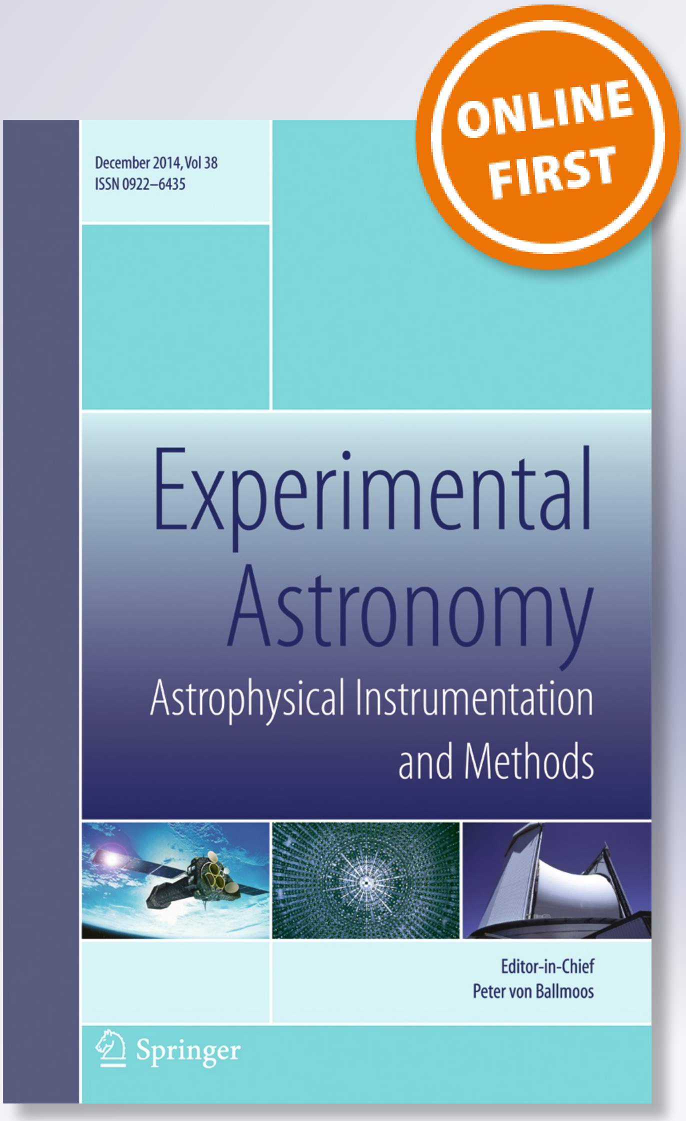
Photometric redshift estimation based on data mining with PhotoRApToR

S. Cavuoti, M. Brescia, V. De Stefano & G. Longo

Experimental Astronomy
Astrophysical Instrumentation and Methods

ISSN 0922-6435

Exp Astron
DOI 10.1007/s10686-015-9443-4



December 2014, Vol 38
ISSN 0922–6435

Experimental Astronomy

Astrophysical Instrumentation and Methods

Editor-in-Chief
Peter von Ballmoos

Springer

ONLINE FIRST

Springer

Exp Astron
DOI 10.1007/s10686-015-9443-4

ORIGINAL ARTICLE

Photometric redshift estimation based on data mining with PhotoRApToR

S. Cavuoti · M. Brescia · V. De Stefano · G. Longo

Received: 30 September 2014 / Accepted: 26 January 2015

Abstract Photometric redshifts (photo-z) are crucial to the scientific exploitation of modern panchromatic digital surveys. In this paper we present PhotoRApToR (Photometric Research Application To Redshift): a Java/C++ based desktop application capable to solve non-linear regression and multi-variate classification problems, in particular specialized for photo-z estimation. It embeds a machine learning algorithm, namely a multi-layer neural network trained by the Quasi Newton learning rule, and special tools dedicated to pre- and post-processing data. PhotoRApToR has been successfully tested on several scientific cases. The application is available for free download from the DAME Program web site.

Keywords Techniques: photometric · Galaxies: distances and redshifts · Galaxies: photometry · Cosmology: observations · Methods: data analysis

1 Introduction

The ever growing amount of astronomical data provided by the new large scale digital surveys in a wide range of wavelengths of the electromagnetic spectrum has been challenging the way astronomers carry out their everyday analysis of astronomical sources and we can safely assert that the human ability to directly visualize and correlate astronomical data has been pushed to its limits in the past few years. As a

S. Cavuoti · M. Brescia (✉)
INAF - Astronomical Observatory of Capodimonte, Napoli, Italy
e-mail: brescia@na.astro.it

S. Cavuoti
e-mail: cavuoti@na.astro.it

V. De Stefano · G. Longo
Department of Physics, University Federico II of Naples, Naples, Italy

Published online: 18 February 2015

consequence of the fact that data have become too complex to be effectively managed and analysed with traditional tools, a new methodological shift is emerging and Data Mining (DM) techniques are becoming more and more popular in tackling knowledge discovery problems. A typical problem which is addressed with these new techniques is that of the evaluation of photometric redshifts. The request for accurate photometric redshifts (photo-z) has increased over the years due both to the advent of a new generation of multi-band surveys [17] and to the availability of large public datasets which allowed to pursue a wide variety of scientific cases. Ongoing and future large-field public imaging projects, such as Pan-STARRS [24], KiDS[1] (Kilo-Degree Survey), DES [19], the planned surveys with LSST (Large Synoptic Survey Telescope, [31]) and Euclid [23], rely on accurate photo-z to achieve their scientific goals.

Photo-z are in fact essential in constraining dark matter and dark energy through weak gravitational lensing [44], for the identification of galaxy clusters and groups (e.g. [13]), for type Ia Supernovae, and for the study of the mass function of galaxy clusters [1, 37, 48]), just to quote a few applications. Photometric filters integrate fluxes over a quite large interval of wavelengths and, therefore, the accuracy of photometric redshift reconstruction is worse than that of spectroscopic redshifts. On the other hand, in the absence of the minimal telescope time necessary to determine spectroscopically the redshifts for all sources in a sample, photometric redshifts methods provide a much more convenient way to estimate the distance of such sources. The physical mechanism responsible for the correlation existing between the photometric features and the redshift of an astronomical source, is the change in the observed fluxes caused by the fact that, due to the stretch introduced by the redshift, prominent features of the spectrum move across the different filters of a photometric system.

This mechanism implies a non-linear mapping between the photometric parameter space of the galaxies and the redshift values. This non linear mapping function can be inferred using advanced statistical and data mining methods in order to evaluate photometric estimates of the redshift for a large number of sources. All existing implementations can be broadly categorized into two classes of methods: theoretical and empirical. *Theoretical methods* use template based Spectral Energy Distributions (SEDs), obtained either from observed galaxy spectra or from synthetic models. These methods require an extensive a-priori knowledge about the physical properties of the objects, hence they may be biased by such information. They, however, represent the only viable method when dealing with faint objects outside the spectroscopic limit ([27] and references therein).

When accurate and multi-band photometry for a large number of objects is complemented by spectroscopic redshifts for a statistically significant sub-sample of the same objects, *empirical methods* might offer greater accuracy. This sample needs, however, to be statistically representative of the parent population. The spectroscopic redshifts of this sub-sample are then used to constrain the fit of an interpolating function mapping the photometric parameter space. Different methods differ mainly in the way such interpolation is performed.

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

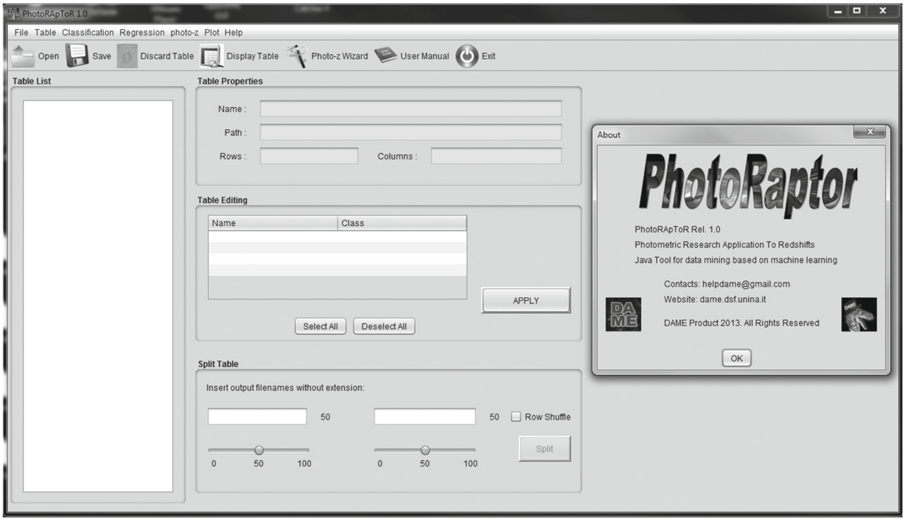
From the data mining point of view, the evaluation of photo-z is a supervised learning problem [27, 46], where a set of *examples* is used by the method to learn how to reconstruct the relation between the *parameters* and the *target* [6]. In the specific case of photometric redshifts, the parameters are fluxes, magnitudes or colors of the sources while the targets are the spectroscopic redshifts.

