluation in terms of photometric redshift reconstruction, all statistical results reported throughout this paper are referred to test data sets only. In fact, it is good practice to evaluate the results on data (i.e. the test set) which have never been presented to the network during the training and/or validation phases. The usage of *test plus training* data might introduce an obvious positive systematic bias which could mask reality.

More in general, empirical methods, such as MLPQNA, have the advantage that the training set is made up of real sky objects. Hence they do not suffer from the uncertainty of having accurate templates. In this sense any empirical method intrinsically includes effects such as the filter band-pass and flux calibrations. In fact, as deeply discussed by Collister & Lahav (2004), one of the main drawbacks of these methods is the difficulty in extrapolating to regions of the input parameter space that are not covered and well sampled by the training data. Therefore the efficiency of empirical methods degrades for objects at fainter magnitudes than those included in the training set, as this would require an extrapolation capability on data having properties, such as redshift and photometry, not included in the learned sample. In fact,

another strong requirement of such methods is that the training set must be large enough to cover properly the parameter space in terms of fluxes, colors, magnitudes, object types and redshift. In this case the calibrations and corresponding uncertainties are well known and only limited extrapolations beyond the observed locus in color-magnitude space are required. In conclusion, under the conditions described above about the consistency of the training set, a realistic way to measure photometric uncertainties is to compare the photometric redshifts estimation with spectroscopic measures in the test samples.

The obtained results of the individual experiments have to be evaluated in a consistent and objective manner through an homogeneous set of statistical indicators. Within PhotoRApToR we use a specific algorithm to generate statistics.

For each experiment, given a list of N blind test samples for z_{spec} and z_{phot}, we define:

$$\begin{aligned} \Delta z &= z_{spec} - z_{phot} \\ \Delta z_{norm} &= \frac{z_{spec} - z_{phot}}{1 + z_{spec}} \end{aligned}$$

where Δz_{norm} is the normalized Δz. By indicating with x either Δz or Δz_{norm}, we calculate the following statistical indicators:

$$\begin{aligned} bias(x) &= \frac{\sum_{i=1}^{N} x_i}{N} \\ \sigma(x) &= \sqrt{\frac{\sum_{i=1}^{N} \left[x_i - \left(\frac{\sum_{i=1}^{N} x_i}{N} \right) \right]^2}{N}} \\ RMS(x) &= \sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}} \\ MAD(x) &= Median(\mid x \mid) \\ NMAD(x) &= 1.4826 \times Median(\mid x \mid) \end{aligned}$$

There is also a relation between the Root Mean Square (RMS) and the Standard Deviation σ: $RMS = \sqrt{mean^2 + \sigma^2}$, but σ^2 is the *variance*, so we have $RMS = \sqrt{mean^2 + variance}$. Therefore, for a direct comparison of results, in terms of distance of $m\sigma$ $(m = 1, 2, ...)$ from the distribution of Δz, it is much more precise to use the Standard Deviation as main indicator, rather than the simple RMS.

There is often a confusion about the relation between photometric and spectroscopic redshifts used to apply the statistical indicators. For instance,

the performance could be very different if the simple Δz is used instead of the Δz_{norm}. The idea is that the Δz cannot represent the best choice in the specific case of photometric redshift prediction.

The velocity dispersion error, intrinsically present within the photometric estimation, is not uniform in a wide spectroscopic sample, and the related statistics is not able to give a consistent estimation at all ranges of redshift. On the contrary, the normalized term Δz_{norm} introduces a more uniform information, correlating in a more correct way the variation of photometric estimation, thus permitting a more consistent statistical evaluation at all ranges of spectroscopic redshift.

More in detail:

$$\begin{aligned} z &= \frac{\Delta\lambda}{\lambda} = \frac{\lambda_{obs} - \lambda_{emit}}{\lambda_{emit}} = \\ &= \frac{\lambda_{obs}}{\lambda_{emit}} - 1 \\ => 1 &+ z = \frac{\lambda_{obs}}{\lambda_{emit}} \end{aligned}$$

So, differentiating the Eq. 7.1:

$$\begin{aligned} dz &= d\left(\frac{\lambda_{obs} - \lambda_{emit}}{\lambda_{emit}}\right) = \frac{d\lambda_{obs}}{\lambda_{emit}} = \\ &= \frac{d\lambda_{obs}}{\lambda_{emit}}\frac{\lambda_{obs}}{\lambda_{obs}} = \frac{d\lambda_{obs}}{\lambda_{obs}}(1+z) \end{aligned} \tag{7.1}$$

We then obtain:

$$\frac{dz}{1+z} = \frac{d\lambda_{obs}}{\lambda_{obs}} \tag{7.2}$$

The term on the right of the Eq. 7.2 is exactly the variation between photometric and spectroscopic observed redshift, which is the main focus of the photometric redshift estimation for empirical models which learn its prediction based on the spectroscopic information. This result is invariant to the redshift range considered. In conclusion the term $\frac{dz}{1+z}$ is the best choice on which to apply the statistical operators.