A con of this approach being that, as it happens for all interpolative problems, such methods may suffer to extrapolate and therefore they are effective only when applied to galaxies with photometry that lie within the range of fluxes/magnitudes and redshifts well sampled by the training set. In this paper we present PhotoRApToR (Photometric Research Application To Redshift), namely a Java based desktop application capable to solve regression and classification problems which has been finely tuned for photo-z estimation. It embeds a Machine Learning (ML) algorithm, in the specific case a particular instance of a multi-layer neural network, and special tools dedicated to pre- and post-processing data. The machine learning model is the MLPQNA (Multi Layer Perceptron trained by the Quasi Newton Algorithm), which has proven to be particularly powerful photo-z estimator, also in presence of relatively small spectroscopic Knowledge Base (KB) [8, 14]. The application is available for download from the DAME program web site.[2] This paper is organized as follows: in Section 2 we describe the Java application; in Section 3 we discuss in some details how the evaluation of photometric redshifts is performed. Section 4 describe other functionalities provided by the application, while Section 5 is dedicated to a comparison between PhotoRApToR and an alternative public machine learning tool. Finally in Section 6 we outline some lessons which were learned during the implementation of PhotoRaPToR and draw some future developments.

2 PhotoRApToR

Everyone who has used neural methods to produce photometric redshift evaluation knows that, in order to optimize the results in terms of features, neural network architecture, evaluation of the internal and external errors, many experiments are needed. When coupled with the needs of modern surveys, which require huge data sets to be processed, it clearly emerges the need for a user friendly, fast and scalable application. This application needs to run client-side, since a great part of astronomical data is stored in private archives that are not fully accessible on line, thus preventing the use of remote applications, such as those provided by the DAMEWARE tool [9]. The code of the application was developed in Java language and runs on top of a standard Java Virtual Machine, while the machine learning model was implemented in C++ language to increase the core execution speed. Therefore different installation packages are provided to support the most common platforms. Moreover, the application includes a wizard, which can easily introduce the user through the various functionalities offered by the tool. The Fig. 1 shows the main window of the program. The main features of PhotoRApToR can be summarized as it follows:

[2] http://dame.dsf.unina.it/dame_photoz.html#photoraptor



PhotoRApToR Rel. 1.0
Photometric Research Application To Redshifts
Java Tool for data mining based on machine learning

Contacts: helpdame@gmail.com
Website: dame.dsf.unina.it
DAME Product 2013. All Rights Reserved

Fig. 1 The PhotoRApToR main window

- *Data table manipulation*. It allows the user to navigate throughout his/her data sets and related *metadata*, as well as to prepare data tables to be submitted for experiments. It includes several options to perform the editing, ordering, splitting and shuffling of 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 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 output 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. After the training phase, the system will be able to predict the right photo-z value

for any new sky object belonging to the same type (in terms of photometric input features) of the Knowledge Base;

- *Data visualization*. The application includes some $2D$ and $3D$ graphics tools, for instance multiple histograms and multiple $2D/3D$ scatter plots. Such tools are often required to visually inspect and explore data distributions and trends;
- *Data statistics*. For both classification and regression experiments a statistical report is provided about their output. In the first case, the typical confusion matrix [45] is given, including related statistical indicators such as classification efficiency, completeness, purity and contamination for each of the classes defined by the specific problem. 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 (bias), standard deviation (σ), Root Mean Square (RMS), Median Absolute Deviation (MAD) and the *Normalized* MAD (NMAD, [28]), the latter specific for the photo-z quality estimation, together with percentages of *outliers* at the common threshold 0.15 and at different multiples of σ [10, 30].

In Fig. 2 the layout of a general PhotoRApToR experiment workflow is shown. It is valid for either regression and classification cases.

2.1 The machine learning model

The core of the PhotoRApToR application is its ML model, for instance the MLPQNA method. It is a Multi Layer Perceptron (MLP; [42]) neural network (Fig. 3), which is among the most used feed-forward neural networks in a large variety of scientific and social contexts. The MLP is trained by a learning rule based on the Quasi Newton Algorithm (QNA).

The QNA is a variable metric method for finding local maxima and minima of functions [20]. 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. In particular, the QNA is an optimization of Newton's learning rule, because the implementation is based on a statistical approximation of the Hessian of the error function, obtained through a cyclic gradient calculation.

In PhotoRApToR the Quasi Newton method was implemented by following the known L-BFGS algorithm (Limited memory - Broyden Fletcher Goldfarb Shanno; [12]), which was originally designed for problems with a very large number of features (hundreds to thousands), because in this case it is worth having an increased iteration number due to the lower approximation precision because the overheads become much lower. This is particularly useful in astrophysical data mining problems, where usually the parameter space is dimensionally huge and is often afflicted by a low signal-to-noise ratio.

The analytical description of the method has been described in the contexts of both classification [7] and regression [8] and [14]. In the present work, we focus the attention on its parameter setup and correct use within the presented framework.

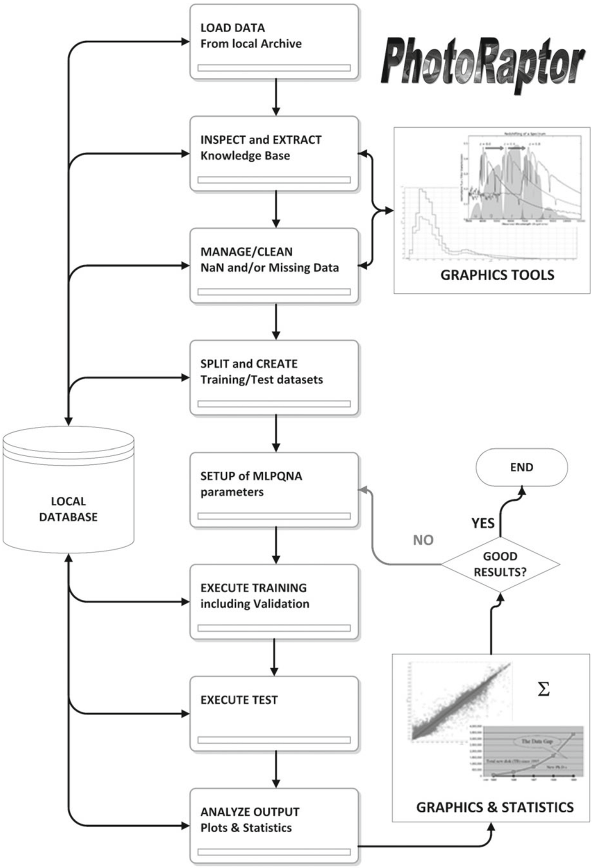


Fig. 2 The workflow of a generic experiment performed with PhotoRApToR

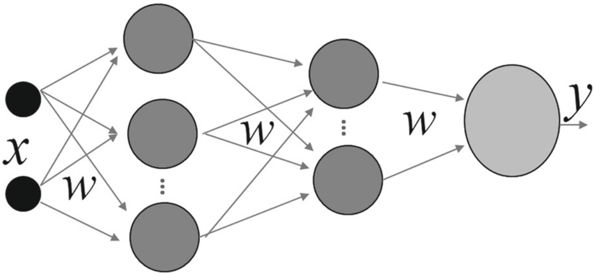


Fig. 3 The typical topology of a generic feed-forward neural network, in this case representing the architecture of MLPQNA. In the simple example there are two hidden layers (the two blocks of dark gray circles) between the input (X) and output (Y) layers, corresponding to the architecture mostly used in the case of photo-z estimation. Arrows between layers indicate the connections (weights *w*) among neurons. These weights are changed during the training iteration loop, according to the learning rule QNA

3 Photometric redshift estimation

In practice, the problem of photo-z evaluation consists in finding the unknown function which maps the photometric set of features (magnitudes and/or colors) into the spectroscopic redshift space. If a consistent fraction of the objects with spectroscopic redshifts is available, the problem can in fact be approached as a data mining regression problem, where the a priori knowledge (i.e. the spectroscopic redshifts forming the KB), is used to uncover the mapping function. This function can then be used to derive photo-z for objects without the spectroscopic counterpart information. Without entering into much details, which can be found in the literature quoted below and in the references therein, we just outline that our method has been successfully used in many experiments done on different KBs, often composed through accurate cross-matching among public surveys, such as SDSS for galaxies [10], UKIDSS, SDSS, GALEX and WISE for quasars (many of the following figures are referring to this experiment; [8], GOODS-North for the PHAT1 contest [14] and CLASH-VLT data for galaxies [4]. Other photo-z prediction experiments are in progress as preparatory work for the Euclid Mission [33] and the KiDS[3] survey projects.

3.1 User data handling

The fundamental premise to use PhotoRaPToR is that the user must preliminarily know how to represent the data and, as trivial as it might seem, it is worth to explicitly state that 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 used data. In what follows we shall call features the input parameters (i.e., for instance, fluxes, magnitudes or colors in the case of photo-z estimation).

Data Formats 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 the pattern features, and to force them to assume a uniform representation before submitting them to the training process. In this respect real working cases might be quite different. PhotoRApToR can ingest and/or produce data in any of the following supported formats:

- FITS [49]: tabular/image;
- ASCII [2]: ordinary text, i.e. space separated values;
- VOTable[4]: VO (Virtual Observatory) compliant XML-based documents;
- CSV [41]: Comma Separated Values;
- JPEG [39]: Joint Photographic Expert Group, as image output type.

Missing Data Very frequently, data tables have empty entries (sparse matrix) or missing (lack of observed values for some features in some patterns). Missing values

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

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

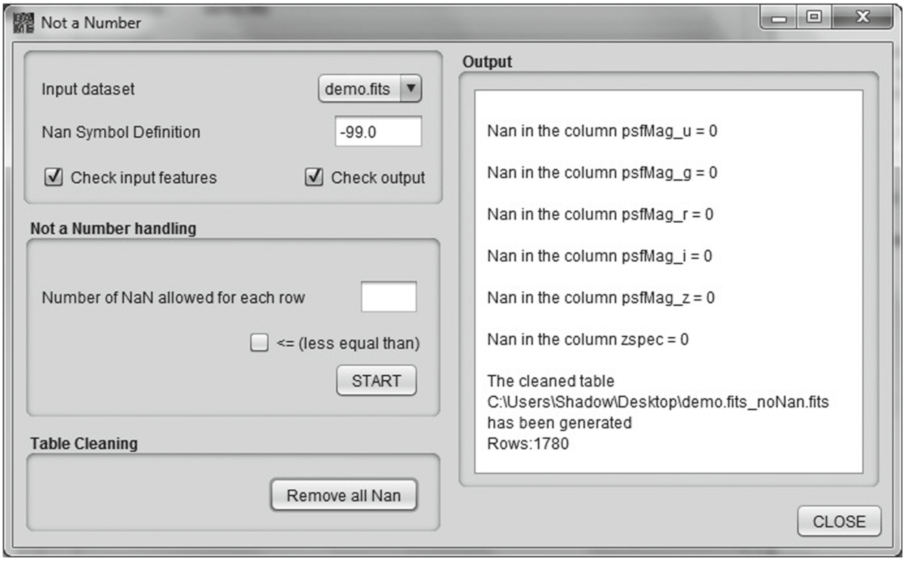
[34] are frequently (but not always) identified by special entries in the patterns, like Not-A-Number, out-of-range, negative values in a numeric field normally accepting only positive entries etc. Missing data is among the most frequent source of perturbation in the learning process, causing confusion in classification experiments or mismatching 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 a single 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 data mining 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 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 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 prevent wrong behaviors in the learning process.

Data entries affected by missing attributes, i.e. patterns having fake values for some features, may be used within the knowledge base used 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 has to 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 catalogues. The organization of data sets with different rates of missing data can be performed through PhotoRApToR by means of a series of options.

The Fig. 4 shows the panel dedicated to define and quantify the presence of missing or bad data within the user tables. The panel allows: (*i*) to quantify the number of wrong values to be retained/removed in/from the data patterns; (*ii*) to completely remove the data patterns affected by the presence of NaN occurrences; (*iii*) to assign arbitrary symbols to wrong or missing entries in the dataset (i.e. symbols like “-999”, “NaN” or whatever).



Input dataset: demo.fits

Nan Symbol Definition: -99.0

☑ Check input features ☑ Check output

Number of NaN allowed for each row:

□ <= (less equal than)

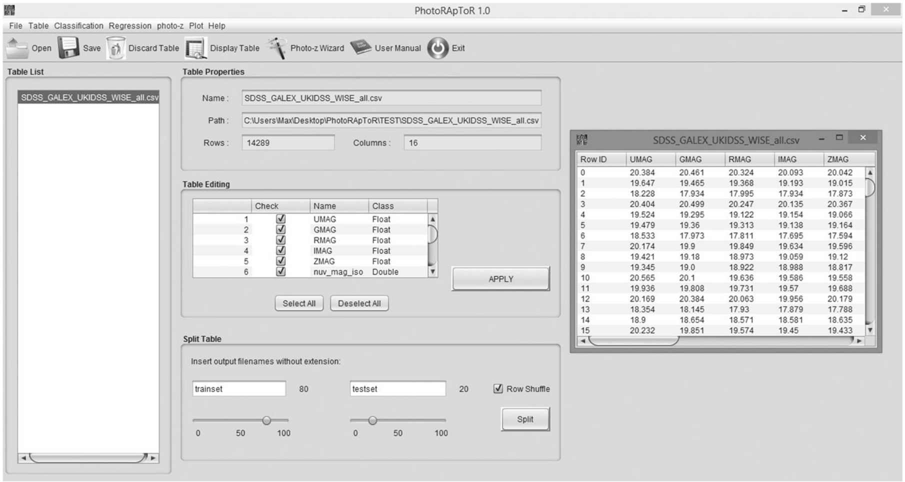
```
Nan in the column psfMag_u = 0

Nan in the column psfMag_g = 0

Nan in the column psfMag_r = 0

Nan in the column psfMag_i = 0

Nan in the column psfMag_z = 0

Nan in the column zspec = 0

The cleaned table
C:\Users\Shadow\Desktop\demo.fits_noNan.fits
has been generated
Rows:1780
```

Fig. 4 Use of the NaN handling tool. After the definition of the NaN symbols, the user can generate a new dataset only with rows containing NaN elements or another one cleaned by the NaN presence

Data Editing At the PhotoRApToR core is the MLPQNA neural model. 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)* both the training and test data files must contain the same number of input and target columns, and the columns must be 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).

The application makes available a set of specific options to inspect and modify data file entries. Every time a new data table is loaded, a new window shows the complete table properties (Fig. 5), for instance: name, metadata, path and the number of columns and rows.

For a currently loaded table it is possible to select a subset of the needed columns. After the selection, a table subset is created and, if the option *Row Shuffle* is enabled, the subset rows are also randomly shuffled. The random shuffling operation is useful to avoid systematic trends during the training phase and to ensure the homogeneity in the distribution of training and test patterns. This last property is, in fact, directly connected to the necessity to split the initial data into disjoint data sets, to be used for the training and testing phases, respectively. This is a simple action made possible by the *Split* option. When the table is selected in the *Table List*, the user must give two different names for the split files (in this case *train* and *test*) and two different percentages of the original data set. It is important to observe that, generally speaking, in machine learning supervised methods three different subsets for every



SDSS_GALEX_UKIDSS_WISE_all.csv

Name : SDSS_GALEX_UKIDSS_WISE_all.csv

Path : C:\Users\Max\Desktop\PhotoRApToR\TEST\SDSS_GALEX_UKIDSS_WISE_all.csv

Rows : 14289 Columns : 16

	Check	Name	Class
1	☑	UMAG	Float
2	☑	GMAG	Float
3	☑	RMAG	Float
4	☑	IMAG	Float
5	☑	ZMAG	Float
6	☑	nuv_mag_iso	Double

Insert output filenames without extension:

trainset 80 testset 20 ☑ Row Shuffle

SDSS_GALEX_UKIDSS_WISE_all.csv

Row ID	UMAG	GMAG	RMAG	IMAG	ZMAG
0	20.384	20.461	20.324	20.093	20.042
1	19.647	19.465	19.368	19.193	19.015
2	18.228	17.934	17.995	17.934	17.873
3	20.404	20.499	20.247	20.135	20.367
4	19.524	19.295	19.122	19.154	19.066
5	19.479	19.36	19.313	19.138	19.164
6	18.533	17.973	17.811	17.695	17.594
7	20.174	19.9	19.849	19.634	19.596
8	19.421	19.18	18.973	19.059	19.12
9	19.345	19.0	18.922	18.988	18.817
10	20.565	20.1	19.636	19.586	19.558
11	19.936	19.808	19.731	19.57	19.688
12	20.169	20.384	20.063	19.956	20.179
13	18.354	18.145	17.93	17.879	17.788
14	18.9	18.654	18.571	18.581	18.635
15	20.232	19.851	19.574	19.45	19.433

Fig. 5 The main panel showing details about the loaded data table and the editing options

experiment are generally required from the available KB: *(i)* the *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 [8]. In the version of the MLPQNA model implemented 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 [25].

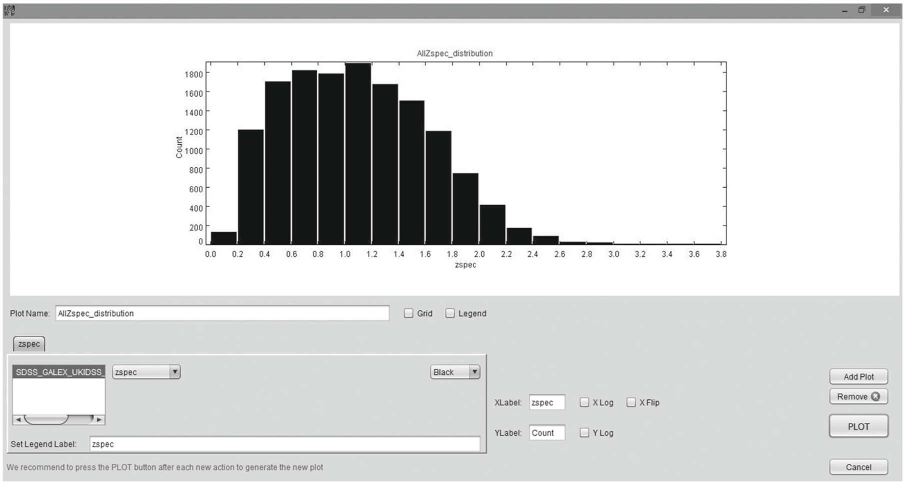
Therefore, before any photo-z experiment, it is needed to split the data set in only two subsets, for instance, the training and test sets. There is no any analytical rule to *a priori* decide the percentages of the splitting operation. According to the direct experience, an empirical rule of thumb suggests to use 80 % and 20 % for training and test sets, respectively [32]. 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). The percentage depends also on the quality of the available KB. When both photometry and spectroscopy are particularly clean and precise, with a high S/N, there could also be possible to obtain high performances by training just on half of the KB.

On the other hand, the more patterns are available for test, the more consistent will be the statistical evaluation of the experiment performances.

Data Plotting Within the PhotoRApToR application there are also instruments, based on STILTS toolset [47], capable to generate different types of plots (some examples are shown in Figs. 6, 7 and 8). These options are particularly suited during the preparation phase of the data for the experiments.