All the described statistical indicators are provided as output of any photo-z estimation test and stored in a dedicated file (by default named as *test_statistics.txt*). For completeness we also provide a similar statistics file as output of any training session. But its use is suggested only as a quick comparison between training and test, just in order to verify the absence of any overfitting occurrence.

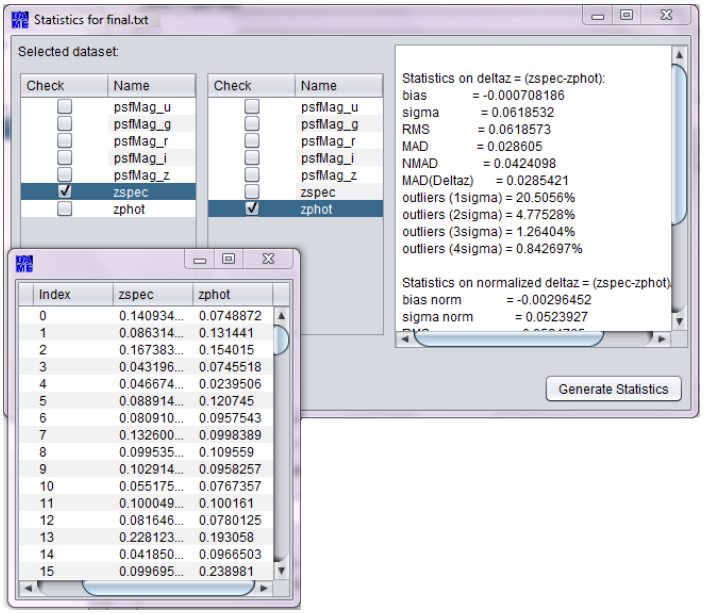


Figure 7.1: Statistic window.

By clicking on **Regression>Statistic** a new window is opened where it is possible to generate statistics using the described statistical indicators on local dataset previously loaded in the *Table List* (Fig. 7.1).
The name of the selected dataset is shown for two lists of its column items, where the user must select one item from the first one and another from the second one. After a click on **Generate Statistics** button, in the panel on the right the statistical indicators, calculated on the two selected features, will be displayed.

7.1.1 Confusion Matrix

Another difference in respect of regression experiments is of course the statistics produced to evaluate the results outcoming from a classification experiment. In this case, at the base of the statistical indicators adopted, there is the commonly known confusion matrix, which can be calculated to easily visualize the classification performance (Provost et al., 1998): each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of a confusion matrix is the simple way to see if the system is mixing different classes.

More specifically, for a generic two-class confusion matrix,

		OUTPUT	
	–	Class A	Class B
TARGET	Class A	N_{AA}	N_{AB}
	Class B	N_{BA}	N_{BB}

we then use its entries to define the following statistical quantities:

- total efficiency: te. Defined as the ratio between the number of correctly classified objects and the total number of objects in the data set. In our confusion matrix example it would be:

$$te = \frac{N_{AA} + N_{BB}}{N_{AA} + N_{AB} + N_{BA} + N_{BB}}$$

- purity of a class: pcN. Defined as the ratio between the number of correctly classified objects of a class and the number of objects classified in that class. In our confusion matrix example it would be:

$$pcA = \frac{N_{AA}}{N_{AA} + N_{BA}}$$

$$pcB = \frac{N_{BB}}{N_{AB} + N_{BB}}$$

- completeness of a class: $cmpN$. Defined as the ratio between the number of correctly classified objects in that class and the total number of objects of that class in the data set. In our confusion matrix example it would be:

$$cmpA = \frac{N_{AA}}{N_{AA} + N_{AB}}$$

$$cmpB = \frac{N_{BB}}{N_{BA} + N_{BB}}$$

- contamination of a class: $cntN$. It is the dual of the purity, namely it is the ratio between misclassified object in a class and the number of objects classified in that class. In our confusion matrix example will be:

$$cntA = 1 - pcA = \frac{N_{BA}}{N_{AA} + N_{BA}}$$

$$cntB = 1 - pcB = \frac{N_{AB}}{N_{AB} + N_{BB}}$$

All these statistical indicators are packed in an output file, produced at the end of the test phase of any classification experiment.

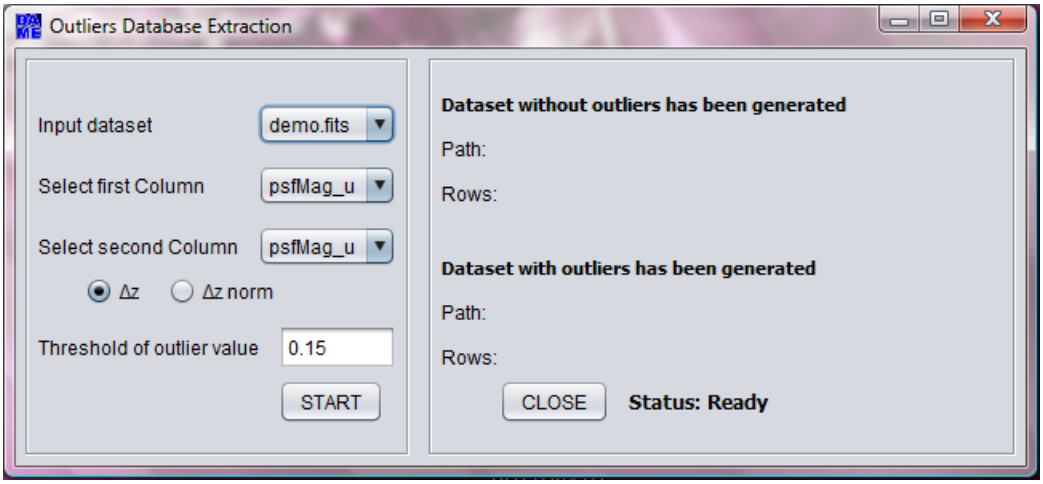


Figure 7.2: Outliers analysis setup window.

7.2 Outliers analysis

For what the analysis of the catastrophic outliers is concerned, according to Mobasher et al. (2007), the parameter $D_{95} \equiv \Delta_{95}/(1+z_{phot})$ enables the identification of outliers in photometric redshifts derived through SED fitting methods (usually evaluated through numerical simulations based on mock catalogues). In fact, in the hypothesis that the redshift error $\Delta z_{norm} = (z_{spec} - z_{phot})/(1+z_{spec})$ is Gaussian, the catastrophic redshift error limit would be constrained by the width of the redshift probability distribution, corresponding to the 95% confidence interval, i.e. with $\Delta_{95} = 2\sigma(\Delta z_{norm})$. In our case, however, photo-z are empirical, i.e. not based on any specific fitting model and it is preferable to use the standard deviation value $\sigma(\Delta z_{norm})$ derived from the photometric cross matched samples, although it could overestimate the theoretical Gaussian σ, due to the residual spectroscopic uncertainty as well as to the method training error. Therefore, we consider as catastrophic outliers the objects with $|\Delta z_{norm}| > 2\sigma(\Delta z_{norm})$.

It is also important to notice that for empirical methods it is useful to analyze the correlation between the $NMAD(\Delta z_{norm}) = 1.48 \times median(|\Delta z_{norm}|)$ and the standard deviation $\sigma_{clean}(\Delta z_{norm})$ calculated on the data sample for which $|\Delta z_{norm}| \leq 2\sigma(\Delta z_{norm})$. In fact, it is normally expected that the quantity $NMAD$ would be less than the value of the σ_{clean}. In such condition we can assert that the pseudo-gaussian distribution of (Δz_{norm}) is mostly influenced by the presence of catastrophic outliers.

Like for the statistical indicators, PhotoRApToR has also a tool to check the presence of outliers in local datasets. For this reason it is useful to rewrite

the relations for Δz and Δz_{norm} in a more general way:

$$\begin{aligned} \Delta z &= Col_1 - Col_2 \\ \Delta z_{norm} &= \frac{Col_1 - Col_2}{1 + Col_1} \end{aligned}$$

By clicking on **Regression>Outliers** a window is opened, where the user can select the dataset to check, and two drop down menus appear (Fig. 7.2), to select the *first* and the *second* columns between which to estimate Δz or Δz_{norm}. The text field below allows to set the threshold over which to consider an object as an outlier: if this tool is used after a regression experiment, the user can assign such threshold equal to the value of σ, as given by the statistics report.
The **START** button enables the outlier analysis. The tool generates a subset without outliers and one with only the outliers and finally, in the right panel, the path of the two files and the number of objects in each subset are reported.