The graphical options selectable by user are:

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



Plot Name: AllZspec_distribution □ Grid □ Legend

zspec

SDSS_GALEX_UKIDSS_ zspec ▾ Black ▾

XLabel: zspec □ X Log □ X Flip

YLabel: Count □ Y Log

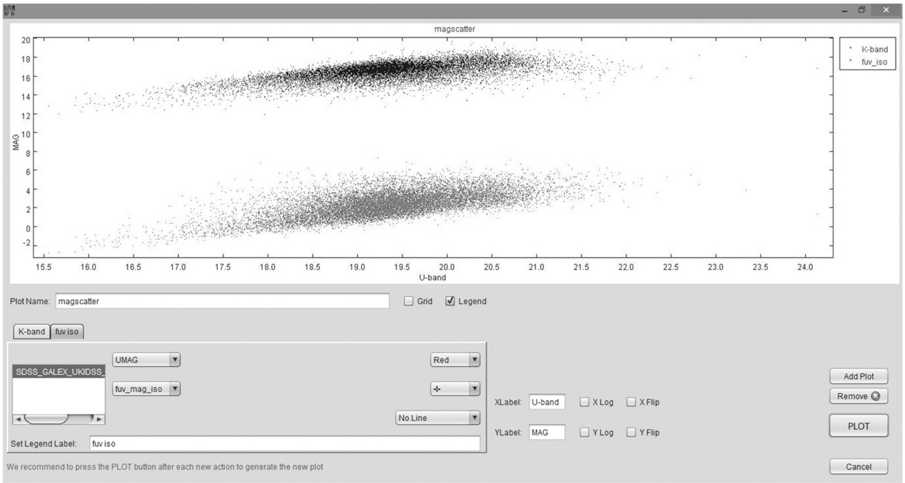
Set Legend Label: zspec

We recommend to press the PLOT button after each new action to generate the new plot

Fig. 6 An example of zspec distribution diagram, showing the options available within the histogram plotting panel

Data Feature Selection Learning by examples stands for a training scheme operating under supervision of an oracle capable to provide the correct, already known, outcome for each of the training sample. This outcome is properly a class or value of the examples and its representation depends on the available KB and on its intrinsic nature even though in most cases it is based on a series of numerical attributes, related to the extracted KB, organized and submitted in an homogeneous way.

Therefore, a fundamental step for any machine learning experiment is to decide which features to use as input attributes for the patterns to be learned. In the



Plot Name: magscatter □ Grid ☑ Legend

K-band fuv iso

SDSS_GALEX_UKIDSS_ UMAG ▾ Red ▾

fuv_mag_iso ▾ + ▾

No Line ▾

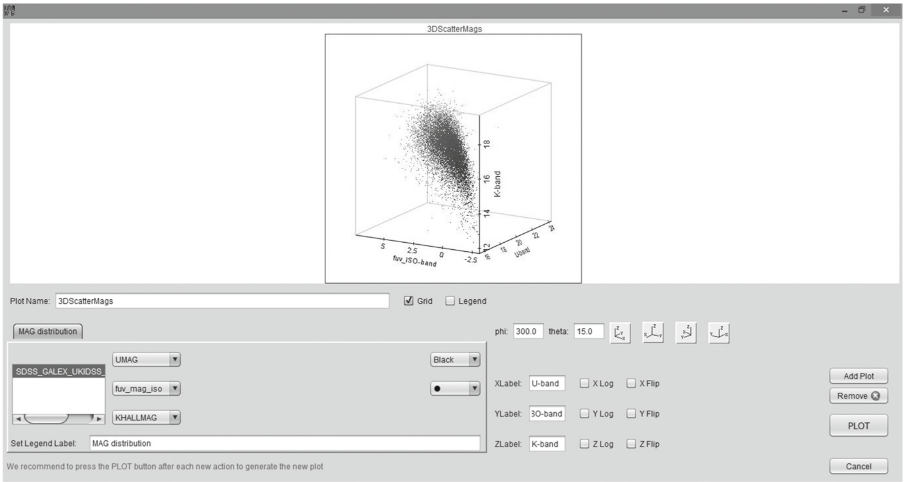
XLabel: U-band □ X Log □ X Flip

YLabel: MAG □ Y Log □ Y Flip

Set Legend Label: fuv iso

We recommend to press the PLOT button after each new action to generate the new plot

Fig. 7 An example of magnitude distributions, showing the options available within the 2D scatter plotting panel



Plot Name: 3DScatterMags ☑ Grid □ Legend

phi: 300.0 theta: 15.0

UMAG
fuv_mag_iso
KHALLMAG
Black
●

XLabel: U-band □ X Log □ X Flip

YLabel: 3O-band □ Y Log □ Y Flip

ZLabel: K-band □ Z Log □ Z Flip

Set Legend Label: MAG distribution

We recommend to press the PLOT button after each new action to generate the new plot

Fig. 8 An example of magnitude distributions, showing the options available within the 3D scatter plotting panel

specific case of photo-z estimation, for a given data sets, it is necessary to inspect and check which types of fluxes (bands) and combinations (magnitudes, colors) is more effective.

In practice, the user must 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 suitable to train a machine learning model. The experience demonstrates, in fact, that it is more the quality of data, than the quantity of features and patterns, the crucial key to obtain the best prediction results [8]. This phase is very time consuming and usually requires many tens or even hundreds of experiments. Of course, the exact number of experiments depends on a variety of factors, among which, the number of photometric bands and magnitudes for which a high quality of zspec entries is available in the KB; the photometric and spectroscopic quality of the data, the type of magnitudes (i.e. aperture, total or isophotal magnitudes, etc.), the completeness of the spectroscopic coverage within the KB and the spectroscopic range. In the authors experience, quite often, the optimal combination turned out to be the feature set obtained from the colors plus one reference magnitude for each region of the electro-magnetic spectrum (broadly divided in UV, optical, Near Infrared, Far Infrared, etc.) [8]. This can be understood by remembering that colors convey more information than the single related magnitudes, since from the basic equation defining magnitudes it is easy to see that a magnitude difference corresponds to a flux ratio and hence in the derived colors an ordering relationship among features is always implicitly assumed.

3.2 Performing experiments

After having prepared the KB, the user should have two subset tables ready to be submitted for a photo-z experiment. By looking at the Fig. 2 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)* Run, i.e. the execution on new data samples of a well trained, validated and tested network.

We outline that for the first two steps, the basic rule is to use disjointed but homogeneous data subsets, because all empirical photo-z methods in general may suffer to extrapolate outside the range of parameter distributions covered by the training. In other words, outside the limits of magnitudes and spectroscopic redshift (zspec) imposed by the training set, these methods do not ensure optimal performances. Therefore, in order to remain in a safe condition, the user must perform a selection of test data according to the training sample limits.

None of the objects included in the training sample should be included in the test sample and, 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: the training of the MLPQNA model network and its 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 computationally stiff is the model during training, the higher would be the risk to fit the noise and other peculiarities of the training sample in the new data [21]. The technique implemented within PhotoRaPToR, i.e. the so called *leave-one-out 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. In this way, validation can be implicitly performed during training, by enabling at setup the standard leave-one-out k-fold cross validation mechanism [25]. The automatized process of the cross-validation consists in performing k different training runs with the following procedure: (i) splitting of the training set into k random subsets, each one composed by the same percentage of the data set (depending on the k choice); (ii) at each run the remaining part of the data set is used for training and the excluded percentage for validation. While avoiding overfitting, the k-fold cross validation leads to an increase of the execution time estimable around $k - 1$ times the total number of runs.

Concerning the photo-z experiment setup, special care must be paid to the setup of the training parameters, because all the other use cases, for instance the Test and Run (i.e. the execution on new data), require only the specification of the proper input data set, and to recall the internal model configuration as it was frozen at the end of training (Fig. 9). We can group the MLPQNA model training parameters into three subsets: *network topology*, *learning rule* setup and *validation* setup.

- **Network topology**. It includes all parameters related to the MLP network architecture;
    - *Number of input neurons*. In terms of input data set it corresponds to the number of columns of the data table, (also named as input features of

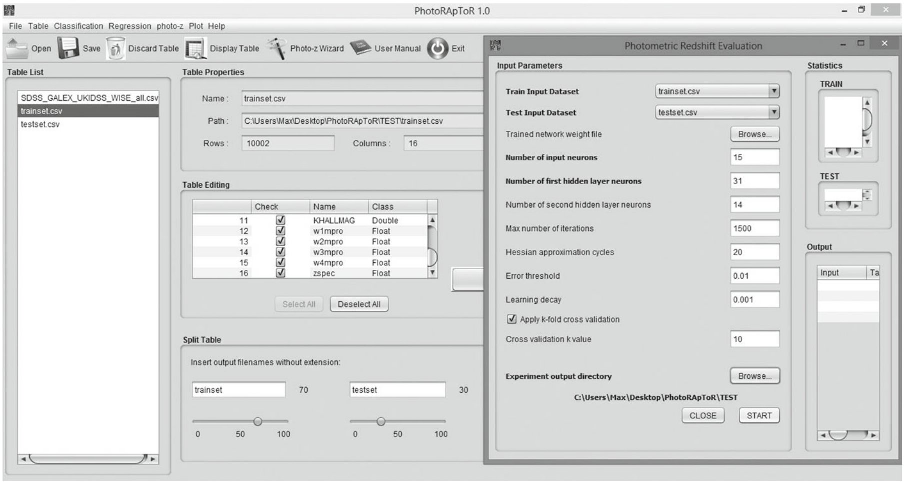


Table List

SDSS_GALEX_UKIDSS_WISE_all.csv
trainset.csv
testset.csv

Table Properties

Name : trainset.csv
Path : C:\Users\Max\Desktop\PhotoRApToR\TEST\trainset.csv
Rows : 10002 Columns : 16

Table Editing

	Check	Name	Class
11	☑	KHALLMAG	Double
12	☑	w1mpro	Float
13	☑	w2mpro	Float
14	☑	w3mpro	Float
15	☑	w4mpro	Float
16	☑	zspec	Float