7.3 Plotting tools

Besides statistics, the PhotoRApToR application makes available also some graphical tools, useful to perform a visual inspection of any experiment. In particular a $2D$ scatter plot to show the trend of photo-z vs zspec, as well as histograms to graphically evaluate the distributions of quantities Δz and Δz_{norm}.

Within the PhotoRApToR application there are present also instruments to generate different types of plot. These options are particularly suited during the preparation phase of data for experiments. They in fact enable the possibility to inspect trends, to quantify specific subsets of data, to compare distributions, as well as to inspect particular trends of a generic data table. The graphical options selectable by user are:

- multi-column histograms;
- multiple 2D plots;
- multiple 3D scatter plots.

When one of the plot options (Histo Plot, Scatter Plot or 3D Plot) is clicked, a new window is opened (Figures 7.3, 7.4 and 7.5) where to set the plot parameters: in the upper panel will be displayed the plot, below

there are a text field where it is possible to set the name of the plot and two checkboxes that allow to enable/disable a grid and a legend for the plot. The **Add Plot** button adds other tabs to the previous panel where it is possible to set the parameters of the combined plot with different colours in such a way to compare data from different tables.
By clicking on the **Plot** button, in the upper panel the plot is displayed and stored on local directory in JPEG file format.

7.3.1 Histo Plot

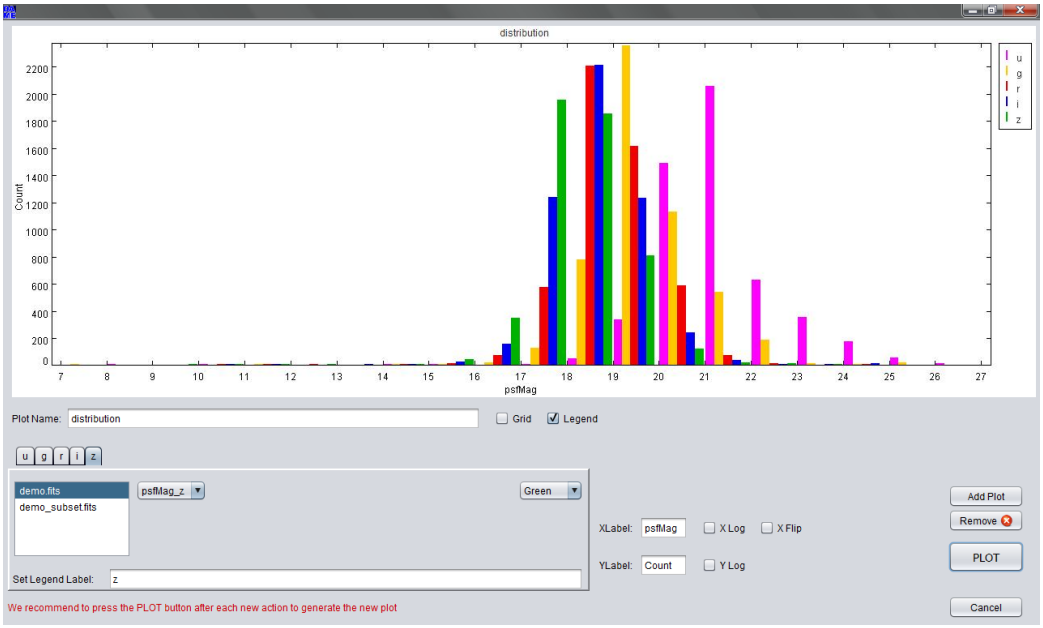


Figure 7.3: Histogram (Histo Plot option) panel.

Each plot option has a different panel where to set parameters. For the **Histo Plot** (Fig.7.3):

- a **Table List** that is the same of the main window;
- a drop-down menu to set the **X-axis** of the diagram;
- two text fields where it is possible to change the labels for the axes X and Y;
- two checkboxes for each axis, one to flip and another to set the axis in logarithmic scale;

- another drop-down menu allows to set the colour;
- there is also another text field where to change the label for the plot legend.

7.3.2 Scatter Plot

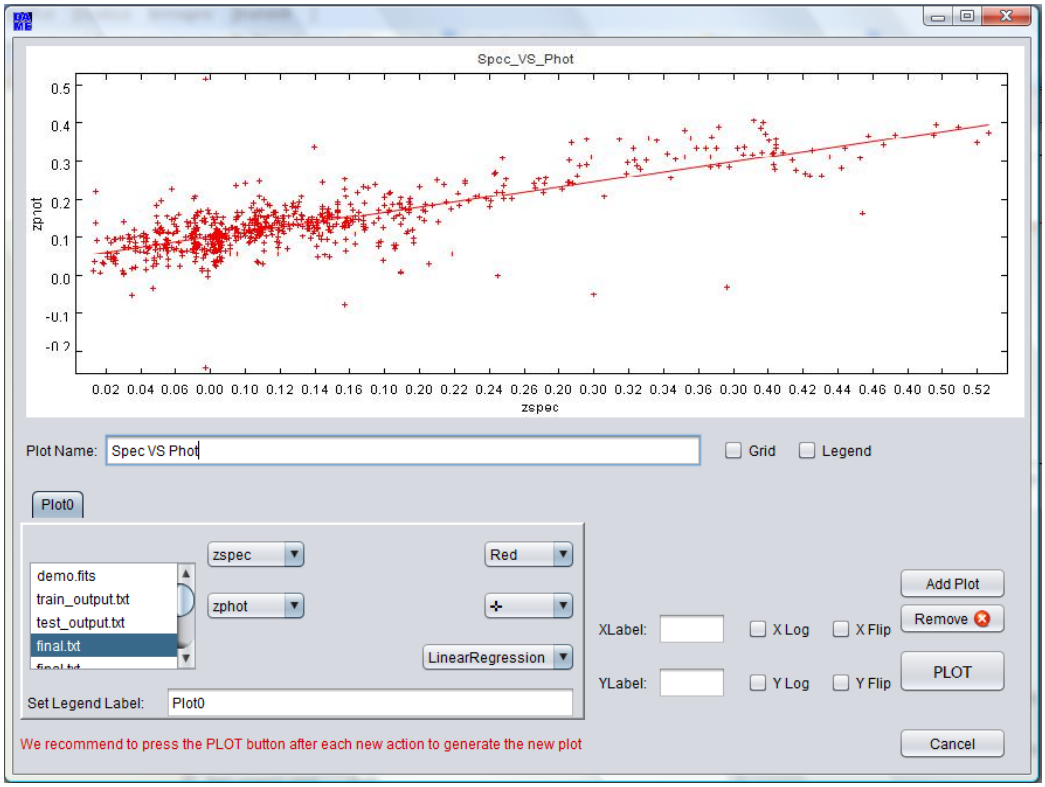


Figure 7.4: Scatter Plot panel.

For the **Scatter Plot** option (Fig.7.4) there are:

- two drop-down menus to set the **X-axis** and **Y-axis** of the diagram;
- two text fields where it is possible to change the labels for the axes X and Y;
- two checkboxes for each axis, one to flip and another to set the axis in logarithmic scale;
- three drop-down menus allowing to set the *Line Style*, the *Colour* and the *Marker*;

- there is also another text field where to change the label for the plot legend.