Split Table

Insert output filenames without extension:

trainset 70 testset 30

0 50 100 0 50 100

Photometric Redshift Evaluation

Input Parameters

Train Input Dataset: trainset.csv
Test Input Dataset: testset.csv
Trained network weight file
Number of input neurons: 15
Number of first hidden layer neurons: 31
Number of second hidden layer neurons: 14
Max number of iterations: 1500
Hessian approximation cycles: 20
Error threshold: 0.01
Learning decay: 0.001
☑ Apply k-fold cross validation
Cross validation k value: 10
Experiment output directory
C:\Users\Max\Desktop\PhotoRApToR\TEST

Statistics

TRAIN

TEST

Output

Input	Ta

Fig. 9 An example of setup phase for a photo-z regression experiment

the data sample, i.e. number of fluxes, magnitudes or 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. More in general, in the case of classification problems, the number of output neurons depends on the number of desired classes;

- *Number of neurons in the first hidden layer*. As a rule of thumb, it is common practice to set this number to $2N + 1$, where N is the number of input neurons. But it can be arbitrarily chosen by the user;
- *Number of neurons in the second hidden layer*. This is an optional parameter. Although not required in normal conditions, as stated by the known universal approximation theorem [18], some problems dealing with a parameter space of very high complexity, 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 [3]. As a rule of thumb, it is reasonable to set this number to $N - 1$, where N is the number of input neurons. But it is strongly suggested to use a number strictly lower than the dimension of the first hidden layer;
- *Number of neurons in the output layer*. This number is obviously forced to 1 for regression problems, while in case of classification this quantity depends on the number of classes as present within the treated problem;
- *Trained network weights*. This parameter is related to the matrix of weights (internal connections among neurons). A weight matrix exists only after having performed one training session at least. Therefore, this parameter is left empty at the beginning of any experiment. But, for all other use cases (Test or Run), it is required to load a previously trained network. However this parameter could also be used to perform

further training cycles for an already trained network (i.e. in case of an incremental learning experiment).

- **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, applying a k-step cycle in which the training error is evaluated and its performances are validated. Reasonable values are between 5 and 10, depending on the amount of training data used. The k-fold cross validation intrinsically tries to avoid overfitting. Nonetheless, in rare cases (such as a wrong choice of the k parameter with respect to the train set dimension), a residual overfitting may occur. Therefore if the user wants to verify it, he/she should simply inspect the results, usually by comparing train with test performance. Whenever training accuracy is much better than test one, this is a typical clue of overfitting presence. Therefore, when cross validation with a proper k choice is enabled, by definition, it should avoid such events. The k parameter choice is not deterministic, but regulated by a rule of thumb, depending on the amount of training patterns. We remind also that this value strongly affects the overall computing time of the experiment.

- **Learning rule setup**. It includes all parameters related to the QNA learning rule;

    - *Maximum number of iterations at each Hessian approximation cycle*. The typical range for such value is [1000, 10000], depending on the best compromise between the requested precision and the complexity of the problem. It can affect the computing time of the training;
    - *Number of Hessian approximation cycles*. Namely the 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 is [20, 60], although also in this case the exact value depends 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 one of the stopping criteria of the algorithm (alternative to the couple of the parameters *iterations* and *cycles*). It is the training error threshold (a value of 0.001 is typical for photo-z experiments).
    - *Learning decay*. This value determines the analytical *stiffness* of the approximation process. It affects the expression of the weight updating law, by adding the term $decay * ||networkweights||^2$. Its range may vary from a minimum value of 0.0001 (very low stiffness) up to 1000.0

(very high stiffness). Also in this case if a very low value is adopted, it is recommended to enable the cross validation option (see below), to prevent overfitting occurrence.

The error calculated by the MLPQNA model during the training is evaluated for all the presented input patterns in terms of the difference between the known target values and the calculated outputs of the model. The error function in the regression case is based on the Least Mean Square (LSE) + Tychonov regularization [26]. This function is defined as follows:

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

where N is the number of input patterns, y and t are the network output and the pattern target respectively, $decay$ is the decay input parameter and $||W||$ the norm of 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 the generalization error of the trained neural network can be improved, and the training can be accelerated. If the best regularization parameter $decay$ is unknown, it could be experimented by varying its value within the allowed range, from a weak up to the strong regularization. In order to achieve the weight decay rule, we internally minimize a more complex merit function:

$$f = E + \frac{decay * S}{2}$$

Here E is the training error, S is the sum of the squares of the network weights, and the decay coefficient $decay$ 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 parameter $decay$, from the lower value (no regularization) to the infinite value (strongest regularization). By inspecting statistical results at each stage of the procedure the overfitting tendency can be monitored by continuously changing the decay factor. A zero decay usually corresponds to an overfitted network. Very large decay means instead an underfitted network. Between these extreme values there is a range of networks which reproduce the dataset with different degrees of precision and smoothness.

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 the respective input data sets.

3.3 Inspection of results

Interpolative methods, such as MLPQNA, have the advantage that the training set is made up of real objects. In this sense, any empirical method intrinsically takes into account effects such as the filter band-pass and flux calibrations, even though the difficulty in extrapolating to regions of the input parameter space not well sampled in the training data is one of the main drawbacks [16].

This is why a strong requirement of empirical methods is that the training set must be large enough to cover properly the parameter space in terms of colors, magnitudes, object types and redshift. If this is true, then the calibrations and corresponding uncertainties are well known and only limited extrapolations beyond the observed locus in color-magnitude space are required. Hence, 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.

All individual experiments should be evaluated in a consistent and objective manner through an homogeneous set of statistical indicators. We remark that all statistical results reported throughout this paper are referred to the blind 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 any of the training or validation phases. As easy to understand, the combination of test and training data might introduce a straightforward systematic bias which could mask reality.

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 statistical indicator in the specific case of photometric redshift prediction.

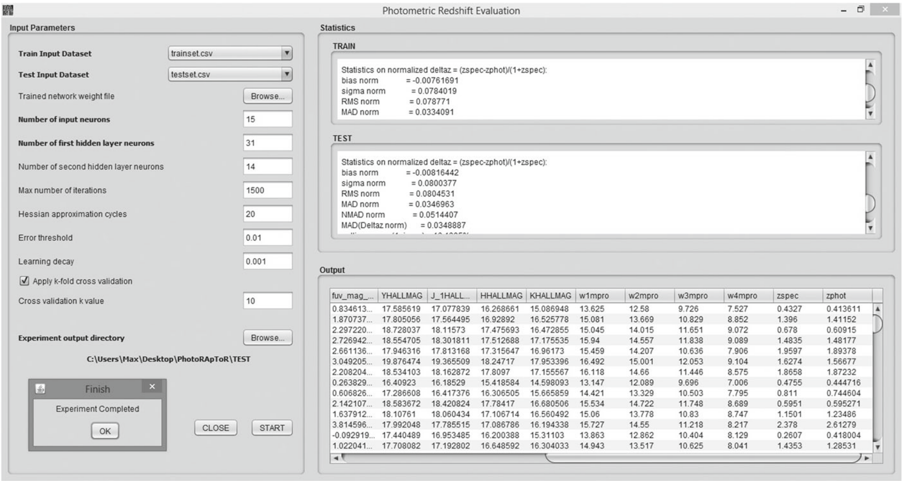
The velocity dispersion error, intrinsically present within the photometric estimation, is not uniform over a wide range of spectroscopic redshift and therefore the related statistics is not able to give a consistent estimation. On the contrary, the normalized term Δz_{norm} introduces a more uniform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

For what the analysis of the catastrophic outliers is concerned, according to [35], the parameter $D_{95} \equiv \Delta_{95}/\left(1 + z_{phot}\right)$ 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 Δz_{norm} 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 $|\Delta z_{norm}| > 2\sigma\ (\Delta z_{norm})$. This although it is common practice to indicate as outliers all objects with $|\Delta z_{norm}| > 0.15$, (thus included in the provided statistics).

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, the quantity $NMAD$ is smaller 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.

All the described statistical indicators are provided by PhotoRaPToR as the output of every photo-z estimation test and are stored in specific files (by default named as *test_statistics.txt*). For completeness we also provide a similar statistics file as the output of any training session (Fig. 10). But its use is only to allow a quick comparison between training and test, just in order to verify the absence of any overfitting occurrence.

Besides the statistics files, PhotoRApToR makes also available some graphical tools, useful to perform a visual inspection of photo-z experiments. In particular a $2D$ scatter plot to show the trend of photo-z vs zspec (Fig. 11), as well as a set of histograms useful to graphically evaluate the distributions of quantities Δz and Δz_{norm}.