7.3.3 3D Plot

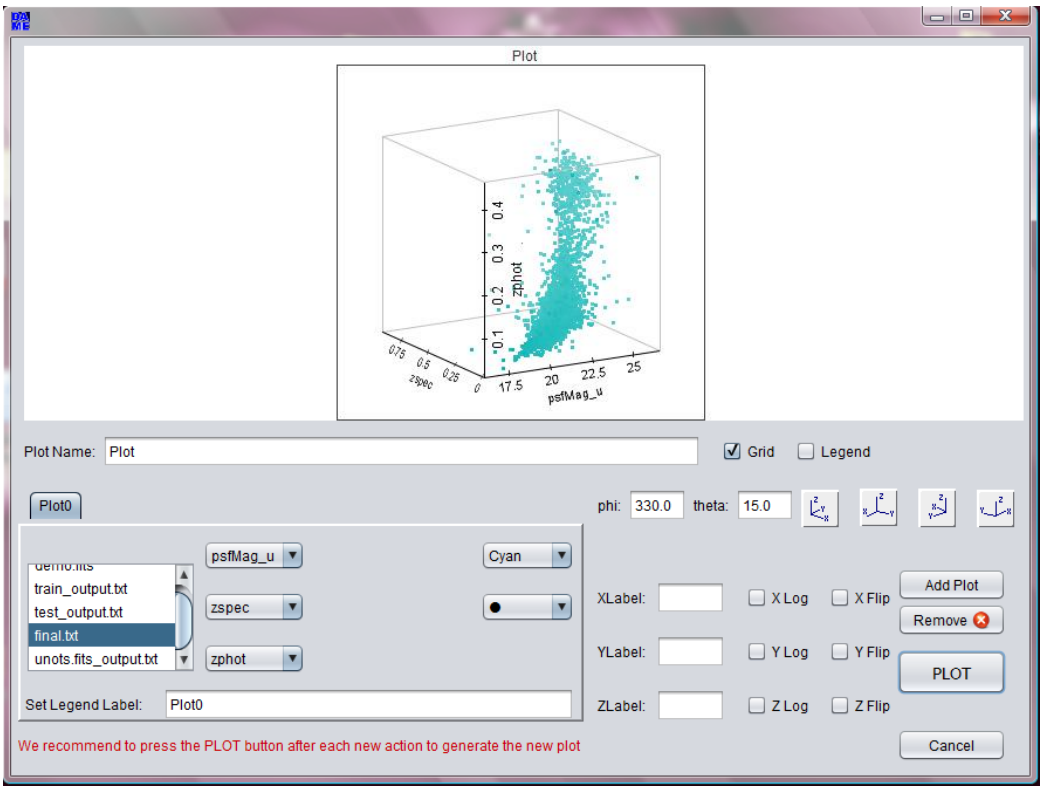


Figure 7.5: 3D Plot panel.

Finally, for **3D Plot** (Fig.7.5):

- three drop-down menus to set the **X-axis**, **Y-axis** and **Z-axis** of the diagram;
- three text fields where it is possible to change the labels for the axes X, Y and Z;
- two checkboxes for each axis, one to flip and another to set the axis in logarithmic scale;
- two drop-down menus allowing to set the *Colour* and the *Marker*;

- there is also another text field where to change the label for the plot legend.

Chapter 8

Troubleshooting

This chapter provides answers to specific troubles arising from the use of the application. In order to improve your user experience with PhotoRApToR, it is recommended to read this section to learn more about common pitfalls and to get recommendations on how to correctly use the application and to recover wrong situations as well.

For any request the user can send an e-mail to *helpdame@gmail.com* and will be re-contacted as soon as possible.

8.1 Heap memory limit

During the Pre-processing phase, the user must check that the dimension of the loaded dataset file is smaller than the JVM assigned memory.

The Java heap is where the objects of a Java program lives. It is a repository for alive and dead objects, and free memory as well. When an object can no longer be reached from any pointer in the running program, it is considered *garbage*. If the dataset is not allocated in the Java heap, this error does not necessarily imply a memory leak. The problem can be as simple as a configuration issue, where the default heap size is insufficient for the application.

In these cases, users must specify a new Java heap size values. This can be done executing PhotoRaptor from a terminal with a command line as follow:

java -jar -Xms*m -Xmx***m PhotoRApToR.jar**

that allows to allocate *** megabytes of memory to the minimum (-Xms) and maximum (-Xmx) heap sizes. The default size for these values is measured in bytes. Append the letter 'k' or 'K' to the value to indicate kilobytes, 'm' or 'M' to indicate megabytes, and 'g' or 'G' to indicate gigabytes.

8.2 Corrupted datasets

PhotoRApToR allows to open and to edit datasets in different file formats, but when users try to load a file, the application checks if data are not corrupted, or loaded as a wrong type specification.
A warning dialog is shown if the file is wrong or corrupted and it is not added to the Table List.

One of most frequent cases of a corrupted dataset happens when users edit data tables ignoring table metadata.

8.3 Filenames with spaces

Datasets whose name contains spaces or that are located in folders whose name contains spaces, generate errors if used during the Experiment phase. If users try to use this type of file as input for an experiment or if the name chosen for the output folder contains spaces, PhotoRApToR will show a warning dialog and the operation is stopped.

It is necessary to use data folders whose names are specified without spaces and to control dataset names before to run PhotoRApToR.

8.4 Mac OS X Window Focus problem

There could be occasional incompatibilities between the Java Swing tool and Mac OS X systems. Within the application this may primarily occur with a particular panel: after having selected the photo-z Menu button, the new setup window may result not editable. In such situation it is suggested to change focus, by clicking on any other open window in your desktop and then to select again the photo-z setup panel. Hereinafter the user should be able to proceed with the normal setup.

8.5 Generic problems

- ***After installation the program doesn't start***: the most common reason could be the absence or wrong setup of the JVM on the user machine. Please check it and in case download and install the latest version of the JVM compatible with the OS running on the user machine. The official website is `http://www.oracle.com/technetwork/java/index.html`. Another source of failure could be the wrong downloaded version of the package, not matching the user OS running on the local machine. Please verify carefully this requirement. In case of still wrong execution of the program, please contact us by specifying the wrong condition/message.

Appendix A

The Machine Learning model

As introduced, the core engine of the PhotoRApToR application is the ML model underlying all data mining experiments, for instance the MLPQNA method. It is a Multi Layer Perceptron (MLP; Rosenblatt 1961) neural network trained by a learning rule based on the Quasi Newton Algorithm (QNA). It is one of the widely used feed-forward neural networks in a large variety of scientific and social contexts.

The Quasi Newton Algorithm (QNA) is a variable metric method for finding local maxima and minima of functions (Davidon, 1991). The model based on this learning rule and on the MLP network topology is then called MLPQNA. QNA is based on Newton's method to find the stationary (i.e. the zero gradient) point of a function. The QNA is an optimization of Newton based learning rule, because the implementation is based on an incremental approximation of the Hessian by a cyclic gradient calculation. In PhotoRApToR the Quasi Newton method has been implemented by following the known L-BFGS algorithm (Limited memory - Broyden Fletcher Goldfarb Shanno; Byrd et al. 1994). As a matter of fact, this method was designed to optimize the functions with a variable number of arguments (hundreds to thousands), because in this case it is worth to have an increased number of iterations, due to the lower approximation precision. This is particularly useful in astrophysical data mining problems, where usually the parameter space is dimensionally huge and is often affected by a low signal-to-noise ratio.

Most of the analytical characteristics of the method have been deeply described in the contexts of both classification (Brescia et al., 2012b) and regression (Brescia et al., 2013a; Cavuoti et al., 2012). We suggest to refer to these articles in case of interest about mathematical theory behind the method as well as on examples of its applications in real astrophysical contexts.

Acknowledgments

The authors wish to thank the whole DAME working group, whose huge efforts made the DM facility available to the scientific community. MB wishes to thank the fiNaNcial support of PRIN-INAF 2010, *Architecture and Tomography of Galaxy Clusters*. The authors also wish to thank the fiNaNcial support of Project F.A.R.O. III Tornata (University Federico II of Naples). GL acknowledges fiNaNcial contribution through the PRIN-MIUR 2012 Euclid.

Bibliography

Albrecht, A., Bernstein, G., Cahn, R. et al., 2006. Report of the Dark Energy Task Force.

ANSI (American National Standards Institute) et al. 1977, American National Standard Code for Information Interchange. The Institute.

Baum, W.A., 1962. Photoelectric Magnitudes and Red-Shifts. In Proceedings from IAU Symposium, Problems of Extra-Galactic Research, Edited by George Cunliffe McVittie. International Astronomical Union Symposium no. 15, Macmillan Press, New York, p.390.

Bengio, Y., & LeCun, J., 2007. In Large-Scale Kernel Machines. MIT Press.

Biviano, A., et al., 2013. CLASH-VLT: The mass, velocity-anisotropy, and pseudo-phase-space density profiles of the z=0.44 galaxy cluster MACS 1206.2-0847. A&A, 558, A1, 22 pages

Brescia M., 2012a. New Trends in E-Science: Machine Learning and Knowledge Discovery in Databases. Horizons in Computer Science Research, Thomas S. Clary (eds.), Series Horizons in Computer Science Vol. 7, Nova Science Publishers, ISBN: 978-1-61942-774-7.

Brescia, M., Cavuoti, S., Paolillo, M., Longo, G., Puzia, T., 2012b, MNRAS, 421, 2, 1155.

Brescia, M., Cavuoti, S., D'Abrusco, R., Longo, G., & Mercurio, A., 2013a. Photometric redshifts for Quasars in multi band Surveys. ApJ, 772, 140.

Brescia, M., Cavuoti, S., De Stefano, V., & Longo, G., 2013b. A catalogue of photometric redshifts for the SDSS-DR9 galaxies. Submitted to A&A.

Byrd, R.H, Nocedal, J., and Schnabel, R.B., 1994. Mathematical Programming, 63, 129.

Capozzi, D., De Filippis, E., Paolillo, M., D'Abrusco, R., Longo, G., 2009. The properties of the heterogeneous Shakhbazyan groups of galaxies in the SDSS. Monthly Notices of the Royal Astronomical Society, Volume 396, Issue 2, pp. 900-917

Cavuoti, S.; Brescia, M.; Longo, G.; Mercurio, A.; 2012, Photometric redshifts with Quasi Newton Algorithm (MLPQNA). Results in the PHAT1 contest, A&A, Vol. 546, A13, pp. 1-8

Collister, A. A. & Lahav, O., 2004. ANNz: Estimating Photometric Redshifts Using Artificial Neural Networks. PASP, 116, 345.

Connolly, A.J., Csabai, I., Szalay, A.S., Koo, D.C., Kron, R.G., Munn, J.A., 1995. Slicing Through Multicolor Space: Galaxy Redshifts from Broadband Photometry. Astronomical Journal v.110, p.2655.