Photometric Redshift Evaluation

Input Parameters

Train Input Dataset: trainset.csv

Test Input Dataset: testset.csv

Trained network weight file

Number of input neurons: 15

Number of first hidden layer neurons: 31

Number of second hidden layer neurons: 14

Max number of iterations: 1500

Hessian approximation cycles: 20

Error threshold: 0.01

Learning decay: 0.001

☑ Apply k-fold cross validation

Cross validation k value: 10

Experiment output directory

C:\Users\Max\Desktop\PhotoRApToR\TEST

Finish

Experiment Completed

Statistics

TRAIN

```
Statistics on normalized deltaz = (zspec-zphot)/(1+zspec):
bias norm        = -0.00761691
sigma norm         = 0.0784019
RMS norm          = 0.078771
MAD norm          = 0.0334091
```

TEST

```
Statistics on normalized deltaz = (zspec-zphot)/(1+zspec):
bias norm        = -0.00816442
sigma norm         = 0.0800377
RMS norm          = 0.0804531
MAD norm          = 0.0346963
NMAD norm          = 0.0514407
MAD(Deltaz norm)     = 0.0348887
```

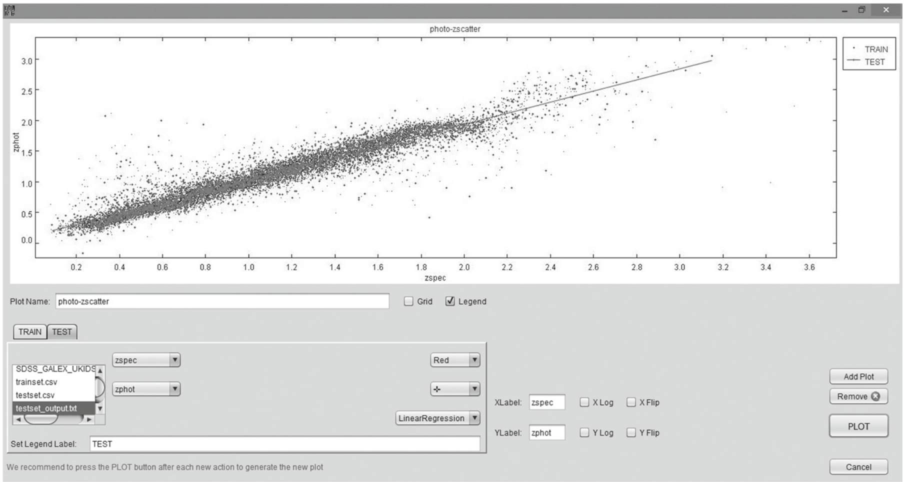
Output

fuv_mag_...	YHALLMAG	J_1HALL...	HHALLMAG	KHALLMAG	w1mpro	w2mpro	w3mpro	w4mpro	zspec	zphot
0.834613...	17.585619	17.077839	16.268661	15.086948	13.625	12.58	9.726	7.527	0.4327	0.413611
1.870737...	17.805056	17.564495	16.92892	16.525778	15.081	13.669	10.829	8.852	1.396	1.41152
2.297220...	18.728037	18.11573	17.475693	16.472855	15.045	14.015	11.651	9.072	0.678	0.60915
2.726942...	18.554705	18.301811	17.512688	17.175535	15.94	14.557	11.838	9.089	1.4835	1.48177
2.661136...	17.946316	17.813168	17.315647	16.96173	15.459	14.207	10.636	7.906	1.9597	1.89378
3.049205...	19.876474	19.365509	18.24717	17.953396	16.492	15.001	12.053	9.104	1.6274	1.56677
2.208204...	18.534103	18.162872	17.8097	17.155567	16.118	14.66	11.446	8.575	1.8658	1.87232
0.263829...	16.40923	16.18529	15.418584	14.598093	13.147	12.089	9.696	7.006	0.4755	0.444716
0.606826...	17.286608	16.417376	16.306505	15.665859	14.421	13.329	10.503	7.795	0.811	0.744604
2.142107...	18.583672	18.420824	17.78417	16.680506	15.534	14.722	11.748	8.689	0.5951	0.595271
1.637912...	18.10761	18.060434	17.106714	16.560492	15.06	13.778	10.83	8.747	1.1501	1.23486
3.814596...	17.992048	17.785515	17.086786	16.194338	15.727	14.55	11.218	8.217	2.378	2.61279
-0.092919...	17.440489	16.953485	16.200388	15.31103	13.863	12.862	10.404	8.129	0.2607	0.418004
1.022041...	17.708082	17.192802	16.648592	16.304033	14.943	13.517	10.625	8.041	1.4353	1.28531

Fig. 10 The statistics produced at the end of a photo-z regression experiment. The training and test results are also automatically stored in the files *train_statistics.txt* and *test_statistics.txt*, respectively

4 Other functionalities

To complete the description of the resources made available by PhotoRApToR, we wish to stress that besides photometric redshift estimation (to be intended as a specific type of regression experiment), the user has the possibility to perform generic regression as well as multi-class classification experiments.



Plot Name: photo-zscatter ☐ Grid ☑ Legend

zspec

zphot

Red

LinearRegression

Set Legend Label: TEST

XLabel: zspec ☐ X Log ☐ X Flip

YLabel: zphot ☐ Y Log ☐ Y Flip

We recommend to press the PLOT button after each new action to generate the new plot

Fig. 11 The photo-z vs zspec plot as produced after a photo-z regression experiment. In this example the diagram shows both training (*black dots*) and test (*gray crosses*) objects, although the blind test objects are the most relevant to evaluate the prediction performances

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

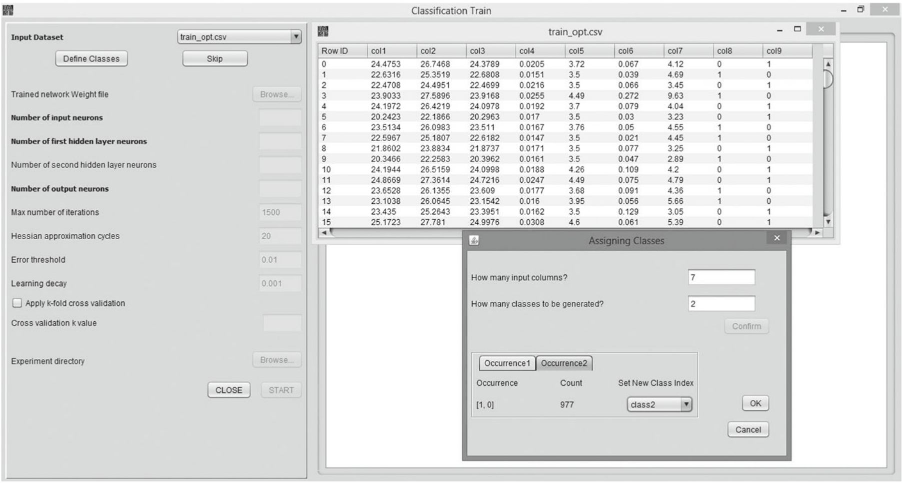
$$\begin{aligned} \Delta out &= target - output \\ \Delta out_{norm} &= \frac{target - output}{1 + target} \end{aligned}$$

Also 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 (Fig. 12), there are two specific options, not foreseen for regression problems:

- *Output neurons*. The number of neurons of the output layer (which is forced to be 1 in the regression experiments), in this case corresponds to the number of different classes present in the training sample. It is required that the class identifiers should have a binary format label. For instance, in a three-class problem, the target classes are represented in three columns labeled respectively, as (100), (010) and (001);
- *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 Cross Entropy (CE) error function was introduced to address classification problem evaluation in a consistent statistical fashion [43]. The CE method consists of two phases: *(i)* it generates a random data sample (trajectories, vectors, etc.) according to a specified mechanism; *(ii)* it updates the parameters of the random mechanism based on the data to produce a *better* sample in the next iteration.



Input Dataset: train_opt.csv

Trained network Weight file

Number of input neurons

Number of first hidden layer neurons

Number of second hidden layer neurons

Number of output neurons

Max number of iterations: 1500

Hessian approximation cycles: 20

Error threshold: 0.01

Learning decay: 0.001

□ Apply k-fold cross validation

Cross validation k value

Experiment directory

Row ID	col1	col2	col3	col4	col5	col6	col7	col8	col9
0	24.4753	26.7468	24.3789	0.0205	3.72	0.067	4.12	0	1
1	22.6316	25.3519	22.6808	0.0151	3.5	0.039	4.69	1	0
2	22.4708	24.4951	22.4699	0.0216	3.5	0.066	3.45	0	1
3	23.9033	27.5896	23.9168	0.0255	4.49	0.272	9.63	1	0
4	24.1972	26.4219	24.0978	0.0192	3.7	0.079	4.04	0	1
5	20.2423	22.1866	20.2963	0.017	3.5	0.03	3.23	0	1
6	23.5134	26.0983	23.511	0.0167	3.76	0.05	4.55	1	0
7	22.5967	25.1807	22.6182	0.0147	3.5	0.021	4.45	1	0
8	21.8602	23.8834	21.8737	0.0171	3.5	0.077	3.25	0	1
9	20.3466	22.2583	20.3962	0.0161	3.5	0.047	2.89	1	0
10	24.1944	26.5159	24.0998	0.0188	4.26	0.109	4.2	0	1
11	24.8669	27.3614	24.7216	0.0247	4.49	0.075	4.79	0	1
12	23.6528	26.1355	23.609	0.0177	3.68	0.091	4.36	1	0
13	23.1038	26.0645	23.1542	0.016	3.95	0.056	5.66	1	0
14	23.435	25.2643	23.3951	0.0162	3.5	0.129	3.05	0	1
15	25.1723	27.781	24.9976	0.0308	4.6	0.061	5.39	0	1

How many input columns? 7

How many classes to be generated? 2

Occurrence	Count	Set New Class Index
[1, 0]	977	class2

Fig. 12 The setup panel of a multi-class classification experiment. It is also possible to assign arbitrary class labels to all output instances in the training and test sets (see subpanel *Assigning Classes*)

In practice, a data model is created based on the training set, and its CE is measured on a test set to assess how accurately the model is predicting the test data. The method compares indeed two probability distributions, p the true distribution of data in the data set, 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:

$$H\left(T,q\right) = -\sum_{i=1}^{N} \frac{1}{N} log_2 q\left(x_i\right)$$

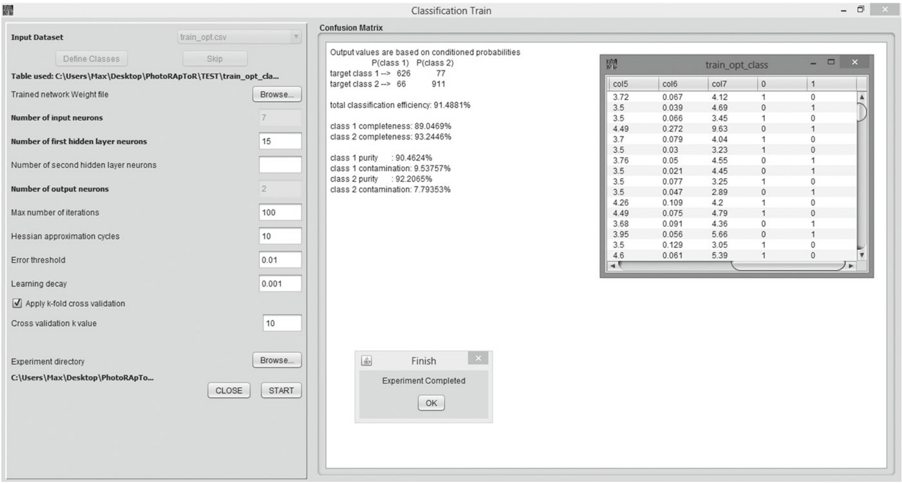
where T is the chosen training set, corresponding to the true distribution p, N is the number of objects in the test set, and $q(x)$ is the probability of the event x as estimated from the training set.