Cybenko, G., 1989. Approximations by superpositions of sigmoidal functions. Mathematics of Control, Signals, and Systems, 2, 303.

The Dark Energy Survey Collaboration, 2005, The Dark Energy Survey, White Paper submitted to the Dark Energy Task Force, 42 pages, arXiv:0510346.

Davidon, W.C., 1991. SIAM Journal on Optimization.

Dietterich, T., 1995. Overfitting and Undercomputing in Machine Learning. Computing Surveys, 27, 326.

Euclid Red Book, ESA Technical Document, 2011, ESA/SRE(2011)12, Issue 1.1, [arXiv:astro-ph/1110.3193].

Geisser, S., 1975. The predictive sample reuse method with applications. Journal of the American Statistical Association, 70 (350), 320-328.

Groetsch 1984, The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind, Pitman, Boston.

Hoaglin, D.C., Mosteller, F., & Tukey, J.W., 1983. Understanding Robust and Exploratory Data Analysis, New York: Wiley.

Ilbert, O., Capak, P., Salvato, M., et al., 2009. Cosmos Photometric Redshifts with 30-bands for $2 - deg^2$. The Astrophysical Journal 690, 1236.

Ivezic, Z., et al. (the LSST team), 2009. The LSST Science Book, v2.0.

Kearns, M., 1996. A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for Training-Test Split, Neural Information Processing 8, D.S. Touretzky, M,C. Mozer and M.E. Hasselmo (eds.), Morgan Kaufmann, pp. 183-189.

Koo, D.C., 1985. Optical multicolors - A poor person's Z machine for galaxies. Astronomical Journal, vol. 90, March 1985, p. 418-440

Loh, E.D. & Spillar, E.J., 1986. Photometric redshifts of galaxies. Astrophysical Journal, Part 1, vol. 303, April 1, p. 154-161.

Marlin, B.M., 2008. Missing data problems in machine learning, Library and Archives:Canada.

Mobasher, B., Capak, P., Scoville, N. Z., Dahlen, T., Salvato, M., Aussel, H., Thompson, D. J., Feldmann, R., Tasca, L., Le Fevre, O., Lilly, S., Carollo, C. M., Kartaltepe, J. S., McCracken, H., Mould, J., Renzini, A., Sanders, D. B., Shopbell, P. L., Taniguchi, Y., Ajiki, M., Shioya, Y., Contini, T., Giavalisco, M., Ilbert, O., Iovino, A., Le Brun, V., Mainieri, V., Mignoli, M., Scodeggio, M., 2007. Photometric Redshifts of Galaxies in COSMOS. The Astrophysical Journal Supplement Series, Volume 172, Issue 1, pp. 117-131

Peacock, J.A., Schneider, P., Efstathiou, G. et al., 2006. ESA-ESO Working Group on Fundamental Cosmology, ESA-ESO Working Group on Fundamental Cosmology, Tech. Rep.

Pello, R., Miralles, J.M., Le Borgne, J.F., Picat, J.P., Soucail, G., Bruzual, G., 1996. Identification of a high redshift cluster in the field of Q2345+007 through deep BRIJK' photometry. A&A, v.314, p.73,86.

Pennebaker, W.B., and Mitchell, J.L., 1993. JPEG still image data compression standard, (3rd ed.).

Provost, F., Fawcett, T., Kohavi, R., 1998. The Case Against Accuracy Estimation for Comparing Induction Algorithms. Proceedings of the 15th International Conference on Machine Learning. Morgan Kaufmann. pp. 445-553.

Puschell, J.J, Owen, F.N., & Laing, R.A., 1982. Near-infrared photometry of distant radio galaxies - Spectral flux distributions and redshift estimates. Astrophysical Journal, Part 2 - Letters to the Editor, vol. 257, June 15, 1982, p. L57-L61.

Repici, J., (2010). How To: The Comma Separated Value (CSV) File Format. 2010, Creativyst Inc.

Rosenblatt, F., 1961. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC.

Rubinstein, R.Y., Kroese D.P., 2004. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning, Springer-Verlag, New York.

S.V. Stehman, 1997. Selecting and interpreting measures of thematic classification accuracy. Remote Sensing of Environment 62 (1): 77,89.

Umetsu, K.; Medezinski, E.; Nonino, M.; et al. 2012. CLASH: Mass Distribution in and around MACS J1206.2-0847 from a Full Cluster Lensing Analysis. ApJ, 755, 1, 56.

Wells, D.C., Greisen, E.W., Harten, R.H., 1981. FITS: a Flexible Image transport System. Astronomy & Astrophysics Supplement Series, Vol. 44, p. 363.