Another difference with respect to 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 used to easily visualize the classification performance [40]: each column of the matrix represents the instances in a predicted class, while each row represents the instances in the real class (Fig. 13). One benefit of a confusion matrix is the simple way in which it allows to see whether the system is mixing different classes or not.

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:



Classification Train

Input Dataset: train_opt.csv

Table used: C:\Users\Max\Desktop\PhotoRApToR\TEST\train_opt_cla...

Trained network Weight file

Number of input neurons: 7

Number of first hidden layer neurons: 15

Number of second hidden layer neurons:

Number of output neurons: 2

Max number of iterations: 100

Hessian approximation cycles: 10

Error threshold: 0.01

Learning decay: 0.001

☑ Apply k-fold cross validation

Cross validation k value: 10

Experiment directory

C:\Users\Max\Desktop\PhotoRApTo...

Confusion Matrix

```
Output values are based on conditioned probabilities
                 P(class 1)   P(class 2)
target class 1 -->  626          77
target class 2 -->  66          911

total classification efficiency: 91.4881%

class 1 completeness: 89.0469%
class 2 completeness: 93.2446%

class 1 purity      : 90.4624%
class 1 contamination: 9.53757%
class 2 purity      : 92.2065%
class 2 contamination: 7.79353%
```

train_opt_class

col5	col6	col7	0	1
3.72	0.067	4.12	1	0
3.5	0.039	4.69	0	1
3.5	0.066	3.45	1	0
4.49	0.272	9.63	0	1
3.7	0.079	4.04	1	0
3.5	0.03	3.23	1	0
3.76	0.05	4.55	0	1
3.5	0.021	4.45	0	1
3.5	0.077	3.25	1	0
3.5	0.047	2.89	0	1
4.26	0.109	4.2	1	0
4.49	0.075	4.79	1	0
3.68	0.091	4.36	0	1
3.95	0.056	5.66	0	1
3.5	0.129	3.05	1	0
4.6	0.061	5.39	1	0

Finish

Experiment Completed

Fig. 13 The statistics produced at the end of a 2-class classification experiment

- 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 the misclassified objects 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.

The MLPQNA machine learning method, embedded into PhotoRaptor, has been already tested in several classification cases. In [7], we compared the performances of MLPQNA with other machine learning based classifiers and traditional techniques as well, in terms of accuracy of identifying candidate globular clusters in the NGC 1399 HST single-band data. In Cavuoti et al., [15], we compared MLPQNA with standard MLP and Support Vector Machine to photometrically classify AGNs in the SDSS DR4 archive. Finally, we recently have exploited the MLPQNA to perform classification experiments within SDSS DR10 archive, aimed at photometrically identifying quasars from the whole sample including also galaxies and stars, as well as to verify the possibility to disentangle normal galaxies from objects with a peculiar spectrum, [11].

5 Comparison with public machine learning tools

We performed a simple comparison between PhotoRApToR and an alternative machine learning tool publicly available: the scikit-learn toolset [38]. The comparison is based on the photo-z estimation by means of a supervised non-linear regression experiment, by directly comparing the statistical performances between the MLPQNA model provided through PhotoRApToR and the widely known ensemble method based on Random Forest [5], which uses a random subset of candidate data features to build an ensemble of decision trees.

The data set used for the experiment was obtained by merging the photometry from four different surveys (UKIDSS, SDSS, GALEX and WISE), including derived colors and reference magnitudes for each band as internal features, thus covering a wide range of wavelengths from the UV to the mid-infrared. While the spectroscopic redshifts, (i.e. the zspec target values) were derived from selected quasars of the SDSS-DR7 database. The complete KB consisted of $\sim 1.4 \times 10^4$ objects, from which the 60 % used as training set and the residual 40 % as blind test set (see [8], for more details). We remark also that in that case, our MLPQNA has been directly compared with other several photo-z estimation methods (see references therein), achieving best results.

After having trained the two ML models with the same training set, their photo-z estimation results have been compared in terms of statistics and residual analysis (outlier percentages). The results are shown in Fig. 14 and reported in Table 1. From the comparison, it results apparent that MLPQNA performs better than Random Forest, especially in the high-redshift zone (i.e. at $zspec > 2.0$), showing a more robust prediction capability also in the sparsely populated regions of the parameter space.

In addition, unlike the PhotoRApToR resource, in order to setup and run the Random Forest model provided by the scikit-learn package, as well as to prepare and execute the experiments, some manipulations of the source code have been necessary. The reason is that the scikit-learn package is provided as a library to be imported in a user-defined script code, which implies a certain knowledge of the Python programming language.

Although we reported a use case example where PhotoRaptor has been tested on a dataset of about 10^4 samples, we want to emphasize that the reliability of our resource has been already verified for data sets up to $\sim 10^6$ samples. However, in such cases the computational cost of the experiment becomes very high, although the regression accuracy does not seem to require such amount of data in the training set. Therefore, as general rule of thumb, a good compromise between computational time and performance could be to limit the training sample to about 10^5 samples.

In addition, our model MLPQNA has been tested in a public photo-z contest (PHAT1, [27], and [14]), resulting as one of the best interpolative methods. In another work [10] we published a catalogue of photometric redshifts for the SDSS DR9 release, by comparing our prediction accuracy with other machine learning methods. More recently PhotoRaptor has been used by an independent group [29], that performed a regression feature analysis with SDSS DR10 galaxies by comparing our resource with random forest [22] and FANN artificial neural networks [36].

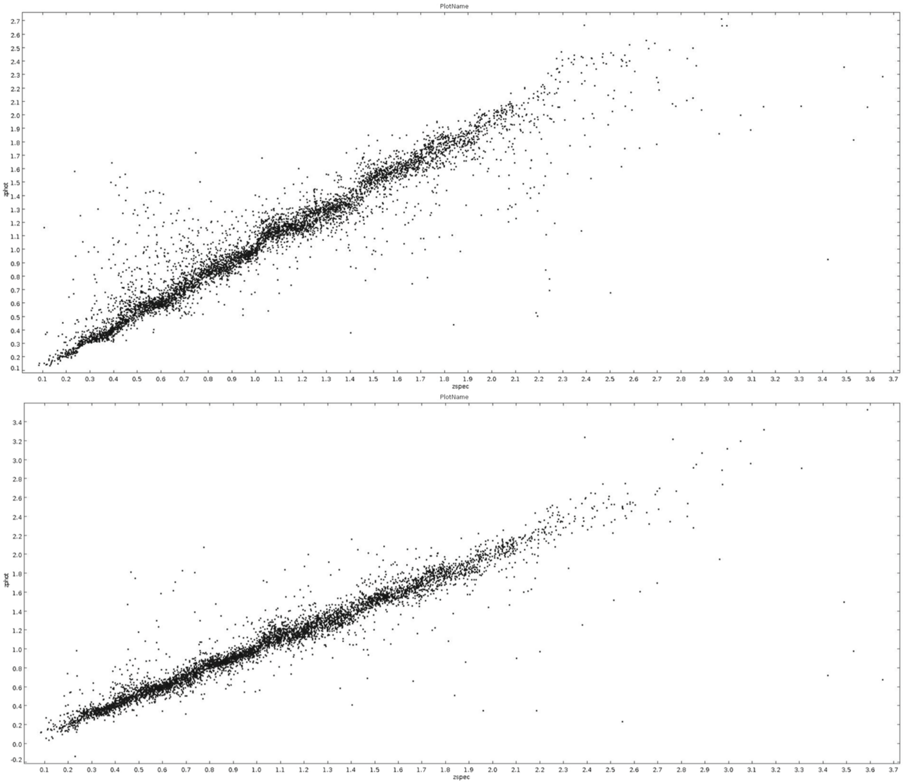


Fig. 14 The photo-z vs zspec scatter plots as produced after the photo-z estimation experiment. The upper plot refers to the Random Forest model while the lower one is related to the MLPQNA model results. Both diagrams show the distributions of the $\sim 5.7 \times 10^3$ objects composing the blind test set

Table 1 Comparison of the performances among the different tools

Photo-z Estimation Statistics			Δz_{norm}		
Model	$BIAS$	σ	MAD	RMS	$NMAD$
PhotoRApToR (MLPQNA)	0.004	0.069	0.020	0.069	0.029
Scikit (Random Forest)	0.009	0.083	0.021	0.084	0.031
Outlier percentages [%]			$\lvert\Delta z_{norm}\rvert$		
Model	> 0.15	$> 1\sigma$	$> 2\sigma$	$> 3\sigma$	$> 4\sigma$
PhotoRApToR (MLPQNA)	2.43	9.39	2.89	1.40	0.91
Scikit (Random Forest)	5.27	11.03	4.48	2.31	1.34

MLPQNA is the ML engine of our application, based on a four-layers neural network, while Random Forest is the ML model provided by the scikit-learn public resource. Both methods were trained on the multi-survey mixed (colors + reference magnitudes) dataset, obtained by cross-matching photometry of UKIDSS, SDSS, GALEX and WISE surveys. The reported statistics is related to the photo-z estimation on the blind test set of about $\sim 5.7 \times 10^3$ QSO objects. For the definition of the parameters and for discussion see text.

6 Perspective and conclusions

Driven by the advances in the digital detectors and computing technology, astronomy has become an immensely data-rich science. This exponential data avalanche continues. It enables some exciting new science, but poses many non-trivial challenges that are common to many other data-driven fields. Nowadays the technological evolution of astronomical instruments has been so fast to render physically impossible to move the data from their original repositories. The real goal of science, namely data analysis and knowledge discovery, begins after all the data processing and data delivery through the archives. This requires some powerful new approaches to data exploration and analysis, leading to knowledge discovery and understanding. This implies that, as it has always been asked for but never implemented, we must be able to move the programs and not the data. Therefore, the future of any data-driven service depends on the capability and possibility of moving the data mining applications to the data centers hosting the data themselves. In such scenario, PhotoRApToR represents our test bench of a desktop application prototype capable to fulfill this concept. As a final perspective, we want to address the still open problem to find an efficient, reliable and standard way to provide single photo-z errors in the case of interpolative methods. We have recently started to investigate such problem and intend to improve PhotoRaptor in the next future with such kind of a tool.

Acknowledgments The authors wish to thank the anonymous referee for all very useful comments and suggestions. MB wishes to thank the financial support of the 7th European Framework Programme for Research Grant FP7-SPACE-2013-1, *VIALACTEA - The Milky Way as a Star Formation Engine*. 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 *Cosmology with the Euclid Space Mission*.

References

1. Albrecht, A., Bernstein, G., Cahn, R., et al.: Report of the Dark Energy Task Force (2006)
2. ANSI (American National Standards Institute) et al.: American National Standard Code for Information Interchange. The Institute (1977)
3. Bengio, Y., LeCun, J.: In Large-Scale Kernel Machines. MIT Press (2007)
4. Biviano, A., et al.: CLASH-VLT: The mass, velocity-anisotropy, and pseudo-phase-space density profiles of the z = 0.44 cluster, galaxy MACS 1206.2-0847. A&A **558**(A1), 22 (2013)
5. Breiman, L.: Random Forests. Machine Learning, Springer Eds., **45**(1), 25–32 (2001)
6. Brescia, M.: New Trends in E-Science: Machine Learning and Knowledge Discovery in Databases, 78 pages, Horizons in Computer Science Research. In: Clary, T.S. (ed.) Series Horizons in Computer Science. ISBN: 978-1-61942-774-7, vol. 7. Nova Science Publishers (2012)
7. Brescia, M., Cavuoti, S., Paolillo, M., Longo, G., Puzia, T.: The detection of Globular Clusters in galaxies as a data mining problem. MNRAS **421**(2), 1155 (2012)
8. Brescia, M., Cavuoti, S., D'Abrusco, R., Longo, G., Mercurio, A.: Photometric redshifts for Quasars in multi band Surveys. ApJ **772**, 140 (2013)
9. Brescia, M., et al.: DAMEWARE: A web cyberinfrastructure for astrophysical data mining. PASP **126**(942), 783–797 (2014)
10. Brescia, M., Cavuoti, S., De Stefano, V., Longo, G.: A catalogue of photometric redshifts for the SDSS-DR9 galaxies. A&A **568**, A126 (2014)
11. Brescia, M., Cavuoti, S., Longo, G.: Automated physical classification in the SDSS DR10. A catalogue of candidate Quasars, MNRAS, accepted (in press) (2015)

12. Byrd, R.H., Nocedal, J., Schnabel, R.B.: Math. Program. **63**, 129 (1994)
13. Capozzi, D., De Filippis, E., Paolillo, M., D'Abrusco, R., Longo, G.: The properties of the heterogeneous Shakhbazyan groups of galaxies in the SDSS. Mon. Not. R. Astron. Soc. **396**(2), 900–917 (2009)
14. Cavuoti, S., Brescia, M., Longo, G., Mercurio, A.: Photometric redshifts with Quasi Newton Algorithm (MLPQNA). Results in the PHAT1 contest. A&A **546**(A13), 1–8 (2012)
15. Cavuoti, S., Brescia, M., D'Abrusco, R., Longo, G., Paolillo, M.: Photometric classification of emission line galaxies with machine-learning methods. MNRAS **437**(1), 968–975 (2014)
16. Collister, A.A., Lahav, O.: ANNz: Estimating Photometric Redshifts Using Artificial Neural Networks. PASP **116**, 345 (2004)
17. Connolly, A.J., Csabai, I., Szalay, A.S., Koo, D.C., Kron, R.G., Munn, J.A.: Slicing Through Multicolor Space: Galaxy Redshifts from Broadband Photometry. Astron. J. **110**, 2655 (1995)
18. Cybenko, G.: Approximations by superpositions of sigmoidal functions. Math. Control Signals Syst. **2**, 303 (1989)
19. The Dark Energy Survey Collaboration, The Dark Energy Survey, White Paper submitted to the Dark Energy Task Force, 42 pages. arXiv:0510346 (2005)
20. Davidon, W.C.: SIAM Journal on Optimization (1991)
21. Dietterich, T.: Overfitting and Undercomputing in Machine Learning. Comput. Surv. **27**, 326 (1995)
22. Drucker, H.: Improving regressors using boosting techniques. In: Proceedings of ICML97, pp. 107–115. Morgan Kaufmann Publishers Inc., San Francisco (1997)
23. Euclid Red Book, ESA Technical Document, ESA/SRE(2011)12, Issue 1.1. arXiv:1110.3193 (2011)
24. Farrow, D.J., et al.: Pan-STARRS1: Galaxy clustering in the Small Area Survey 2. MNRAS **437**, 748–770 (2014)
25. Geisser, S.: The predictive sample reuse method with applications. J. Am. Stat. Assoc. **70**(350), 320-328 (1975)
26. Groetsch, C.V.: The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind, Pitman, Boston (1984)
27. Hildebrandt, H., Arnouts, S., Capak, P., Wolf, C., et al. A&A **523**, 31 (2010)
28. Hoaglin, D.C., Mosteller, F., Tukey, J.W.: Understanding Robust and Exploratory Data Analysis. Wiley C, New York (1983)
29. Hoyle, B., Rau, M.M., Zitlau, R., Seitz, S., Weller, J.: Feature importance for machine learning redshifts applied to SDSS galaxies, Sumitted to MNRAS. arXiv:1410.4696 (2014)
30. Ilbert, O., Capak, P., Salvato, M., et al.: Cosmos Photometric Redshifts with 30-bands for $2 - deg^2$. Astrophys. J. **690**, 1236 (2009)
31. Ivezic, Z., et al.: (the LSST team), The LSST Science Book, v2.0 (2009)
32. Kearns, M. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.): A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for Training-Test Split, Neural Information Processing 8, pp. 183–189. Morgan Kaufmann (1996)
33. Laureijs, R., et al.: Euclid Definition Study Report, ESA/SRE(2011)12, Issue 1.1 (2011)
34. Marlin, B.M.: Missing data problems in machine learning. Library and Archives, Canada (2008)
35. Mobasher, B., Capak, P., Scoville, et al.: Photometric Redshifts of Galaxies in COSMOS. Astrophys. J. Suppl. Ser. **172**(1), 117–131 (2007)
36. Nissen, S.: Implementation of a Fast Artificial Neural Network Library. Technical Report. Department of Computer Science University of Copenhagen (DIKU) (2003)
37. Peacock, J.A., Schneider, P., Efstathiou, G., et al.: ESA-ESO Working Group on Fundamental Cosmology, ESA-ESO Working Group on Fundamental Cosmology. Tech. Rep. (2006)
38. Pedregosa, F., et al.: Scikit-learn: Machine Learning in Python. JMLR **12**, 2825–2830 (2011)
39. Pennebaker, W.B., Mitchell, J.L.: JPEG still image data compression standard, (3rd ed.) (1993)
40. Provost, F., Fawcett, T., Kohavi, R.: The Case Against Accuracy Estimation for Comparing Induction Algorithms. In: Kaufmann, M. (ed.) Proceedings of the 15th International Conference on Machine Learning, pp. 445–553 (1998)
41. Repici, J.: How To: The Comma Separated Value (CSV) File Format. Creativyst Inc (2010)
42. Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington (1961)
43. Rubinstein, R.Y., Kroese D.P.: The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning. Springer, New York (2004)

44. Serjeant, S.: Up to 100,000 Reliable Strong Gravitational Lenses in Future Dark Energy Experiments. ApJ **793**(1), L10 (2014)
45. Stehman, S.V.: Selecting and interpreting measures of thematic classification accuracy. Remote Sens. Environ. **62**(1), 77–89 (1997)
46. Tagliaferri, R., Longo, G., Andreon, S., Capozziello, S., Donalek, C., Giordano, G.: Neural Networks and Photometric Redshifts, Neural Nets. Lect. Notes Comput. Sci **2859**, 226–234 (2002)
47. Taylor, M.B.: STILTS - A Package for Command-Line Processing of Tabular Data . Proceedings of the Astronomical Data Analysis Software and Systems XV ASP Conference Series **351**, 666 (2006)
48. Umetsu, K., Medezinski, E., Nonino, M., et al.: CLASH: Mass Distribution in and around MACS J1206.2-0847 from a Full Cluster Lensing Analysis. ApJ **755**(1), 56 (2012)
49. Wells, D.C., Greisen, E.W., Harten, R.H.: FITS: a Flexible Image transport System. Astron. Astrophys. Supplement Series **44**, 363 (1981)